NAME:R.RAHINI ROLLNO:215229128

LAB-14:TEXT DATASET CREATION AND DESIGN OF SIMPLE RNN FOR SENTIMENT ANALYSIS

In [1]:
```
import csv
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words ('english'))
import pandas as pd
```

In [2]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\2mscdsa28\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[2]:
```
True
```

In [3]:
```
df=pd.read_csv('movie.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	LABEL	TEXT
0	1	films adapted from comic books have had plenty...
1	1	to say moore and campbell thoroughly researche...
2	1	in other words , don't dismiss this film becau...
3	1	if you can get past the whole comic book thing...
4	1	the ghetto in question is , of course , whitec...

In [5]:
```
df.columns
```

Out[5]:
```
Index(['LABEL', 'TEXT'], dtype='object')
```

In [6]:
```
df.size
```

Out[6]:
```
40
```

pre-processing the Text

In [7]:
```
y = df['LABEL']
x=[]
for review in df['TEXT']:
    filtered_sentence=[w.lower() for w in review.split() if not w in STOPWORDS]
    x.append(filtered_sentence)
x = pd.Series(x)
```

5.Dataset prepartaion

In [8]:
```
from sklearn.model_selection import train_test_split
X_train,X_val,y_train,y_val=train_test_split(x,y,train_size=0.7)
```

In [9]:
```
print(X_train.shape)
print(X_val.shape)
print(y_train.shape)
print(y_val.shape)
```

```
(14,)
(6,)
(14,)
(6,)
```

In [10]:
```
train_token = Tokenizer(num_words = 50,oov_token='<oov>')
train_token.fit_on_texts(X_train)
word_index = train_token.word_index
train_sequences=train_token.texts_to_sequences(X_train)
dict(list(word_index.items())[0:10])
```

Out[10]:
```
{'<oov>': 1,
 '.': 2,
 ',': 3,
 'nightclub': 4,
 'comic': 5,
 'story': 6,
 'getting': 7,
 'trouble': 8,
 'abberline': 9,
 'widower': 10}
```

In [11]:
```
vocab_ze=len(train_token.word_index)+1
vocab_ze
```

Out[11]:
```
97
```

In [12]:
```
train_sequences[4]
```

Out[12]:
```
[30, 31, 32, 3, 2, 2, 2, 33, 2]
```

In [13]:
```
train_padded = pad_sequences(train_sequences,maxlen=100,padding='post')
```

In [14]:
```
train_padded[8]
```

Out[14]:
```
array([1, 1, 1, 5, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 2, 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])
```

In [15]:
```
train_padded.shape
```

Out[15]:
```
(14, 100)
```

In [16]:
```
val_token = Tokenizer(num_words = 50,oov_token='<oov>')
val_token.fit_on_texts(X_val)
val_index = val_token.word_index
val_sequences=train_token.texts_to_sequences(X_val)
```

In [17]:
```
val_sequences[4]
```

Out[17]:
```
[1, 1, 1, 3, 1, 1, 1, 2]
```

In [18]:
```
val_padded=pad_sequences(val_sequences,maxlen=100,padding='post')
```

In [19]:
```
val_padded[2]
```

Out[19]:
```
array([ 1,  1,  3, 44,  2,  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])
```

In [20]:
```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding,SimpleRNN
```

6.Model creation

In [21]:
```
model = Sequential()
# Embedding layer
model.add(Embedding(300,70,input_length=100))
model.add(SimpleRNN(70,activation='relu'))
model.add(Dense('1',activation='sigmoid'))
```

In [22]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [23]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 70)           21000

 simple_rnn (SimpleRNN)      (None, 70)                9870

 dense (Dense)               (None, 1)                 71

=================================================================
Total params: 30,941
Trainable params: 30,941
Non-trainable params: 0
_________________________________________________________________
```

In [24]:
```
history=model.fit(train_padded,y_train,epochs=10,verbose=2,batch_size=15)
```

