ARTIFICIAL INTELLIGENCE MINI-PROJECT

MOVIE REVIEW TEXT CLASSIFICATION

RHEA SAWANT

18070123101

E & T C - B

2018-22

In this mini-project, textual movie reviews from IMBD datasets from google & tensorflow are used to train the model and then predict the nature of the review entered by the user.

Text classification is the process of assigning tags or categories to text according to its content. It’s one of the fundamental tasks in natural language processing with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection. A lot of data on the internet is unstructured and hence is said to be just occupying storage space on the cloud. However, if the data is structured some meaningful conclusions can be drawn from it, hence, text classification is one such tool that can help us identify and add value to user inputs, reviews in this case which are otherwise just a set of alphabets.

PART 1

MODEL TRAINING

In [10]:

```
import tensorflow as tf
import tensorflow_datasets as tfds
import tensorflow_hub as hub
import matplotlib.pyplot as plt
import numpy as np
from keras.preprocessing import sequence
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.metrics import accuracy_score
```

All libraries and packages that are required are imported

In [11]:

```
# Check for available GPU
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
    raise SystemError('GPU not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

Code for checking whether GPU is present.

In [12]:

```
(train_x, train_y), (test_x, test_y) = tf.keras.datasets.imdb.load_data(num_words=25000)
```

Code for downloading the IMDB dataset to work on movie review classification with.

In [13]:

```
train_x = sequence.pad_sequences(train_x)
test_x = pad_sequences(test_x)
```

```
test_x = pad_sequences(test_x)
test_x
```

Out[13]:

```
array([[   0,    0,    0, ...,   14,    6,  717],
       [   0,    0,    0, ...,  125,    4, 3077],
       [   0,    0,    0, ...,    9,   57,  975],
       ...,
       [   0,    0,    0, ...,   21,  846, 5518],
       [   0,    0,    0, ..., 2302,    7,  470],
       [   0,    0,    0, ...,   34, 2005, 2643]], dtype=int32)
```

Padding the data and making sure each sentence is of the same length

In [14]:

```
def cnn_model(samples, labels):
  with tf.device(device_name):
    model=tf.keras.Sequential([
        tf.keras.layers.Embedding(25000,128),
        tf.keras.layers.LSTM(32, return_sequences = True),
        tf.keras.layers.GlobalMaxPool1D(),
        tf.keras.layers.Dense(20,activation="relu"),
        tf.keras.layers.Dropout(0.05),
        tf.keras.layers.Dense(1,activation="sigmoid")
    ])
    model.compile(optimizer="adam",
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    history = model.fit(samples, labels, batch_size=100,
             epochs=10)
    print(model.summary())
  return model
```

Initialising the model and adding in layers for filter and selection of parameters.

In [15]:

```
model = cnn_model(train_x, train_y)
```

```
Epoch 1/10
250/250 [==============================] - 38s 151ms/step - loss: 0.5004 - accuracy: 0.7499
Epoch 2/10
250/250 [==============================] - 38s 151ms/step - loss: 0.2179 - accuracy: 0.9193
Epoch 3/10
250/250 [==============================] - 38s 151ms/step - loss: 0.1150 - accuracy: 0.9621
Epoch 4/10
250/250 [==============================] - 38s 151ms/step - loss: 0.0603 - accuracy: 0.9827
Epoch 5/10
250/250 [==============================] - 38s 151ms/step - loss: 0.0343 - accuracy: 0.9906
Epoch 6/10
250/250 [==============================] - 38s 151ms/step - loss: 0.0256 - accuracy: 0.9933
Epoch 7/10
250/250 [==============================] - 38s 150ms/step - loss: 0.0179 - accuracy: 0.9956
Epoch 8/10
250/250 [==============================] - 38s 152ms/step - loss: 0.0173 - accuracy: 0.9950
Epoch 9/10
250/250 [==============================] - 38s 152ms/step - loss: 0.0110 - accuracy: 0.9975
Epoch 10/10
250/250 [==============================] - 38s 152ms/step - loss: 0.0106 - accuracy: 0.9971
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_3 (Embedding)      (None, None, 128)         3200000   
_________________________________________________________________
lstm_3 (LSTM)                (None, None, 32)          20608     
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 32)                0         
_________________________________________________________________
dense_6 (Dense)              (None, 20)                660       
_________________________________________________________________
```

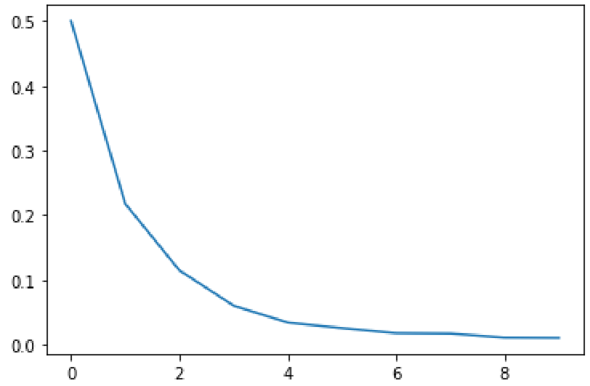
```
dropout_3 (Dropout)          (None, 20)               0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                21
=================================================================
Total params: 3,221,289
Trainable params: 3,221,289
Non-trainable params: 0
_________________________________________________________________
None
```

In [16]:

