NAME:Sujitha.R
ROLLNO:215229140

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

In [5]:
```
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 [6]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[6]: True

In [25]:
```
df=pd.read_csv('/content/txtclass.csv', encoding='unicode_escape')
```

In [26]:
```
df.head()
```

Out[26]:

	sentence	label
0	Change is the end result of all true learning.	1
1	An investment in knowledge pays the best inter...	1
2	The roots of education are bitter, but the fru...	1
3	Education is what remains after one has forgot...	1
4	The more that you read, the more things you wi...	1

In [27]:
```
df.columns
```

Out[27]: Index(['sentence', 'label'], dtype='object')

In [28]:
```
df.size
```

Out[28]: 40

pre-processing the Text

In [31]:
```
y = df['label']
x=[]
for review in df['sentence']:
  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 [32]:
```
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 [33]:
```
print(X_train.shape)
print(X_val.shape)
print(y_train.shape)
print(y_val.shape)
```

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

In [ ]:
```
import
```

In [48]:
```
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[48]:
```
{'<oov>': 1,
 'education': 2,
 'without': 3,
 '\x93education': 4,
 'one': 5,
 'learning': 6,
 'knowledge': 7,
 'remains': 8,
 'forgotten': 9,
 'learned': 10}
```

In [49]:
```
vocab_ze=len(train_token.word_index)+1
vocab_ze
```

Out[49]:
```
105
```

In [51]:
```
train_sequences[4]
```

Out[51]:
```
[49, 6, 1, 1, 1, 1]
```

In [52]:
```
train_padded = pad_sequences(train_sequences,maxlen=100,padding='post')
```

In [54]:
```
train_padded[8]
```

Out[54]:
```
array([14,  1,  1,  7, 15, 16,  1, 15, 16,  1,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
      dtype=int32)
```

In [55]:
```
train_padded.shape
```

Out[55]:
```
(14, 100)
```

In [56]:
```
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 [57]:
```
val_sequences[4]
```

Out[57]:
```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

In [58]:
```
val_padded=pad_sequences(val_sequences,maxlen=100,padding='post')
```

In [62]:
```
val_padded[2]
```

Out[62]:
```
array([49,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
      dtype=int32)
```

In [64]:
```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding,SimpleRNN
```