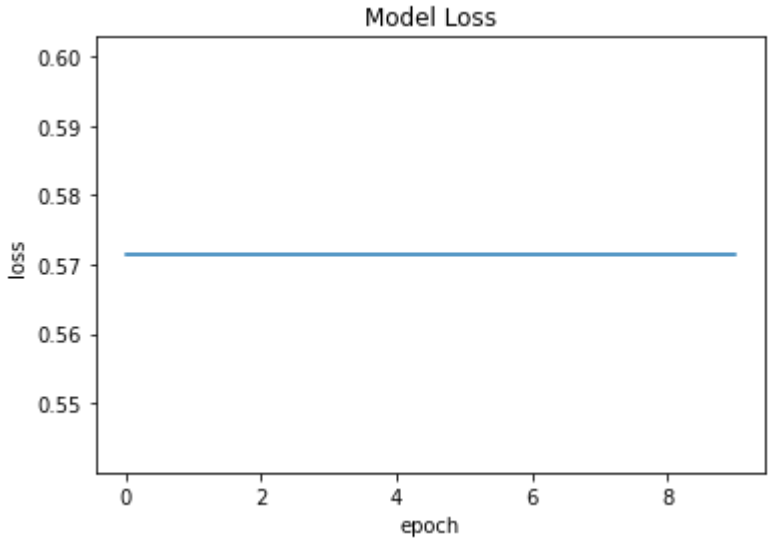
```
Epoch 1/10
1/1 - 1s - loss: 0.6909 - accuracy: 0.5714 - 746ms/epoch - 746ms/step
Epoch 2/10
1/1 - 0s - loss: 0.6888 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
Epoch 3/10
1/1 - 0s - loss: 0.6872 - accuracy: 0.5714 - 12ms/epoch - 12ms/step
Epoch 4/10
1/1 - 0s - loss: 0.6858 - accuracy: 0.5714 - 8ms/epoch - 8ms/step
Epoch 5/10
1/1 - 0s - loss: 0.6847 - accuracy: 0.5714 - 8ms/epoch - 8ms/step
Epoch 6/10
1/1 - 0s - loss: 0.6837 - accuracy: 0.5714 - 8ms/epoch - 8ms/step
Epoch 7/10
1/1 - 0s - loss: 0.6831 - accuracy: 0.5714 - 8ms/epoch - 8ms/step
Epoch 8/10
1/1 - 0s - loss: 0.6829 - accuracy: 0.5714 - 8ms/epoch - 8ms/step
Epoch 9/10
1/1 - 0s - loss: 0.6833 - accuracy: 0.5714 - 8ms/epoch - 8ms/step
Epoch 10/10
1/1 - 0s - loss: 0.6836 - accuracy: 0.5714 - 8ms/epoch - 8ms/step
```

In [25]:
```
model.summary()
```

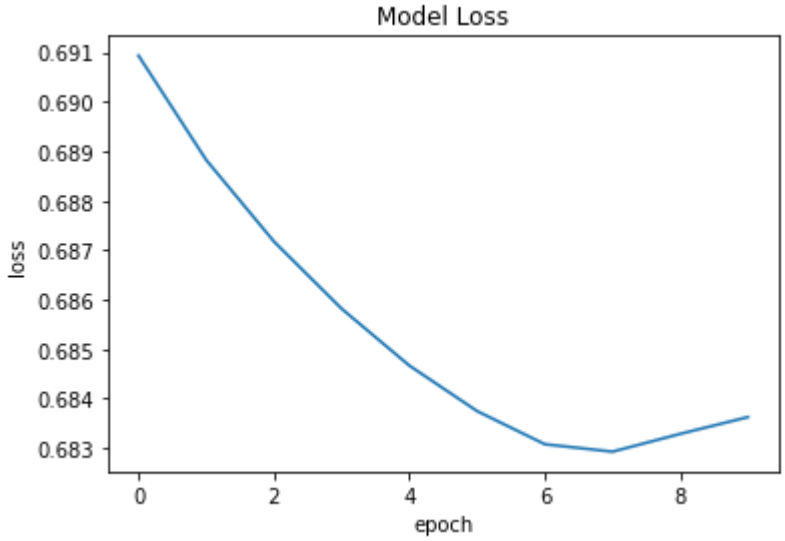
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 70)           21000

 simple_rnn (SimpleRNN)      (None, 70)                9870

 dense (Dense)               (None, 1)                 71

=================================================================
Total params: 30,941
Trainable params: 30,941
Non-trainable params: 0
_________________________________________________________________
```

In [26]:
```
import matplotlib.pyplot as plt
```

In [27]:
```
plt.plot(history.history['accuracy'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```