```
plt.plot(model.history.history['loss'])
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x7f41b6a679b0>]
```



To plot the curve for loss to check how it has progressed.

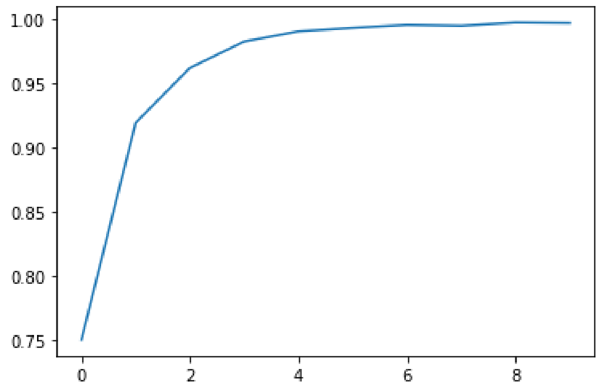
CHECKING ACCURACY

In [17]:

```
plt.plot(model.history.history['accuracy'])
```

Out[17]:

```
[<matplotlib.lines.Line2D at 0x7f41b4163748>]
```



To plot the curve for accuracy to check how it has progressed.

In [18]:

```
pred_y = model.predict(test_x)
```

VALIDATION OF THE MODEL

In [19]:

```
print('Model accuracy: ',accuracy_score(test_y, pred_y.round()))
```

```
Model accuracy:  0.84852
```

After training, the model can predict classes of new data with an accuracy of about 85%

PART 2

PREDICTION USING MODEL

If the predicted value from the array is close to 0, it means that the review entered by the user is a negative review but if the value is close to 1, it means that the review entered by the user is a positive review.

In [20]:

```
(training_data, validation_data), testing_data = tfds.load(
                                                          'imdb_reviews',
                                                          split=['train[:60%]', 'train[60%:100%]'
                                                          with_info=True,
                                                          as_supervised=True)
```

```
Downloading and preparing dataset imdb_reviews/plain_text/1.0.0 (download: 80.23 MiB, generated: U
nknown size, total: 80.23 MiB) to /root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0...
```

```
Shuffling and writing examples to
/root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0.incompleteVX0BS1/imdb_reviews-
train.tfrecord
```

```
Shuffling and writing examples to
/root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0.incompleteVX0BS1/imdb_reviews-
test.tfrecord
```

```
Shuffling and writing examples to
/root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0.incompleteVX0BS1/imdb_reviews-
unsupervised.tfrecord
```

```
WARNING:absl:Dataset is using deprecated text encoder API which will be removed soon. Please use t
he plain_text version of the dataset and migrate to `tensorflow_text`.
```

```
Dataset imdb_reviews downloaded and prepared to
/root/tensorflow_datasets/imdb_reviews/plain_text/1.0.0. Subsequent calls will reuse this data.
```

Importing the dataset from tensorflow datasets and dividing the entire set into training and validation set.

In [21]:

```
training_examples_set,training_labels_set = next(iter(training_data.batch(10)))
training_examples_set
```

Out[21]:

```
<tf.Tensor: shape=(10,), dtype=string, numpy=
array([b"This was an absolutely terrible movie. Don't be lured in by Christopher Walken or Michael
Ironside. Both are great actors, but this must simply be their worst role in history. Even their g
reat acting could not redeem this movie's ridiculous storyline. This movie is an early nineties US
propaganda piece. The most pathetic scenes were those when the Columbian rebels were making their
cases for revolutions. Maria Conchita Alonso appeared phony, and her pseudo-love affair with Walke
n was nothing but a pathetic emotional plug in a movie that was devoid of any real meaning. I am d
isappointed that there are movies like this, ruining actor's like Christopher Walken's good name.
I could barely sit through it.",
       b'I have been known to fall asleep during films, but this is usually due to a combination o
f things including, really tired, being warm and comfortable on the sette and having just eaten a
lot. However on this occasion I fell asleep because the film was rubbish. The plot development was
constant. Constantly slow and boring. Things seemed to happen, but with no explanation of what was
causing them or why. I admit, I may have missed part of the film, but i watched the majority of it
and everything just seemed to happen of its own accord without any real concern for anything else.
I cant recommend this film at all.',
       b'Mann photographs the Alberta Rocky Mountains in a superb fashion, and Jimmy Stewart and W
alter Brennan give enjoyable performances as they always seem to do. <br /><br />But come on
Hollywood - a Mountie telling the people of Dawson City, Yukon to elect themselves a marshal (yes
a marshal!) and to enforce the law themselves, then gunfighters battling it out on the streets for
control of the town? <br /><br />Nothing even remotely resembling that happened on the Canadian si
de of the border during the Klondike gold rush. Mr. Mann and company appear to have mistaken Dawso
n City for Deadwood, the Canadian North for the American Wild West.<br /><br />Canadian viewers be
prepared for a Reefer Madness type of enjoyable howl with this ludicrous plot, or, to shake your h
ead in disgust.',
       b'This is the kind of film for a snowy Sunday afternoon when the rest of the world can go a
head with its own business as you descend into a big arm-chair and mellow for a couple of hours. W
onderful performances from Cher and Nicolas Cage (as always) gently row the plot along. There are
no rapids to cross, no dangerous waters, just a warm and witty paddle through New York life at its
best. A family film in every sense and one that deserves the praise it received.',
       b'As others have mentioned, all the women that go nude in this film are mostly absolutely
gorgeous. The plot very ably shows the hypocrisy of the female libido. When men are around they wa
nt to be pursued, but when no "men" are around, they become the pursuers of a 14 year old boy. And
the boy becomes a man really fast (we should all be so lucky at this age!). He then gets up the co
urage to pursue his true love.',
       b"This is a film which should be seen by anybody interested in, effected by, or suffering
from an eating disorder. It is an amazingly accurate and sensitive portrayal of bulimia in a teena
ge girl, its causes and its symptoms. The girl is played by one of the most brilliant young actres
ses working in cinema today, Alison Lohman, who was later so spectacular in 'Where the Truth Lies'
. I would recommend that this film be shown in all schools, as you will never see a better on this
subject. Alison Lohman is absolutely outstanding, and one marvels at her ability to convey the ang
uish of a girl suffering from this compulsive disorder. If barometers tell us the air pressure, Al
ison Lohman tells us the emotional pressure with the same degree of accuracy. Her emotional range
is so precise, each scene could be measured microscopically for its gradations of trauma, on a sca
le of rising hysteria and desperation which reaches unbearable intensity. Mare Winningham is the p
erfect choice to play her mother, and does so with immense sympathy and a range of emotions just a
s finely tuned as Lohman's. Together, they make a pair of sensitive emotional oscillators
vibrating in resonance with one another. This film is really an astonishing achievement, and direc
tor Katt Shea should be proud of it. The only reason for not seeing it is if you are not intereste
d in people. But even if you like nature films best, this is after all animal behaviour at the sha
rp edge. Bulimia is an extreme version of how a tormented soul can destroy her own body in a frenz
y of despair. And if we don't sympathise with people suffering from the depths of despair, then we
are dead inside.",
       b'Okay, you have:<br /><br />Penelope Keith as Miss Herringbone-Tweed, B.B.E. (Backbone of E
ngland.) She\'s killed off in the first scene - that\'s right, folks; this show has no backbone!<b
r /><br />Peter O\'Toole as Ol\' Colonel Cricket from The First War and now the emblazered Lord of
the Manor.<br /><br />Joanna Lumley as the ensweatered Lady of the Manor, 20 years younger than th
e colonel and 20 years past her own prime but still glamourous (Brit spelling, not mine) enough to
have a toy-boy on the side. It\'s alright, they have Col. Cricket\'s full knowledge and consent (t
hey guy even comes \'round for Christmas!) Still, she\'s considerate of the colonel enough to have
said toy-boy her own age (what a gal!)<br /><br />David McCallum as said toy-boy, equally as point
lessly glamourous as his squeeze. Pilcher couldn\'t come up with any cover for him within the stor
y, so she gave him a hush-hush job at the Circus.<br /><br />and finally:<br /><br />Susan
Hampshire as Miss Polonia Teacups, Venerable Headmistress of the Venerable Girls\' Boarding-School
, serving tea in her office with a dash of deep, poignant advice for life in the outside world jus
t before graduation. Her best bit of advice: "I\'ve only been to Nancherrow (the local Stately Hom
e of England) once. I thought it was very beautiful but, somehow, not part of the real world." Wel
l, we can\'t say they didn\'t warn us.<br /><br />Ah, Susan - time was, your character would have
been running the whole show. They don\'t write \'em like that any more. Our loss, not yours.<br />
<br />So - with a cast and setting like this, you have the re-makings of "Brideshead Revisited," r
ight?<br /><br />Wrong! They took these 1-dimensional supporting roles because they paid so well.
After all, acting is one of the oldest temp-jobs there is (YOU name another!)<br /><br />First war
ning sign: lots and lots of backlighting. They get around it by shooting outdoors - "hey, it\'s ju
st the sunlight!"<br /><br />Second warning sign: Leading Lady cries a lot. When not crying, her e
yes are moist. That\'s the law of romance novels: Leading Lady is "dewy-eyed."<br /><br
/>Henceforth, Leading Lady shall be known as L.L.<br /><br />Third warning sign: L.L. actually has
```