6.Model creation

In [65]:
```
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 [66]:
```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [67]:
```
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 [68]:
```
history=model.fit(train_padded,y_train,epochs=10,verbose=2,batch_size=15)
```

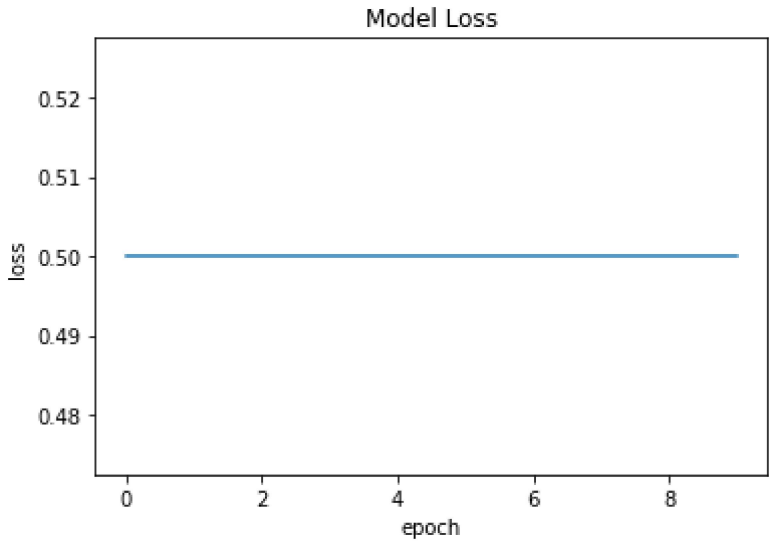
```
Epoch 1/10
1/1 - 2s - loss: 0.6932 - accuracy: 0.5000 - 2s/epoch - 2s/step
Epoch 2/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 28ms/epoch - 28ms/step
Epoch 3/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 27ms/epoch - 27ms/step
Epoch 4/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 27ms/epoch - 27ms/step
Epoch 5/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 31ms/epoch - 31ms/step
Epoch 6/10
1/1 - 0s - loss: 0.6931 - accuracy: 0.5000 - 29ms/epoch - 29ms/step
Epoch 7/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 27ms/epoch - 27ms/step
Epoch 8/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 27ms/epoch - 27ms/step
Epoch 9/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 27ms/epoch - 27ms/step
Epoch 10/10
1/1 - 0s - loss: 0.6931 - accuracy: 0.5000 - 31ms/epoch - 31ms/step
```

In [69]:
```
model.summary()
```

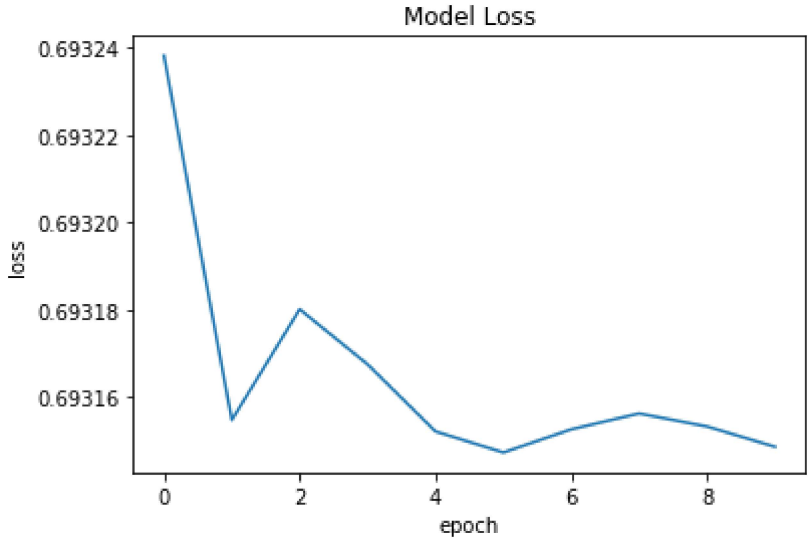
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 70)           21000

 simple_rnn (SimpleRNN)      (None, 70)                9870

 dense (Dense)               (None, 1)                 71

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

In [73]:
```
import matplotlib.pyplot as plt
```

In [74]:
```
plt.plot(history.history['accuracy'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```



In [75]:
```
plt.plot(history.history['loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```



In [77]:
```
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 [78]:
```
trail_pad
```

Out[78]:
```
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]], dtype=int32)
```

In [79]:
```
res = model.predict(trail_pad)
label = ['positive','negative']
print(res,label[np.argmax(trail_pad)>50])
```

```
[[0.49977595]] positive


/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarn
ing: In future, it will be an error for 'np.bool_' scalars to be interpreted as
an index
  This is separate from the ipykernel package so we can avoid doing imports unt
il
```

In [80]:
```
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 [81]:
```
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 [82]:
```
model1.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [83]:
```
history1=model1.fit(train_padded,y_train,epochs=10,verbose=2,batch_size=15)
```

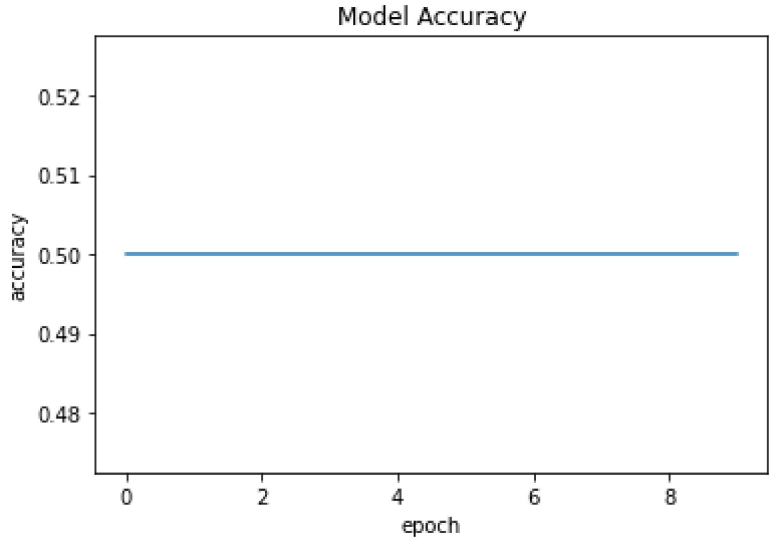
```
Epoch 1/10