In [28]:
```
plt.plot(history.history['loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```



In [29]:
```
text = ['Education is what remains after one has forgotten what one has learned i
#sent = [w.lower() for w in text.split() if not w in STOPWORDS]
trail_token = Tokenizer()
trail_token.fit_on_texts(text)
#word_index = trail_token.word_index
trail_seq = trail_token.texts_to_sequences(text)
#dict(list(word_index.items())[0:10])
trail_pad = pad_sequences(trail_seq,maxlen=100,padding='post')
```

In [30]:
```
trail_pad
```

Out[30]:
```
array([[ 4,  5,  1,  6,  7,  2,  3,  8,  1,  2,  3,  9, 10, 11,  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]])
```

In [31]:
```
res = model.predict(trail_pad)
label = ['positive','negative']
print(res,label[np.argmax(trail_pad)>50])
```

```
1/1 [==============================] - 0s 109ms/step
[[0.59026337]] positive


C:\Users\2mscdsa28\AppData\Local\Temp\ipykernel_12452\1535703611.py:3: Deprecat
ionWarning: In future, it will be an error for 'np.bool_' scalars to be interpr
eted as an index
  print(res,label[np.argmax(trail_pad)>50])
```

In [32]:
```
model1 = Sequential()
# Embedding layer
model1.add(Embedding(5000,64,input_length=100))
model1.add(SimpleRNN(32,activation='tanh'))
model1.add(Embedding(5000,32,input_length=100))
model1.add(SimpleRNN(32,activation='tanh' ))
model1.add(Dense('1',activation='sigmoid'))
```

In [33]:
```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 100, 64)           320000

 simple_rnn_1 (SimpleRNN)    (None, 32)                3104

 embedding_2 (Embedding)     (None, 32, 32)            160000

 simple_rnn_2 (SimpleRNN)    (None, 32)                2080

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 485,217
Trainable params: 485,217
Non-trainable params: 0
_________________________________________________________________
```

In [34]:
```
model1.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [35]:
```
history1=model1.fit(train_padded,y_train,epochs=10,verbose=2,batch_size=15)
```

