Step 1 - Install BERT and necessary libraries

In [1]:

```
!pip install bert-for-tf2
!pip install sentencepiece
```

```
Collecting bert-for-tf2
  Downloading bert-for-tf2-0.14.9.tar.gz (41 kB)
     ---------------------------------------- 41.2/41.2 kB ? eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting py-params>=0.9.6
  Downloading py-params-0.10.2.tar.gz (7.4 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting params-flow>=0.8.0
  Downloading params-flow-0.8.2.tar.gz (22 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-pa
ckages (from params-flow>=0.8.0->bert-for-tf2) (1.20.3)
Requirement already satisfied: tqdm in c:\programdata\anaconda3\lib\site-pac
kages (from params-flow>=0.8.0->bert-for-tf2) (4.62.3)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site
-packages (from tqdm->params-flow>=0.8.0->bert-for-tf2) (0.4.4)
Building wheels for collected packages: bert-for-tf2, params-flow, py-params
  Building wheel for bert-for-tf2 (setup.py): started
  Building wheel for bert-for-tf2 (setup.py): finished with status 'done'
  Created wheel for bert-for-tf2: filename=bert_for_tf2-0.14.9-py3-none-any.
whl size=30534 sha256=78f961ce7836ca4f75f655bdb95df2ec57477a8d1f629e38e81b90
d00c6a356d
  Stored in directory: c:\users\pratik\appdata\local\pip\cache\wheels\6f\c7
\91\f2b2c2b3cec30578c5de7c27ac99659a2013501dd66e7e3db0
  Building wheel for params-flow (setup.py): started
  Building wheel for params-flow (setup.py): finished with status 'done'
  Created wheel for params-flow: filename=params_flow-0.8.2-py3-none-any.whl
size=19473 sha256=494fe7d66a4baf52ef76a93afb88ad800b4ec9fb03987981dd35976f0e
f34170
  Stored in directory: c:\users\pratik\appdata\local\pip\cache\wheels\be\17
\6c\5c924411a614ee0a74b2dc4f04c9e61dacc4e60fe9854f4f70
  Building wheel for py-params (setup.py): started
  Building wheel for py-params (setup.py): finished with status 'done'
  Created wheel for py-params: filename=py_params-0.10.2-py3-none-any.whl si
ze=7912 sha256=68feda6b6429b70c8713c63ccf3d51031df7dc8558f6041c923f12fe82782
322
  Stored in directory: c:\users\pratik\appdata\local\pip\cache\wheels\29\ff
\b1\77192657c3311dcae256412a7f36f73b064ace9c98312f5347
Successfully built bert-for-tf2 params-flow py-params
Installing collected packages: py-params, params-flow, bert-for-tf2
Successfully installed bert-for-tf2-0.14.9 params-flow-0.8.2 py-params-0.10.
2

[notice] A new release of pip available: 22.1.2 -> 22.2.2
[notice] To update, run: python.exe -m pip install --upgrade pip

WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
```

```
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
```

```
Collecting sentencepiece
  Downloading sentencepiece-0.1.97-cp39-cp39-win_amd64.whl (1.1 MB)
     ---------------------------------------- 1.1/1.1 MB 18.1 MB/s eta 0:00:
00
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.97

[notice] A new release of pip available: 22.1.2 -> 22.2.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
```

```
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
```

Step 2 - Set for tensorflow 2.0

In [6]:

```
!pip install tensorflow
```

```
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\sit
e-packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda
3\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3
\lib\site-packages)
```

In [36]:

```
# try:
#     %tensorflow_version 2.x
# except Exception:
#     pass
import tensorflow as tf

# import tensorflow_hub as hub

# from tensorflow.keras import layers
# import bert
```

In [4]:

```
% tensorflow_version 2.x
```

```
UsageError: Line magic function `%` not found.
```

As we are going to work on tensorflow 2.0, we need to set it to the required one.

