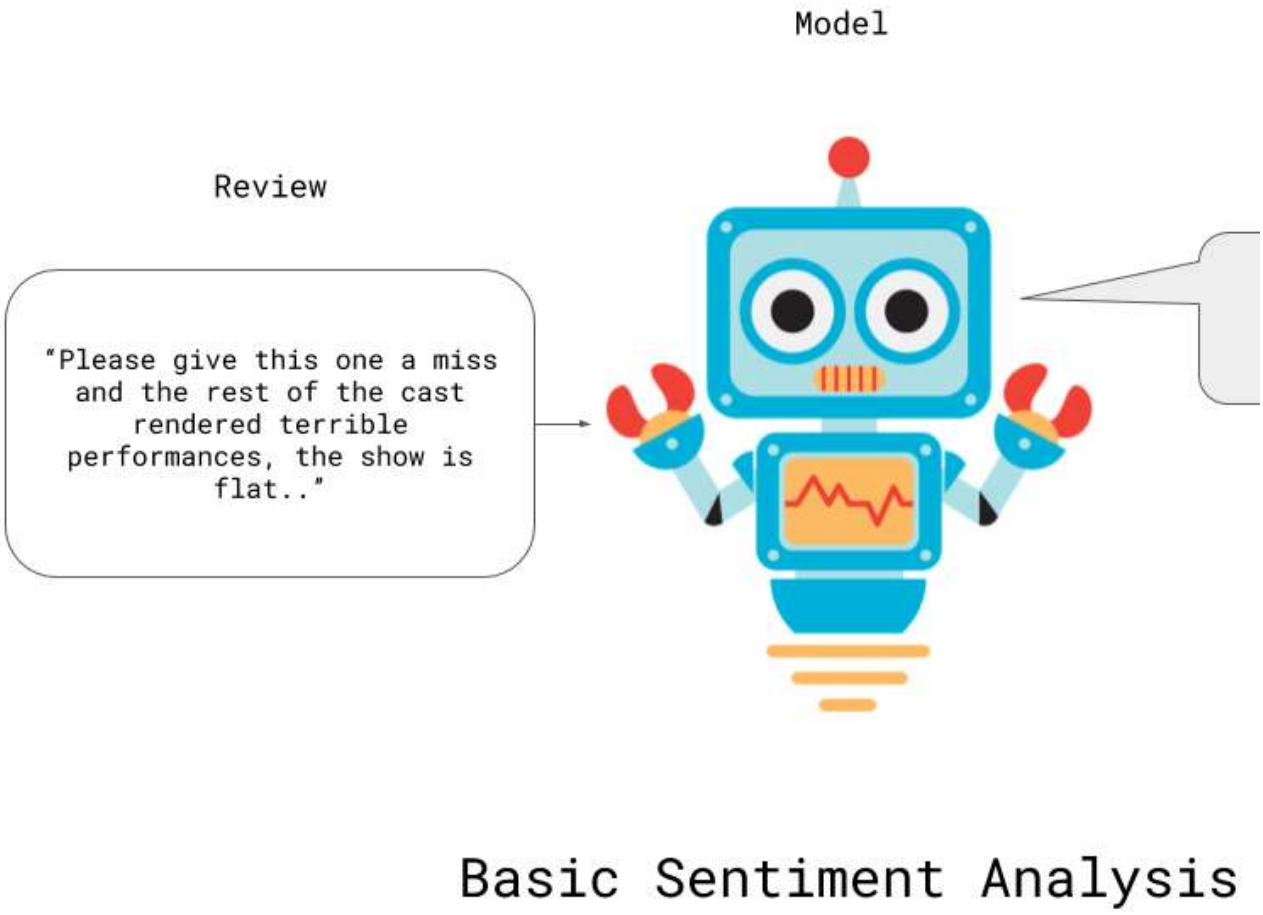
# Task 1: Introduction

Welcome to **Sentiment Analysis with Keras and TensorFlow**.




# Task 2: The IMDB Reviews Dataset

Note: If you are starting the notebook from this task, you can run cells from all previ
going to the top menu and then selecting **Kernel > Restart and Run All**

In [2]:
```
from tensorflow.python.keras.datasets import imdb
(X_train, y_train), (X_test, y_test)= imdb.load_data(num_words= 10000)
#only 10000 words from bag of words will be used and rest will be ignored
```

In [3]:
```
print(X_train[0])#1st review encoded in a way where each word is represented by c
#bag of words->all unique words in all reviews ->assign a numeric token which rep
```

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838
5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6,
13, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2,
4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 3
4, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 15, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400,
8, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 226, 22, 21, 134, 476, 26, 480, 5,
92, 25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 17
```

In [4]:
```
print(y_train[0])#0->negative, 1->positive
```

```
1
```

In [5]:
```
class_names=['Negative' , 'Positive']
```

In [6]:
```
word_index= imdb.get_word_index()
print(word_index['hello'])
```