```
stars in her eyes when she\'s in love. Still, I\'ll give Emily Mortimer an award just for having t
o act with that spotlight in her eyes (I wonder . did they use contacts?)<br /><br />And lastly, f
ourth warning sign: no on-screen female character is "Mrs." She\'s either "Miss" or "Lady."<br /><
br />When all was said and done, I still couldn\'t tell you who was pursuing whom and why. I could
n\'t even tell you what was said and done.<br /><br />To sum up: they all live through World War I
I without anything happening to them at all.<br /><br />OK, at the end, L.L. finds she\'s lost her
parents to the Japanese prison camps and baby sis comes home catatonic. Meanwhile (there\'s always
a "meanwhile,") some young guy L.L. had a crush on (when, I don\'t know) comes home from some
wartime tough spot and is found living on the street by Lady of the Manor (must be some street if
SHE\'s going to find him there.) Both war casualties are whisked away to recover at Nancherrow
(SOMEBODY has to be "whisked away" SOMEWHERE in these romance stories!)<br /><br />Great drama.',
       b'The film is based on a genuine 1950s novel.<br /><br />Journalist Colin McInnes wrote a se
t of three "London novels": "Absolute Beginners", "City of Spades" and "Mr Love and Justice". I ha
ve read all three. The first two are excellent. The last, perhaps an experiment that did not come
off. But McInnes\'s work is highly acclaimed; and rightly so. This musical is the novelist\'s ulti
mate nightmare - to see the fruits of one\'s mind being turned into a glitzy, badly-acted, soporif
ic one-dimensional apology of a film that says it captures the spirit of 1950s London, and does no
thing of the sort.<br /><br />Thank goodness Colin McInnes wasn\'t alive to witness it.',
       b'I really love the sexy action and sci-fi films of the sixties and its because of the actr
ess\'s that appeared in them. They found the sexiest women to be in these films and it didn\'t
matter if they could act (Remember "Candy"?). The reason I was disappointed by this film was becau
se it wasn\'t nostalgic enough. The story here has a European sci-fi film called "Dragonfly" being
made and the director is fired. So the producers decide to let a young aspiring filmmaker (Jeremy
Davies) to complete the picture. They\'re is one real beautiful woman in the film who plays Dragon
fly but she\'s barely in it. Film is written and directed by Roman Coppola who uses some of his fa
thers exploits from his early days and puts it into the script. I wish the film could have been an
homage to those early films. They could have lots of cameos by actors who appeared in them. There
is one actor in this film who was popular from the sixties and its John Phillip Law (Barbarella).
Gerard Depardieu, Giancarlo Giannini and Dean Stockwell appear as well. I guess I\'m going to have
to continue waiting for a director to make a good homage to the films of the sixties. If any are r
eading this, "Make it as sexy as you can"! I\'ll be waiting!',
       b'Sure, this one isn\'t really a blockbuster, nor does it target such a position. "Dieter"
is the first name of a quite popular German musician, who is either loved or hated for his kind of
acting and thats exactly what this movie is about. It is based on the autobiography "Dieter
Bohlen" wrote a few years ago but isn\'t meant to be accurate on that. The movie is filled with so
me sexual offensive content (at least for American standard) which is either amusing (not for the
other "actors" of course) or dumb - it depends on your individual kind of humor or on you being a
"Bohlen"-Fan or not. Technically speaking there isn\'t much to criticize. Speaking of me I find th
is movie to be an OK-movie.'],
      dtype=object)>
```

In [22]:

```python
training_labels_set
```

Out[22]:

```
<tf.Tensor: shape=(10,), dtype=int64, numpy=array([0, 0, 0, 1, 1, 1, 0, 0, 0, 0])>
```

In [23]:

```python
pretrained_model = "https://tfhub.dev/google/tf2-preview/gnews-swivel-20dim/1"
hub_layer = hub.KerasLayer(pretrained_model, input_shape=[], dtype=tf.string , trainable=True)
```

Importing the model that does word to vector embedding from tensorflow.

In [24]:

```python
training_examples_set[:3]
```

Out[24]:

```
<tf.Tensor: shape=(3,), dtype=string, numpy=
array([b"This was an absolutely terrible movie. Don't be lured in by Christopher Walken or Michael
Ironside. Both are great actors, but this must simply be their worst role in history. Even their g
reat acting could not redeem this movie's ridiculous storyline. This movie is an early nineties US
propaganda piece. The most pathetic scenes were those when the Columbian rebels were making their
cases for revolutions. Maria Conchita Alonso appeared phony, and her pseudo-love affair with Walke
n was nothing but a pathetic emotional plug in a movie that was devoid of any real meaning. I am d
isappointed that there are movies like this, ruining actor's like Christopher Walken's good name.
I could barely sit through it.",
       b'I have been known to fall asleep during films, but this is usually due to a combination o
f things including, really tired, being warm and comfortable on the sette and having just eaten a
```

```
lot. However on this occasion I fell asleep because the film was rubbish. The plot development was
constant. Constantly slow and boring. Things seemed to happen, but with no explanation of what was
causing them or why. I admit, I may have missed part of the film, but i watched the majority of it
and everything just seemed to happen of its own accord without any real concern for anything else.
I cant recommend this film at all.',
       b'Mann photographs the Alberta Rocky Mountains in a superb fashion, and Jimmy Stewart and W
alter Brennan give enjoyable performances as they always seem to do. <br /><br />But come on
Hollywood - a Mountie telling the people of Dawson City, Yukon to elect themselves a marshal (yes
a marshal!) and to enforce the law themselves, then gunfighters battling it out on the streets for
control of the town? <br /><br />Nothing even remotely resembling that happened on the Canadian si
de of the border during the Klondike gold rush. Mr. Mann and company appear to have mistaken Dawso
n City for Deadwood, the Canadian North for the American Wild West.<br /><br />Canadian viewers be
prepared for a Reefer Madness type of enjoyable howl with this ludicrous plot, or, to shake your h
ead in disgust.'],
      dtype=object)>
```

In [25]:

```
model=tf.keras.Sequential()
model.add(hub_layer)
model.add(tf.keras.layers.Dense(16,activation="relu"))
model.add(tf.keras.layers.Dense(1,activation="sigmoid"))
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
keras_layer (KerasLayer)     (None, 20)                400020    
_________________________________________________________________
dense_8 (Dense)              (None, 16)                336       
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 17        
=================================================================
Total params: 400,373
Trainable params: 400,373
Non-trainable params: 0
_________________________________________________________________
```

Adding neural layers for better performance and accuracy of the model.

In [26]:

```
model.compile(optimizer="adam",
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

In [27]:

```
history = model.fit(training_data.shuffle(10000).batch(2500),
          epochs=100,
          validation_data = validation_data.batch(2500),
          verbose=1)
```

```
Epoch 1/100
6/6 [==============================] - 2s 308ms/step - loss: 2.3154 - accuracy: 0.4965 - val_loss:
1.9051 - val_accuracy: 0.4945
Epoch 2/100
6/6 [==============================] - 2s 291ms/step - loss: 1.6610 - accuracy: 0.4824 - val_loss:
1.3472 - val_accuracy: 0.4583
Epoch 3/100
6/6 [==============================] - 2s 300ms/step - loss: 1.1914 - accuracy: 0.4357 - val_loss:
1.0272 - val_accuracy: 0.4053
Epoch 4/100
6/6 [==============================] - 2s 296ms/step - loss: 0.9681 - accuracy: 0.4116 - val_loss:
0.9253 - val_accuracy: 0.4125
Epoch 5/100
6/6 [==============================] - 2s 293ms/step - loss: 0.9028 - accuracy: 0.4356 - val_loss:
0.8855 - val_accuracy: 0.4518
Epoch 6/100
6/6 [==============================] - 2s 298ms/step - loss: 0.8491 - accuracy: 0.4699 - val_loss:
0.8091 - val_accuracy: 0.4798
```

```
0.8091 - val_accuracy: 0.4790
Epoch 7/100
6/6 [==============================] - 2s 300ms/step - loss: 0.7676 - accuracy: 0.5009 - val_loss: 
0.7334 - val_accuracy: 0.5142
Epoch 8/100
6/6 [==============================] - 2s 296ms/step - loss: 0.7092 - accuracy: 0.5420 - val_loss: 
0.7001 - val_accuracy: 0.5476
Epoch 9/100
6/6 [==============================] - 2s 296ms/step - loss: 0.6838 - accuracy: 0.5721 - val_loss: 
0.6783 - val_accuracy: 0.5801
Epoch 10/100
6/6 [==============================] - 2s 290ms/step - loss: 0.6605 - accuracy: 0.6057 - val_loss: 
0.6577 - val_accuracy: 0.6130
Epoch 11/100
6/6 [==============================] - 2s 292ms/step - loss: 0.6445 - accuracy: 0.6307 - val_loss: 
0.6470 - val_accuracy: 0.6284
Epoch 12/100
6/6 [==============================] - 2s 294ms/step - loss: 0.6338 - accuracy: 0.6439 - val_loss: 
0.6366 - val_accuracy: 0.6394
Epoch 13/100
6/6 [==============================] - 2s 293ms/step - loss: 0.6228 - accuracy: 0.6574 - val_loss: 
0.6279 - val_accuracy: 0.6444
Epoch 14/100
6/6 [==============================] - 2s 293ms/step - loss: 0.6137 - accuracy: 0.6661 - val_loss: 
0.6199 - val_accuracy: 0.6527
Epoch 15/100
6/6 [==============================] - 2s 295ms/step - loss: 0.6047 - accuracy: 0.6763 - val_loss: 
0.6115 - val_accuracy: 0.6638
Epoch 16/100
6/6 [==============================] - 2s 293ms/step - loss: 0.5961 - accuracy: 0.6844 - val_loss: 
0.6039 - val_accuracy: 0.6730
Epoch 17/100
6/6 [==============================] - 2s 297ms/step - loss: 0.5879 - accuracy: 0.6930 - val_loss: 
0.5967 - val_accuracy: 0.6801
Epoch 18/100
6/6 [==============================] - 2s 293ms/step - loss: 0.5797 - accuracy: 0.7021 - val_loss: 
0.5897 - val_accuracy: 0.6871
Epoch 19/100
6/6 [==============================] - 2s 298ms/step - loss: 0.5719 - accuracy: 0.7091 - val_loss: 
0.5829 - val_accuracy: 0.6938
Epoch 20/100
6/6 [==============================] - 2s 300ms/step - loss: 0.5641 - accuracy: 0.7189 - val_loss: 
0.5760 - val_accuracy: 0.7024
Epoch 21/100
6/6 [==============================] - 2s 304ms/step - loss: 0.5564 - accuracy: 0.7265 - val_loss: 