Step 3 - Import the necessary libraries

In [2]:

```
!pip install tensorflow_hub
```

```
Collecting tensorflow_hub
  Downloading tensorflow_hub-0.12.0-py2.py3-none-any.whl (108 kB)
     -------------------------------------- 108.8/108.8 kB 6.6 MB/s eta 0:0
0:00
Requirement already satisfied: numpy>=1.12.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow_hub) (1.20.3)
Requirement already satisfied: protobuf>=3.8.0 in c:\programdata\anaconda3\l
ib\site-packages (from tensorflow_hub) (3.19.6)
Installing collected packages: tensorflow_hub
Successfully installed tensorflow_hub-0.12.0

[notice] A new release of pip available: 22.1.2 -> 22.2.2
[notice] To update, run: python.exe -m pip install --upgrade pip

WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
WARNING: Ignoring invalid distribution -rkupsafe (c:\programdata\anaconda3\l
ib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-
packages)
WARNING: Ignoring invalid distribution -arkupsafe (c:\programdata\anaconda3
\lib\site-packages)
```

In [3]:

```
from tensorflow.keras import layers
import bert
import pandas as pd
import tensorflow_hub as hub
import re
```

In [ ]:

```
```

Step 4 - Load the Dataset

In [4]:

```
reviews_data = pd.read_csv("C:\\Users\\Pratik\\Downloads\\IMDB Dataset.csv\\IMDB Dataset.cs

reviews_data.isnull().values.any()

reviews_data.shape
```

Out[4]:

```
(50000, 2)
```

For data we are going to use IMDB movie rating Data set

Step 5 - Remove punctuation and special character

In [12]:

```
def Mytext_preprocess(sentnc):

    text1 = tags_remove(sentnc) # Remove html tags


    text1 = re.sub('[^a-zA-Z]', ' ', text1) # Remove punctuations and numbers


    text1 = re.sub(r"\s+[a-zA-Z]\s+", ' ', text1) # Single character removal


    text1 = re.sub(r'\s+', ' ', text1) # Removing multiple spaces

    return text1
```

Here in the above we are removing punctuations and specials characters from our data set, there are html tags, extra spaces are present in our data so we need to remove them for better result.

In [13]:

```
re_tag = re.compile(r'<[^>]+>')

def tags_remove(text2):
    return re_tag.sub('', text2)
```

Step 6 - Clean the text

In [14]:

```
movie_reviews = []
sentences = list(reviews_data['review'])
for data in sentences:
    movie_reviews.append(Mytext_preprocess(data))
```

In [16]:

```
movie_reviews
```

```
an adult movie and wasn much of either Just awful real disappointment to s
ay the least Save your money ',
 'I remember this film it was the first film had watched at the cinema the
picture was dark in places was very nervous it was back in my Dad took me
my brother sister to Newbury cinema in Newbury Berkshire England recall th
e tigers and the lots of snow in the film also the appearance of Grizzly A
dams actor Dan Haggery think one of the tigers gets shot and dies If anyon
e knows where to find this on DVD etc please let me know The cinema now ha
s been turned in fitness club which is very big shame as the nearest cinem
a now is miles away would love to hear from others who have seen this film
or any other like it ',
 'An awful film It must have been up against some real stinkers to be nomi
nated for the Golden Globe They ve taken the story of the first famous fem
ale Renaissance painter and mangled it beyond recognition My complaint is
not that they ve taken liberties with the facts if the story were good tha
t would perfectly fine But it simply bizarre by all accounts the true stor
y of this artist would have made for far better film so why did they come
up with this dishwater dull script suppose there weren enough naked people

in the factual version It hurriedly capped off in the end with summary of
```

Step 7 - Print the Review column values

In [15]:

```
print(reviews_data.columns.values)
```

```
['review' 'sentiment']
```

The movie_reviews here contains two columns review and sentiments. In review column it contains the text data while in sentiment column it contains the sentiments in the form of text.