```
4822
```

# Task 3: Decoding the Reviews

Note: If you are starting the notebook from this task, you can run cells from all previ
going to the top menu and then selecting **Kernel > Restart and Run All**

In [7]:
```
#just for reference
reverse_word_index= dict((value, key) for key, value in word_index.items())

def decode(review):
    text= ''
    for i in review:
        text += reverse_word_index[i]
        text += ' '
    return text
```

In [8]:
```
decode(X_train[0])
```

```
"the as you with out themselves powerful lets loves their becomes reaching had journalist of lot from a
never more room and it so heart shows to years of every never going and help moments or of every chest
of enough more with is now current film as you of mine potentially unfortunately of you than him that w
camp of you movie sometimes movie that with scary but and to story wonderful that in seeing in characte
d shadows they of here that with her serious to have does when from why what have critics they is you t
f with other and in of seen over landed for anyone of and br show's to whether from than out themselves
d odd was two most of mean for 1 any an boat she he should is thought frog but of script you not while
but when from one bit then have two of script their with her nobody most that with wasn't to with armed
film want an "
```

In [10]:
```
def show_len():
    print('Length of 1st training example: ', len(X_train[0]))
    print('Length of 1st training example: ', len(X_train[1]))
    print('Length of 1st test example: ', len(X_test[0]))
    print('Length of 1st test example: ', len(X_test[1]))

show_len()
```

```
Length of 1st training example:  218
Length of 1st training example:  189
Length of 1st test example:  68
Length of 1st test example:  260
```

# Task 4: Padding the Examples

Note: If you are starting the notebook from this task, you can run cells from all previ
going to the top menu and then selecting **Kernel > Restart and Run All**

In [11]:
```
word_index['the']
```

```
1
```

In [12]:
```
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
X_train= pad_sequences(X_train, value= word_index['the'], padding= 'post',
                       maxlen=256)
X_test= pad_sequences(X_test, value= word_index['the'], padding= 'post',
                       maxlen=256)
```

In [13]:
```
show_len()
```

```
Length of 1st training example:  256
Length of 1st training example:  256
Length of 1st test example:  256
Length of 1st test example:  256
```

In [14]:
```
decode(X_train[0])
```

```
"the as you with out themselves powerful lets loves their becomes reaching had journalist of lot from a
never more room and it so heart shows to years of every never going and help moments or of every chest
of enough more with is now current film as you of mine potentially unfortunately of you than him that w
camp of you movie sometimes movie that with scary but and to story wonderful that in seeing in characte
d shadows they of here that with her serious to have does when from why what have critics they is you t
f with other and in of seen over landed for anyone of and br show's to whether from than out themselves
d odd was two most of mean for 1 any an boat she he should is thought frog but of script you not while
but when from one bit then have two of script their with her nobody most that with wasn't to with armed
film want an the the the the the the the the the the the the the the the the the the the the the the th
he the the the the the the "
```

# Task 5: Word Embeddings

Note: If you are starting the notebook from this task, you can run cells from all previ
going to the top menu and then selecting **Kernel > Restart and Run All**

Word Embeddings:

## One Hot Encoding

If the algorithm learns:

> This tuna sandwich is quite tasty.

It can not translate the learning to:

> This chicken _____ is quite t

---

## Word Embeddings

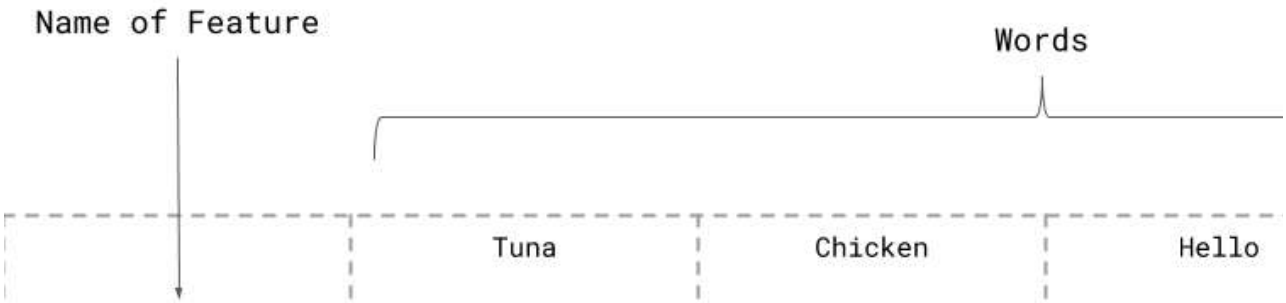
If the algorithm learns:

> This tuna sandwich is quite tasty.

It CAN extrapolate the learning to:

> This chicken sandwich is quite t

Feature Vectors:




| | Tuna | Chicken | Hello |
|---|---|---|---|
| Greeting | 0.01 | -0.04 | 0.99 |
| Gender | 0.09 | -0.72 | 0.02 |
| Age | 0.01 | -0.08 | 0.00 |
| Meat | 0.99 | 0.94 | 0.00 |
| Food | 0.92 | 1.00 | 0.11 |
| Animal | 0.39 | 0.88 | -0.08 |
| Number | 0.00 | -0.01 | -0.08 |

# Task 6: Creating and Training the Model

Note: If you are starting the notebook from this task, you can run cells from all previ
going to the top menu and then selecting **Kernel > Restart and Run All**

In one hot encoding algorithm does not understand feature correlation at all. With w
algorithm can understand related words. Here each word can have a fbunch of diffe
word's feature representation.->f.r meanings cannot be understood gemerally by hu
above. Rows represent embeddings.Embeddings are feature representations for va
representations are learned as we train our model and they become more accurate

In [16]:

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Embedding, GlobalAveragePooling
#embedding- vocab size, size of feature vector
#GAP1D- converts feature representation of (10000,16) to a 16 dimensional vector
#batch and then it can be fed to the Dense layer
#Sigmoid function gives binary classification output
#adam->variant of the stochastic gradient descent, training metric
model= Sequential([Embedding(10000, 16),
                   GlobalAveragePooling1D(),
                   Dense(16, activation='relu'),
                   Dense(1, activation='sigmoid')])
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics= ['accuracy'])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 16)          160000
_________________________________________________________________
global_average_pooling1d (Gl (None, 16)                0
_________________________________________________________________
dense (Dense)                (None, 16)                272
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 160,289
Trainable params: 160,289
Non-trainable params: 0
_________________________________________________________________
```