0.5693 - val_accuracy: 0.7091
Epoch 22/100
6/6 [==============================] - 2s 300ms/step - loss: 0.5488 - accuracy: 0.7325 - val_loss: 
0.5627 - val_accuracy: 0.7167
Epoch 23/100
6/6 [==============================] - 2s 301ms/step - loss: 0.5412 - accuracy: 0.7397 - val_loss: 
0.5562 - val_accuracy: 0.7236
Epoch 24/100
6/6 [==============================] - 2s 299ms/step - loss: 0.5336 - accuracy: 0.7472 - val_loss: 
0.5497 - val_accuracy: 0.7301
Epoch 25/100
6/6 [==============================] - 2s 295ms/step - loss: 0.5260 - accuracy: 0.7546 - val_loss: 
0.5432 - val_accuracy: 0.7355
Epoch 26/100
6/6 [==============================] - 2s 291ms/step - loss: 0.5183 - accuracy: 0.7615 - val_loss: 
0.5367 - val_accuracy: 0.7411
Epoch 27/100
6/6 [==============================] - 2s 298ms/step - loss: 0.5106 - accuracy: 0.7691 - val_loss: 
0.5302 - val_accuracy: 0.7475
Epoch 28/100
6/6 [==============================] - 2s 295ms/step - loss: 0.5028 - accuracy: 0.7745 - val_loss: 
0.5237 - val_accuracy: 0.7532
Epoch 29/100
6/6 [==============================] - 2s 292ms/step - loss: 0.4950 - accuracy: 0.7796 - val_loss: 
0.5172 - val_accuracy: 0.7599
Epoch 30/100
6/6 [==============================] - 2s 301ms/step - loss: 0.4872 - accuracy: 0.7850 - val_loss: 
0.5106 - val_accuracy: 0.7655
Epoch 31/100
6/6 [==============================] - 2s 297ms/step - loss: 0.4793 - accuracy: 0.7914 - val_loss: 
0.5041 - val_accuracy: 0.7728
Epoch 32/100
6/6 [==============================] - 2s 297ms/step - loss: 0.4713 - accuracy: 0.7975 - val_loss: 
```

```
0.4976 - val_accuracy: 0.7763
Epoch 33/100
6/6 [==============================] - 2s 298ms/step - loss: 0.4634 - accuracy: 0.8018 - val_loss: 0.4911 - val_accuracy: 0.7790
Epoch 34/100
6/6 [==============================] - 2s 298ms/step - loss: 0.4554 - accuracy: 0.8071 - val_loss: 0.4846 - val_accuracy: 0.7836
Epoch 35/100
6/6 [==============================] - 2s 300ms/step - loss: 0.4475 - accuracy: 0.8137 - val_loss: 0.4783 - val_accuracy: 0.7873
Epoch 36/100
6/6 [==============================] - 2s 301ms/step - loss: 0.4393 - accuracy: 0.8205 - val_loss: 0.4716 - val_accuracy: 0.7923
Epoch 37/100
6/6 [==============================] - 2s 293ms/step - loss: 0.4313 - accuracy: 0.8241 - val_loss: 0.4652 - val_accuracy: 0.7962
Epoch 38/100
6/6 [==============================] - 2s 297ms/step - loss: 0.4232 - accuracy: 0.8286 - val_loss: 0.4589 - val_accuracy: 0.7996
Epoch 39/100
6/6 [==============================] - 2s 291ms/step - loss: 0.4152 - accuracy: 0.8347 - val_loss: 0.4525 - val_accuracy: 0.8042
Epoch 40/100
6/6 [==============================] - 2s 296ms/step - loss: 0.4073 - accuracy: 0.8398 - val_loss: 0.4463 - val_accuracy: 0.8072
Epoch 41/100
6/6 [==============================] - 2s 298ms/step - loss: 0.3994 - accuracy: 0.8439 - val_loss: 0.4403 - val_accuracy: 0.8106
Epoch 42/100
6/6 [==============================] - 2s 302ms/step - loss: 0.3916 - accuracy: 0.8481 - val_loss: 0.4342 - val_accuracy: 0.8141
Epoch 43/100
6/6 [==============================] - 2s 294ms/step - loss: 0.3839 - accuracy: 0.8524 - val_loss: 0.4283 - val_accuracy: 0.8172
Epoch 44/100
6/6 [==============================] - 2s 295ms/step - loss: 0.3762 - accuracy: 0.8563 - val_loss: 0.4225 - val_accuracy: 0.8216
Epoch 45/100
6/6 [==============================] - 2s 294ms/step - loss: 0.3687 - accuracy: 0.8606 - val_loss: 0.4169 - val_accuracy: 0.8251
Epoch 46/100
6/6 [==============================] - 2s 297ms/step - loss: 0.3613 - accuracy: 0.8633 - val_loss: 0.4114 - val_accuracy: 0.8266
Epoch 47/100
6/6 [==============================] - 2s 300ms/step - loss: 0.3540 - accuracy: 0.8669 - val_loss: 0.4060 - val_accuracy: 0.8313
Epoch 48/100
6/6 [==============================] - 2s 296ms/step - loss: 0.3467 - accuracy: 0.8711 - val_loss: 0.4007 - val_accuracy: 0.8348
Epoch 49/100
6/6 [==============================] - 2s 295ms/step - loss: 0.3396 - accuracy: 0.8739 - val_loss: 0.3956 - val_accuracy: 0.8369