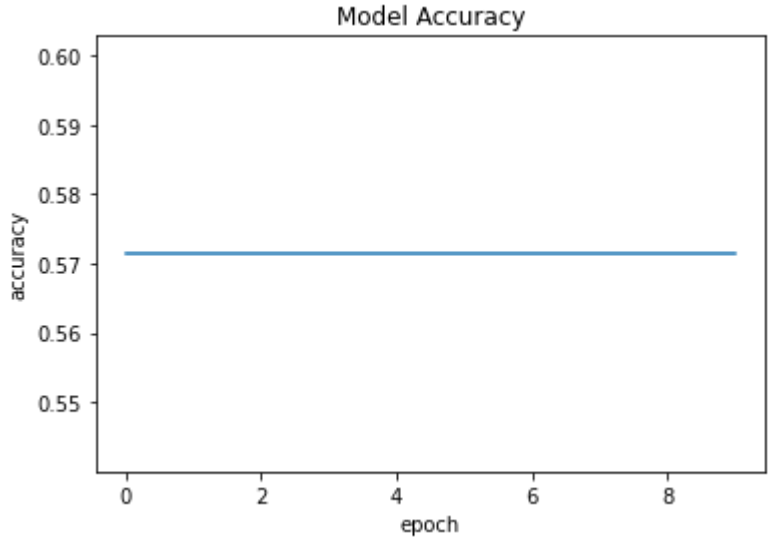
```
Epoch 1/10
WARNING:tensorflow:Gradients do not exist for variables ['embedding_1/embedding
s:0', 'simple_rnn_1/simple_rnn_cell_1/kernel:0', 'simple_rnn_1/simple_rnn_cell_
1/recurrent_kernel:0', 'simple_rnn_1/simple_rnn_cell_1/bias:0'] when minimizing
the loss. If you're using `model.compile()`, did you forget to provide a `loss`
argument?
WARNING:tensorflow:Gradients do not exist for variables ['embedding_1/embedding
s:0', 'simple_rnn_1/simple_rnn_cell_1/kernel:0', 'simple_rnn_1/simple_rnn_cell_
1/recurrent_kernel:0', 'simple_rnn_1/simple_rnn_cell_1/bias:0'] when minimizing
the loss. If you're using `model.compile()`, did you forget to provide a `loss`
argument?
1/1 - 1s - loss: 0.6840 - accuracy: 0.5714 - 754ms/epoch - 754ms/step
Epoch 2/10
1/1 - 0s - loss: 0.6853 - accuracy: 0.5714 - 6ms/epoch - 6ms/step
Epoch 3/10
1/1 - 0s - loss: 0.6839 - accuracy: 0.5714 - 6ms/epoch - 6ms/step
Epoch 4/10
1/1 - 0s - loss: 0.6829 - accuracy: 0.5714 - 4ms/epoch - 4ms/step
Epoch 5/10
1/1 - 0s - loss: 0.6832 - accuracy: 0.5714 - 5ms/epoch - 5ms/step
Epoch 6/10
1/1 - 0s - loss: 0.6835 - accuracy: 0.5714 - 6ms/epoch - 6ms/step
Epoch 7/10
1/1 - 0s - loss: 0.6833 - accuracy: 0.5714 - 5ms/epoch - 5ms/step
Epoch 8/10
1/1 - 0s - loss: 0.6830 - accuracy: 0.5714 - 5ms/epoch - 5ms/step
Epoch 9/10
1/1 - 0s - loss: 0.6830 - accuracy: 0.5714 - 5ms/epoch - 5ms/step
Epoch 10/10
1/1 - 0s - loss: 0.6832 - accuracy: 0.5714 - 4ms/epoch - 4ms/step
```

In [36]:
```
model1.evaluate(val_padded,y_val)
```

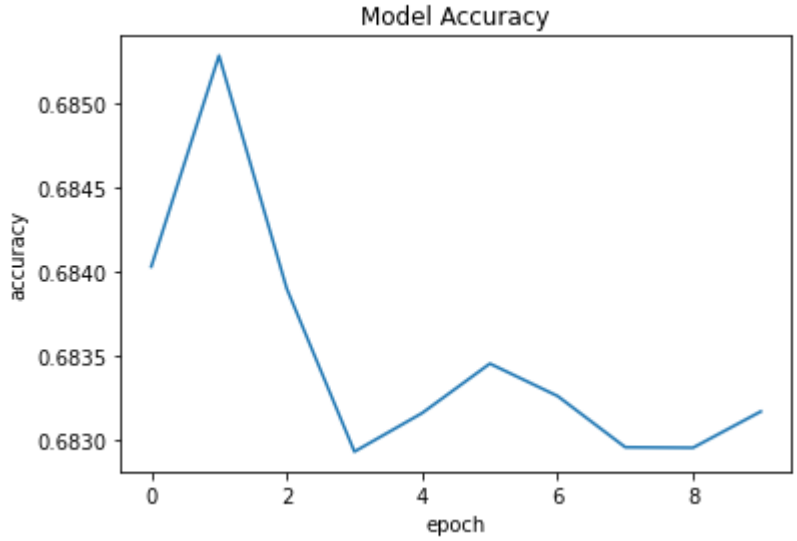
```
1/1 [==============================] - 0s 293ms/step - loss: 0.7401 - accuracy:
0.3333
```