Step 8 - Unique values of sentiment column

In [17]:

```
reviews_data.sentiment.unique()
```

Out[17]:

```
array(['positive', 'negative'], dtype=object)
```

Step 9 - Convert the sentiment values with integers

In [18]:

```
import numpy as np
y_var = reviews_data['sentiment']

y_var = np.array(list(map(lambda x: 1 if x=="positive" else 0, y_var)))
```

As we all know that algorithms work with integer values, so we need to convert the text data into integer, for that we are using numpy and aslo with the help of lambda function we will convert the 'positive' text as '1' and remaining all as '0'.

Step 10 - Print the reviews

In [21]:

```
print(movie_reviews[10])
```

```
Phil the Alien is one of those quirky films where the humour is based around
the oddness of everything rather than actual punchlines At first it was very
odd and pretty funny but as the movie progressed didn find the jokes or oddn
ess funny anymore Its low budget film thats never problem in itself there we
re some pretty interesting characters but eventually just lost interest imag
ine this film would appeal to stoner who is currently partaking For somethin
g similar but better try Brother from another planet
```

As we can see that the reviews variables consist of only text data in it while the other variable contains the corresponding labels for the same.

In [22]:

```
print(y_var[10])
```

```
0
```

Step 11 - Create BERT tokenizer

In [23]:

```
Tokenizer_Bert = bert.bert_tokenization.FullTokenizer
layer_bert = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/1
file_vocab = layer_bert.resolved_object.vocab_file.asset_path.numpy()
lower_case = layer_bert.resolved_object.do_lower_case.numpy()
tokenized_result = Tokenizer_Bert(file_vocab, lower_case)
```

```
WARNING:tensorflow:Please fix your imports. Module tensorflow.python.trainin
g.tracking.data_structures has been moved to tensorflow.python.trackable.dat
a_structures. The old module will be deleted in version 2.11.
```

In above we create a FullTokenizer class from the bert.bert_tokenization module. Then by importing the BERT model from hub.KerasLayer we create a BERT embedding layer. We will not be training the BERT embedding, as trainable parameter is set to False. After that we create a BERT vocabulary file in the form a numpy array. We then set the text to lowercase and finally we pass our vocabulary file i.e file_vocab and to lower case i.e lower_case variables to the Tokenizer_Bert object.

Step 12 - Check the tokenizer is working or not

In [24]:

```
tokenized_result.tokenize("don't be so judgmental")
```

Out[24]:

```
['don', "'", 't', 'be', 'so', 'judgment', '##al']
```

In [26]:

```
tokenized_result.convert_tokens_to_ids(tokenized_result.tokenize("dont be so judgmental"))
```

Out[26]:

```
[2123, 2102, 2022, 2061, 8689, 2389]
```

Step 13 - Define a function for single text review

In [27]:

```
def reviews_tokenize(reviews_text):
    return tokenized_result.convert_tokens_to_ids(tokenized_result.tokenize(reviews_text))
```

The above function will accepts a single text review and returns the ids of the tokenized words in the review.

Step 14 - Tokenize all the reviews in the input dataset

In [29]:

```
reviews_tokenized = [reviews_tokenize(review) for review in movie_reviews]
```

Step 15 - Prepare Data for training

In [31]:

```
reviews_with_len = [[review, y_var[i], len(review)]
                 for i, review in enumerate(reviews_tokenized)]
```

Here the following script states that it create of lists of list where each sublist contains tokenized review, the label and length of the review.

Step 16 - Shuffle the reviews randomly

In [32]:

```
import random
random.shuffle(reviews_with_len)
```

We need to shuffle the reviews randomly because in our data set there positive and negative both reviews are there, the first half of the reviews are positive while the last half contains negative reviews. Therefore, in order to have both positive and negative reviews in the training batches we need to shuffle the reviews.

Step 17 - Sort the data by the length of reviews

In [33]:

```
reviews_with_len.sort(key=lambda x: x[2])
```

Step 18 - Remove the length attribute from all the reviews

In [34]:

```
sorted_reviews_labels = [(review_lab[0], review_lab[1]) for review_lab in reviews_with_len]
```

Step 19 - Convert the Data set into tensorflow 2.0-compliant input dataset shape.