Epoch 50/100
6/6 [==============================] - 2s 296ms/step - loss: 0.3326 - accuracy: 0.8773 - val_loss: 0.3907 - val_accuracy: 0.8381
Epoch 51/100
6/6 [==============================] - 2s 296ms/step - loss: 0.3257 - accuracy: 0.8809 - val_loss: 0.3859 - val_accuracy: 0.8393
Epoch 52/100
6/6 [==============================] - 2s 301ms/step - loss: 0.3189 - accuracy: 0.8843 - val_loss: 0.3813 - val_accuracy: 0.8410
Epoch 53/100
6/6 [==============================] - 2s 301ms/step - loss: 0.3122 - accuracy: 0.8871 - val_loss: 0.3769 - val_accuracy: 0.8430
Epoch 54/100
6/6 [==============================] - 2s 304ms/step - loss: 0.3056 - accuracy: 0.8915 - val_loss: 0.3725 - val_accuracy: 0.8457
Epoch 55/100
6/6 [==============================] - 2s 307ms/step - loss: 0.2990 - accuracy: 0.8941 - val_loss: 0.3684 - val_accuracy: 0.8475
Epoch 56/100
6/6 [==============================] - 2s 304ms/step - loss: 0.2925 - accuracy: 0.8974 - val_loss: 0.3643 - val_accuracy: 0.8486
Epoch 57/100
6/6 [==============================] - 2s 301ms/step - loss: 0.2859 - accuracy: 0.9004 - val_loss: 0.3603 - val_accuracy: 0.8507
Epoch 58/100
```

```
Epoch 58/100
6/6 [==============================] - 2s 307ms/step - loss: 0.2792 - accuracy: 0.9037 - val_loss:
0.3567 - val_accuracy: 0.8507
Epoch 59/100
6/6 [==============================] - 2s 314ms/step - loss: 0.2722 - accuracy: 0.9069 - val_loss:
0.3526 - val_accuracy: 0.8527
Epoch 60/100
6/6 [==============================] - 2s 305ms/step - loss: 0.2654 - accuracy: 0.9099 - val_loss:
0.3489 - val_accuracy: 0.8529
Epoch 61/100
6/6 [==============================] - 2s 299ms/step - loss: 0.2588 - accuracy: 0.9119 - val_loss:
0.3454 - val_accuracy: 0.8546
Epoch 62/100
6/6 [==============================] - 2s 300ms/step - loss: 0.2523 - accuracy: 0.9149 - val_loss:
0.3422 - val_accuracy: 0.8561
Epoch 63/100
6/6 [==============================] - 2s 301ms/step - loss: 0.2460 - accuracy: 0.9177 - val_loss:
0.3391 - val_accuracy: 0.8575
Epoch 64/100
6/6 [==============================] - 2s 309ms/step - loss: 0.2399 - accuracy: 0.9199 - val_loss:
0.3362 - val_accuracy: 0.8577
Epoch 65/100
6/6 [==============================] - 2s 290ms/step - loss: 0.2339 - accuracy: 0.9229 - val_loss:
0.3336 - val_accuracy: 0.8601
Epoch 66/100
6/6 [==============================] - 2s 302ms/step - loss: 0.2282 - accuracy: 0.9241 - val_loss:
0.3312 - val_accuracy: 0.8607
Epoch 67/100
6/6 [==============================] - 2s 297ms/step - loss: 0.2227 - accuracy: 0.9260 - val_loss:
0.3290 - val_accuracy: 0.8614
Epoch 68/100
6/6 [==============================] - 2s 294ms/step - loss: 0.2172 - accuracy: 0.9276 - val_loss:
0.3269 - val_accuracy: 0.8615
Epoch 69/100
6/6 [==============================] - 2s 291ms/step - loss: 0.2120 - accuracy: 0.9297 - val_loss:
0.3250 - val_accuracy: 0.8624
Epoch 70/100
6/6 [==============================] - 2s 292ms/step - loss: 0.2070 - accuracy: 0.9313 - val_loss:
0.3232 - val_accuracy: 0.8627
Epoch 71/100
6/6 [==============================] - 2s 297ms/step - loss: 0.2022 - accuracy: 0.9327 - val_loss:
0.3217 - val_accuracy: 0.8630
Epoch 72/100
6/6 [==============================] - 2s 296ms/step - loss: 0.1974 - accuracy: 0.9342 - val_loss:
0.3203 - val_accuracy: 0.8632
Epoch 73/100
6/6 [==============================] - 2s 292ms/step - loss: 0.1928 - accuracy: 0.9359 - val_loss:
0.3190 - val_accuracy: 0.8642
Epoch 74/100
6/6 [==============================] - 2s 292ms/step - loss: 0.1883 - accuracy: 0.9375 - val_loss:
0.3181 - val_accuracy: 0.8646
Epoch 75/100
6/6 [==============================] - 2s 293ms/step - loss: 0.1840 - accuracy: 0.9390 - val_loss:
0.3169 - val_accuracy: 0.8651
Epoch 76/100
6/6 [==============================] - 2s 298ms/step - loss: 0.1797 - accuracy: 0.9409 - val_loss:
0.3160 - val_accuracy: 0.8652
Epoch 77/100
6/6 [==============================] - 2s 296ms/step - loss: 0.1758 - accuracy: 0.9429 - val_loss:
0.3153 - val_accuracy: 0.8662
Epoch 78/100
6/6 [==============================] - 2s 293ms/step - loss: 0.1716 - accuracy: 0.9447 - val_loss:
0.3146 - val_accuracy: 0.8666
Epoch 79/100
6/6 [==============================] - 2s 291ms/step - loss: 0.1677 - accuracy: 0.9469 - val_loss:
0.3143 - val_accuracy: 0.8671
Epoch 80/100
6/6 [==============================] - 2s 291ms/step - loss: 0.1639 - accuracy: 0.9484 - val_loss:
0.3138 - val_accuracy: 0.8670
Epoch 81/100
6/6 [==============================] - 2s 293ms/step - loss: 0.1602 - accuracy: 0.9487 - val_loss:
0.3137 - val_accuracy: 0.8673
Epoch 82/100
6/6 [==============================] - 2s 293ms/step - loss: 0.1566 - accuracy: 0.9511 - val_loss:
0.3132 - val_accuracy: 0.8679
Epoch 83/100
6/6 [==============================] - 2s 296ms/step - loss: 0.1531 - accuracy: 0.9521 - val_loss:
0.3131 - val_accuracy: 0.8678
```

```
0.3131 - val_accuracy: 0.8678
Epoch 84/100
6/6 [==============================] - 2s 296ms/step - loss: 0.1498 - accuracy: 0.9529 - val_loss:
0.3130 - val_accuracy: 0.8677
Epoch 85/100
6/6 [==============================] - 2s 292ms/step - loss: 0.1463 - accuracy: 0.9549 - val_loss:
0.3130 - val_accuracy: 0.8692
Epoch 86/100
6/6 [==============================] - 2s 296ms/step - loss: 0.1431 - accuracy: 0.9559 - val_loss:
0.3134 - val_accuracy: 0.8685
Epoch 87/100
6/6 [==============================] - 2s 297ms/step - loss: 0.1398 - accuracy: 0.9573 - val_loss:
0.3134 - val_accuracy: 0.8697
Epoch 88/100
6/6 [==============================] - 2s 296ms/step - loss: 0.1368 - accuracy: 0.9589 - val_loss:
0.3139 - val_accuracy: 0.8695
Epoch 89/100
6/6 [==============================] - 2s 294ms/step - loss: 0.1336 - accuracy: 0.9610 - val_loss:
0.3141 - val_accuracy: 0.8700
Epoch 90/100
6/6 [==============================] - 2s 288ms/step - loss: 0.1308 - accuracy: 0.9622 - val_loss:
0.3145 - val_accuracy: 0.8698
Epoch 91/100
6/6 [==============================] - 2s 288ms/step - loss: 0.1278 - accuracy: 0.9634 - val_loss:
0.3150 - val_accuracy: 0.8705
Epoch 92/100
6/6 [==============================] - 2s 286ms/step - loss: 0.1248 - accuracy: 0.9651 - val_loss:
0.3155 - val_accuracy: 0.8706
Epoch 93/100
6/6 [==============================] - 2s 288ms/step - loss: 0.1221 - accuracy: 0.9665 - val_loss:
0.3162 - val_accuracy: 0.8715
Epoch 94/100
6/6 [==============================] - 2s 287ms/step - loss: 0.1194 - accuracy: 0.9672 - val_loss:
0.3168 - val_accuracy: 0.8720
Epoch 95/100
6/6 [==============================] - 2s 292ms/step - loss: 0.1168 - accuracy: 0.9681 - val_loss:
0.3176 - val_accuracy: 0.8719
Epoch 96/100
6/6 [==============================] - 2s 288ms/step - loss: 0.1141 - accuracy: 0.9687 - val_loss:
0.3182 - val_accuracy: 0.8721
Epoch 97/100
6/6 [==============================] - 2s 289ms/step - loss: 0.1116 - accuracy: 0.9698 - val_loss:
0.3191 - val_accuracy: 0.8719
Epoch 98/100
6/6 [==============================] - 2s 287ms/step - loss: 0.1090 - accuracy: 0.9704 - val_loss:
0.3202 - val_accuracy: 0.8711
Epoch 99/100
6/6 [==============================] - 2s 290ms/step - loss: 0.1066 - accuracy: 0.9715 - val_loss:
0.3209 - val_accuracy: 0.8711
Epoch 100/100
6/6 [==============================] - 2s 283ms/step - loss: 0.1042 - accuracy: 0.9727 - val_loss:
0.3220 - val_accuracy: 0.8711
```

The model is now ready to predict the type of review by taking input from the user.

In [28]:

```
model.predict(["worst movie ever!"])
```

Out[28]:

```
array([[0.04661591]], dtype=float32)
```

Since, the predicted value is close to 0 it is considered to be a negative review.

In [29]:

```
model.predict(["best movie ever!"])
```

Out[29]:

```
array([[0.89394546]], dtype=float32)
```

Since, the predicted value is close to 1 it is considered to be a positive review.

Conclusion

The accuracy of the model is enough to predict the correct nature of the review that is entered by the user. Hence, this model is successfully able to classify text after learning from the IMDB dataset.