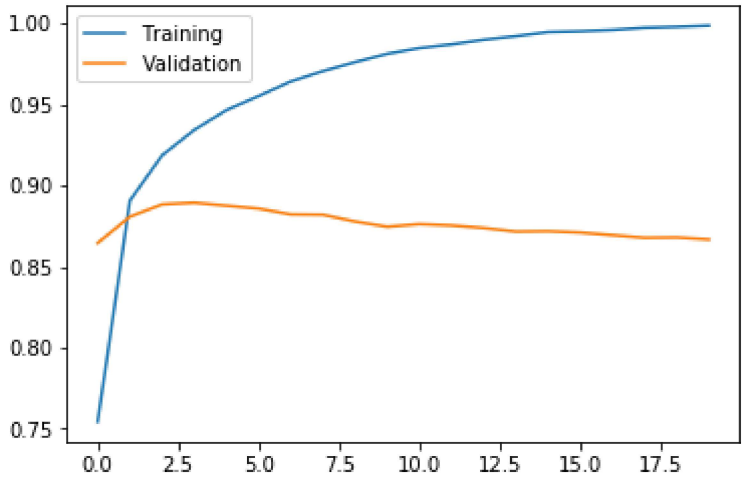
In [17]:
```
from tensorflow.python.keras.callbacks import LambdaCallback
simple_log= LambdaCallback(on_epoch_end= lambda e,l: print(e, end='.'))#not compl
#displayed just epoch no. and '.'. Default logging output is not wanted. epoch,lc
E=20

h= model.fit(X_train, y_train, validation_split=0.2, epochs=E, callbacks= [simple
            verbose=False)
```

```
c:\users\administrator\appdata\local\programs\python\python36\lib\site-packages\tensorflow\python\ops\g
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may consume a large amount of
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "


0.1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.
```

# Task 7: Predictions and Evaluation

Note: If you are starting the notebook from this task, you can run cells from all previ
going to the top menu and then selecting **Kernel > Restart and Run All**

In [18]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(range(E), h.history['acc'], label='Training')
plt.plot(range(E), h.history['val_acc'], label='Validation')
plt.legend()
plt.show()
```



In [19]:
```
loss, acc= model.evaluate(X_test, y_test)
print('Test Set Accuracy: ',acc*100)
```

```
25000/25000 [==============================] - 1s 47us/step
Test Set Accuracy:  84.26
```

In [20]:
```
import numpy as np

p= model.predict(np.expand_dims(X_test[0], axis=0))#expand dim if passing 1 examp
print(class_names[np.argmax(p[0])])
```

```
Negative
```

In [21]:
```
decode(X_test[0])
```

```
"the wonder own as by is sequence i i and and to of hollywood br of down shouting getting boring of eve
does don't close faint after one carry as by are be favourites all family turn in does as three part in
orld and her an have faint beginning own as is sequence the the the the the the the the the the the the
e the the the the the the the the the the the the the the the the the the the the the the the the the t
the the the the the the the the the the the the the the the the the the the the the the the the the the
e the the the the the the the the the the the the the the the the the the the the the the the the the t
the the the the the the the the the the the the the the the the the the the the the the the the the the
e the the the the the the the the the the the the the the the the the the the the the the the the the t
```