Out[36]:
```
[0.7401018142700195, 0.3333333432674408]
```

In [37]:
```
plt.plot(history1.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [38]:
```
plt.plot(history1.history['loss'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [39]:
```
res = model1.predict(trail_pad)
label = ['positive','negative']
print(res,label[np.argmax(trail_pad)>50])
```

```
1/1 [==============================] - 0s 156ms/step
[[0.55946517]] positive

C:\Users\2mscdsa28\AppData\Local\Temp\ipykernel_12452\2701450631.py:3: Deprecat
ionWarning: In future, it will be an error for 'np.bool_' scalars to be interpr
eted as an index
  print(res,label[np.argmax(trail_pad)>50])
```

In [40]:
```
model2 = Sequential()
# Embedding Layer
model2.add(Embedding(4000,128,input_length=100))
model2.add(SimpleRNN(64,activation='tanh'))
model2.add(Embedding(4000,128,input_length=100))
model2.add(SimpleRNN(64,activation='relu' ))
model2.add(Embedding(4000,128,input_length=100))
model2.add(SimpleRNN(64,activation='tanh' ))
model2.add(Dense('1',activation='sigmoid'))
```

In [41]:
```
model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 100, 128)          512000

 simple_rnn_3 (SimpleRNN)    (None, 64)                12352

 embedding_4 (Embedding)     (None, 64, 128)           512000

 simple_rnn_4 (SimpleRNN)    (None, 64)                12352

 embedding_5 (Embedding)     (None, 64, 128)           512000

 simple_rnn_5 (SimpleRNN)    (None, 64)                12352

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 1,573,121
Trainable params: 1,573,121
Non-trainable params: 0
_________________________________________________________________
```