In [37]:

```
Convert_dataset = tf.data.Dataset.from_generator(lambda: sorted_reviews_labels, output_type
```

Step 20 - Pad our Converted Dataset for each batch

In [38]:

```
BATCH_SIZE = 32
dataset_batched = Convert_dataset.padded_batch(BATCH_SIZE, padded_shapes=((None, ), ()))
```

In [39]:

```
next(iter(dataset_batched)) ## print the first batch
```

Out[39]:

```
(<tf.Tensor: shape=(32, 21), dtype=int32, numpy=
 array([[ 3078,  5436,  3078,  3257,  3532,  7613,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 2054,  5896,  2054,  2466,  2054,  6752,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 3191,  1996,  2338,  5293,  1996,  3185,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 2062, 23873,  3993,  2062, 11259,  2172,  2172,  2062, 14888,
             0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 1045,  2876,  9278,  2023,  2028,  2130,  2006,  7922, 12635,
          2305,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 2023,  3185,  2003,  6659,  2021,  2009,  2038,  2070,  2204,
          3896,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 8235,  1998,  3048,  4616,  2011,  3419,  2457, 27727,  1998,
          2848, 16133,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 1045,  3246,  2023,  2177,  1997,  2143, 11153,  2196,  2128,
         15908,  2015,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 7918, 14674,  7662,  2003,  6581,  2003,  2023,  2143,  2002,
          3084, 17160,  2450,     0,     0,     0,     0,     0,     0,
             0,     0,     0],
        [11861,  1996, 21442,  6895,  3238,  2515,  2210, 22759,  6198,
          1998,  3185,  2087, 12487,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 2023,  2003,  2307,  3185,  2205,  2919,  2009,  2003,  2025,
          2800,  2006,  2188,  2678,     0,     0,     0,     0,     0,
             0,     0,     0],
        [ 2017,  2488,  5454,  2703,  2310, 25032,  8913,  8159,  2130,
          2065,  2017,  2031,  3427,  2009,     0,     0,     0,     0,
             0,     0,     0],
        [ 2053,  7615,  5236,  3185,  3772,  2779,  2030,  4788,  9000,
          2053,  3168,  2012,  2035, 13558,  2009,     0,     0,     0,
             0,     0,     0],
        [ 1045,  2123,  2113,  2339,  2066,  2023,  3185,  2061,  2092,
          2021,  2196,  2131,  5458,  1997,  3666,  2009,     0,     0,
             0,     0,     0],
        [ 7615,  2023,  3185,  2003,  5263,  2003,  6659,  2200, 17727,
          3217,  3676,  3468,  2919,  7613,  3257,  2025,  2298,     0,
             0,     0,     0],
        [ 2146, 11771,  1038,  8523,  8458,  6633,  3560,  2196,  2031,
          2042,  2061,  5580,  2000,  2156,  4566,  6495,  4897,     0,
             0,     0,     0],
        [ 2074,  2293,  1996,  6970, 13068,  2090,  2048,  2307,  3494,
          1997,  2754,  3898,  2310,  3593,  2102,  6287,  5974,     0,
             0,     0,     0],
        [ 2023,  2003,  1996, 15764,  3185,  2544,  1997,  8429, 24905,
         17988,  7659,  2498,  2021,  2045,  2024,  2053, 13842,  5312,
             0,     0,     0],
```

```
        [ 2235,  3077,  2792,  3425,  2003,  1996,  2190,  2792,  1997,
          2235,  3077,  2009,  2026,  5440,  2792,  1997,  2235,  3077,
             0,     0,     0],
        [ 1037,  7244,  3185,  2009,  2003,  2440,  1997,  6699,  1998,
          6919,  3772,  2071,  2031,  2938,  2083,  2009,  2117,  2051,
             0,     0,     0],
        [ 2023,  2003,  2204,  2143,  2023,  2003,  2200,  6057,  2664,
          2044,  2023,  2143,  2045,  2020,  2053,  2204,  8471,  3152,
             0,     0,     0],
        [ 7078, 10392,  3649,  2360,  2876,  2079,  2023,  2104,  9250,
          3185,  1996,  3425,  2009, 17210,  3422,  2009,  2085, 10392,
             0,     0,     0],
        [ 5587,  2023,  2210, 17070,  2000,  2115,  2862,  1997,  6209,
         24945,  2009, 26354, 28394,  2102,  6057,  1998,  2203, 27242,
             0,     0,     0],
        [ 2235,  3077,  2792,  3425,  2003,  1996,  2190,  2792,  1997,
          2235,  3077,  2009,  2026,  5440,  2792,  1997,  2235,  3077,
             0,     0,     0],
        [ 1037,  5790,  1997,  2515,  2025,  4088,  2000,  4671,  2129,
         10634,  2139, 24128,  1998, 21660,  2135,  2919,  2023,  3185,
          2003,     0,     0],
        [ 6283,  2009,  2007,  2035,  2026,  2108,  5409,  3185,  2412,
         10597, 21985,  2393,  2033,  2009,  2001,  2008,  2919,  3404,
          2033,     0,     0],
        [ 1037,  2033,  6491, 11124,  6774,  2143,  2008,  5121,  7906,
          2115,  3086,  3841, 13196,  2003, 17160,  1998, 26103,  2000,
          3422,     0,     0],
        [ 2005,  5760,  7788,  4393,  8808,  2498,  2064, 12826,  2000,
          1996, 11056,  3152,  3811, 16755,  2169,  1998,  2296,  2028,
          1997,  2068,     0],
        [ 7244,  2092,  2856, 10828,  1997, 10904,  2402,  2472,  3135,
          2293,  2466,  2007, 10958,  8428, 10102,  1999,  1996,  4281,
          4276,  3773,     0],
        [ 2028,  1997,  1996,  4569, 15580,  2102,  5691,  2081,  1999,
          3522,  2086,  2204, 23191,  5436,  1998, 11813,  6370,  2191,
          2023,  2028,  4438],
        [ 2023,  2003,  6659,  3185,  2123,  5949,  2115,  2769,  2006,
          2009,  2123,  2130,  3422,  2009,  2005,  2489,  2008,  2035,
          2031,  2000,  2360],
        [ 2023,  3185,  2097,  2467,  2022,  5934,  1998,  3185,  4438,
          2004,  2146,  2004,  2045,  2024,  2145,  2111,  2040,  6170,
          3153,  1998,  2552]])>,
 <tf.Tensor: shape=(32,), dtype=int32, numpy=
 array([0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 0, 1, 1, 1, 1, 0, 1])>)
```