In [42]:
```
model2.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [43]:
```
history2=model2.fit(train_padded,y_train,epochs=10,verbose=2,batch_size=15)
```

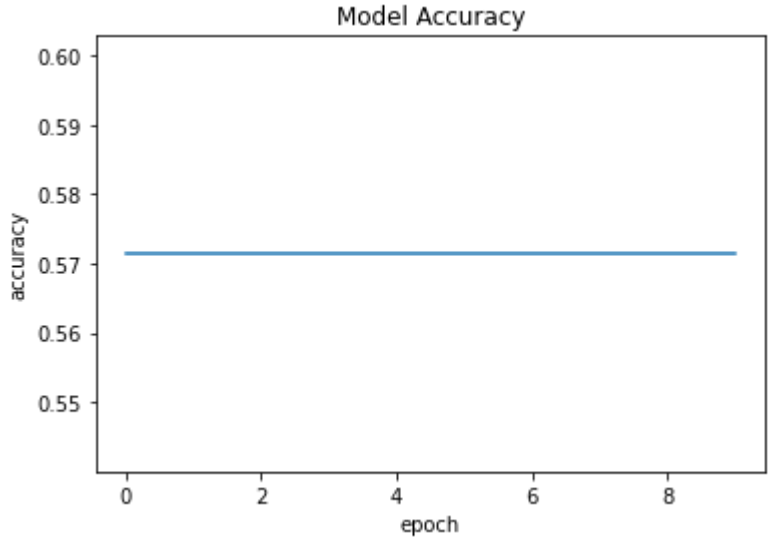
```
Epoch 1/10
WARNING:tensorflow:Gradients do not exist for variables ['embedding_3/embedding
s:0', 'simple_rnn_3/simple_rnn_cell_3/kernel:0', 'simple_rnn_3/simple_rnn_cell_
3/recurrent_kernel:0', 'simple_rnn_3/simple_rnn_cell_3/bias:0', 'embedding_4/em
beddings:0', 'simple_rnn_4/simple_rnn_cell_4/kernel:0', 'simple_rnn_4/simple_rn
n_cell_4/recurrent_kernel:0', 'simple_rnn_4/simple_rnn_cell_4/bias:0'] when min
imizing the loss. If you're using `model.compile()`, did you forget to provide
a `loss` argument?
WARNING:tensorflow:Gradients do not exist for variables ['embedding_3/embedding
s:0', 'simple_rnn_3/simple_rnn_cell_3/kernel:0', 'simple_rnn_3/simple_rnn_cell_
3/recurrent_kernel:0', 'simple_rnn_3/simple_rnn_cell_3/bias:0', 'embedding_4/em
beddings:0', 'simple_rnn_4/simple_rnn_cell_4/kernel:0', 'simple_rnn_4/simple_rn
n_cell_4/recurrent_kernel:0', 'simple_rnn_4/simple_rnn_cell_4/bias:0'] when min
imizing the loss. If you're using `model.compile()`, did you forget to provide
a `loss` argument?
1/1 - 1s - loss: 0.6854 - accuracy: 0.5714 - 834ms/epoch - 834ms/step
Epoch 2/10
1/1 - 0s - loss: 0.6966 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
Epoch 3/10
1/1 - 0s - loss: 0.6887 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
Epoch 4/10
1/1 - 0s - loss: 0.6829 - accuracy: 0.5714 - 9ms/epoch - 9ms/step
Epoch 5/10
1/1 - 0s - loss: 0.6864 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
Epoch 6/10
1/1 - 0s - loss: 0.6861 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
Epoch 7/10
1/1 - 0s - loss: 0.6842 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
Epoch 8/10
1/1 - 0s - loss: 0.6833 - accuracy: 0.5714 - 9ms/epoch - 9ms/step
Epoch 9/10
1/1 - 0s - loss: 0.6830 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
Epoch 10/10
1/1 - 0s - loss: 0.6829 - accuracy: 0.5714 - 10ms/epoch - 10ms/step
```

In [44]:
```
model2.evaluate(val_padded,y_val)
```

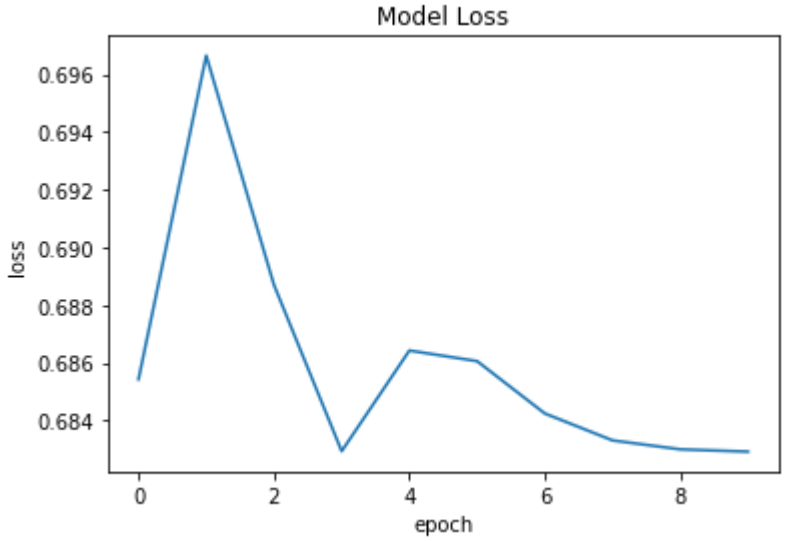
```
1/1 [==============================] - 0s 240ms/step - loss: 0.7520 - accuracy:
0.3333
```

Out[44]:
```
[0.7519607543945312, 0.3333333432674408]
```

In [45]:
```
plt.plot(history2.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [46]:
```
plt.plot(history2.history['loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```



In [47]:
```
res = model2.predict(trail_pad)
label = ['positive','negative']
print(res,label[np.argmax(trail_pad)>50])
```

```
1/1 [==============================] - 0s 315ms/step
[[0.5720007]] positive

C:\Users\2mscdsa28\AppData\Local\Temp\ipykernel_12452\479605851.py:3: Deprecati
onWarning: In future, it will be an error for 'np.bool_' scalars to be interpre
ted as an index
  print(res,label[np.argmax(trail_pad)>50])
```