The padding for next batch will be different depending upon the size of the largest sentence in the batch. As the above output shows the first five and last five padded reviews. From the last five reviews, you can see that the total number of words in the largest sentence were 21. Therefore, in the first five reviews the 0s are added at the end of the sentences so that their total length is also 21

Step 21 - Divide the Datas set into train and test

In [40]:

```
import math
TOTAL_BATCHES = math.ceil(len(sorted_reviews_labels) / BATCH_SIZE)
TEST_BATCHES = TOTAL_BATCHES // 10
dataset_batched.shuffle(TOTAL_BATCHES)
test_data = dataset_batched.take(TEST_BATCHES)
train_data = dataset_batched.skip(TEST_BATCHES)
```

Step 22 - Create the model

In [41]:

```
class TEXT_MODEL(tf.keras.Model):

    def __init__(self,
                 vocabulary_size,
                 embedding_dimensions=128,
                 cnn_filters=50,
                 dnn_units=512,
                 model_output_classes=2,
                 dropout_rate=0.1,
                 training=False,
                 name="text_model"):
        super(TEXT_MODEL, self).__init__(name=name)

        self.embedding = layers.Embedding(vocabulary_size,
                                          embedding_dimensions)
        self.cnn_layer1 = layers.Conv1D(filters=cnn_filters,
                                        kernel_size=2,
                                        padding="valid",
                                        activation="relu")
        self.cnn_layer2 = layers.Conv1D(filters=cnn_filters,
                                        kernel_size=3,
                                        padding="valid",
                                        activation="relu")
        self.cnn_layer3 = layers.Conv1D(filters=cnn_filters,
                                        kernel_size=4,
                                        padding="valid",
                                        activation="relu")
        self.pool = layers.GlobalMaxPool1D()

        self.dense_1 = layers.Dense(units=dnn_units, activation="relu")
        self.dropout = layers.Dropout(rate=dropout_rate)
        if model_output_classes == 2:
            self.last_dense = layers.Dense(units=1,
                                           activation="sigmoid")
        else:
            self.last_dense = layers.Dense(units=model_output_classes,
                                           activation="softmax")

    def call(self, inputs, training):
        l = self.embedding(inputs)
        l_1 = self.cnn_layer1(l)
        l_1 = self.pool(l_1)
        l_2 = self.cnn_layer2(l)
        l_2 = self.pool(l_2)
        l_3 = self.cnn_layer3(l)
        l_3 = self.pool(l_3)

        concatenated = tf.concat([l_1, l_2, l_3], axis=-1) # (batch_size, 3 * cnn_filters)
        concatenated = self.dense_1(concatenated)
        concatenated = self.dropout(concatenated, training)
        model_output = self.last_dense(concatenated)

        return model_output
```

Step 23 - Define the values for hyperparameters

In [42]:

```
VOCAB_LENGTH = len(tokenized_result.vocab)
EMB_DIM = 200
CNN_FILTERS = 100
DNN_UNITS = 256
OUTPUT_CLASSES = 2

DROPOUT_RATE = 0.2

NB_EPOCHS = 5
```

Step 24 - Create a Text model and pass hyperparameters values

In [43]:

```
My_text_model = TEXT_MODEL(vocabulary_size=VOCAB_LENGTH,
                           embedding_dimensions=EMB_DIM,
                           cnn_filters=CNN_FILTERS,
                           dnn_units=DNN_UNITS,
                           model_output_classes=OUTPUT_CLASSES,
                           dropout_rate=DROPOUT_RATE)
```

Step 25 - Compile the model

In [44]:

```
if OUTPUT_CLASSES == 2:
    My_text_model.compile(loss="binary_crossentropy",
                          optimizer="adam",
                          metrics=["accuracy"])
else:
    My_text_model.compile(loss="sparse_categorical_crossentropy",
                          optimizer="adam",
                          metrics=["sparse_categorical_accuracy"])
```

Step 26 - Train the model

In [45]:

```
My_text_model.fit(train_data, epochs=NB_EPOCHS)
```

```
Epoch 1/5
1407/1407 [==============================] - 376s 266ms/step - loss: 0.3030
- accuracy: 0.8667
Epoch 2/5
1407/1407 [==============================] - 355s 252ms/step - loss: 0.1369
- accuracy: 0.9498
Epoch 3/5
1407/1407 [==============================] - 336s 239ms/step - loss: 0.0660
- accuracy: 0.9764
Epoch 4/5
1407/1407 [==============================] - 337s 240ms/step - loss: 0.0428
- accuracy: 0.9847
Epoch 5/5
1407/1407 [==============================] - 326s 232ms/step - loss: 0.0225
- accuracy: 0.9924
```

Out[45]:

```
<keras.callbacks.History at 0x22ac6f3c940>
```

Step 27 - Print the results

In [46]:

```
results = My_text_model.evaluate(test_data)
print(results)
```

```
156/156 [==============================] - 2s 10ms/step - loss: 0.4972 - acc
uracy: 0.8970
[0.497222900390625, 0.8970352411270142]
```

from the above results we can see that we got an accuracy of 89%