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 - 3s - loss: 0.6932 - accuracy: 0.5000 - 3s/epoch - 3s/step
Epoch 2/10
1/1 - 0s - loss: 0.6935 - accuracy: 0.5000 - 24ms/epoch - 24ms/step
Epoch 3/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 21ms/epoch - 21ms/step
Epoch 4/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 22ms/epoch - 22ms/step
Epoch 5/10
1/1 - 0s - loss: 0.6933 - accuracy: 0.5000 - 23ms/epoch - 23ms/step
Epoch 6/10
1/1 - 0s - loss: 0.6933 - accuracy: 0.5000 - 24ms/epoch - 24ms/step
Epoch 7/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 25ms/epoch - 25ms/step
Epoch 8/10
1/1 - 0s - loss: 0.6931 - accuracy: 0.5000 - 21ms/epoch - 21ms/step
Epoch 9/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 23ms/epoch - 23ms/step
Epoch 10/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 24ms/epoch - 24ms/step
```

In [84]:
```
model1.evaluate(val_padded,y_val)
```

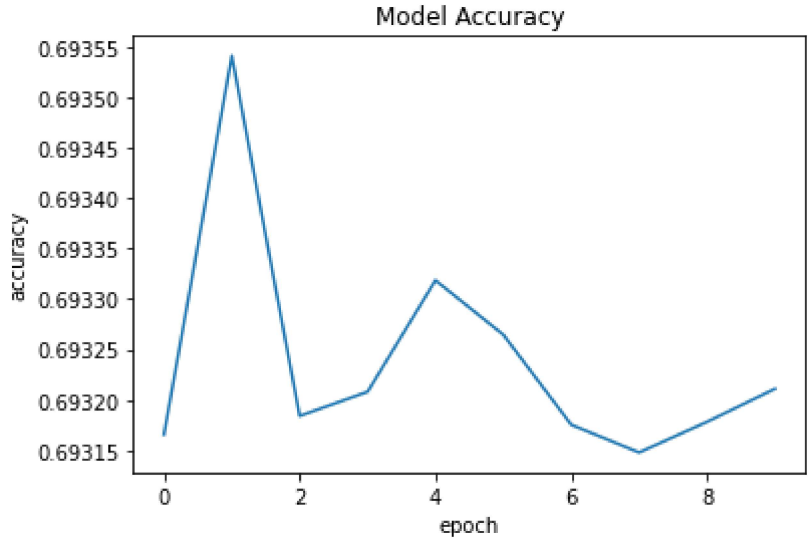
```
1/1 [==============================] - 1s 628ms/step - loss: 0.6932 - accuracy:
0.5000
```

Out[84]:
```
[0.6932110786437988, 0.5]
```

In [85]:
```
plt.plot(history1.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [86]:
```
plt.plot(history1.history['loss'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [87]:
```
res = model1.predict(trail_pad)
label = ['positive','negative']
print(res,label[np.argmax(trail_pad)>50])
```

```
[[0.49434516]] positive

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarn
ing: In future, it will be an error for 'np.bool_' scalars to be interpreted as
an index
  This is separate from the ipykernel package so we can avoid doing imports unt
il
```

In [88]:
```
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 [89]:
```
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 [90]:
```
model2.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [91]:
```
history2=model2.fit(train_padded,y_train,epochs=10,verbose=2,batch_size=15)
```

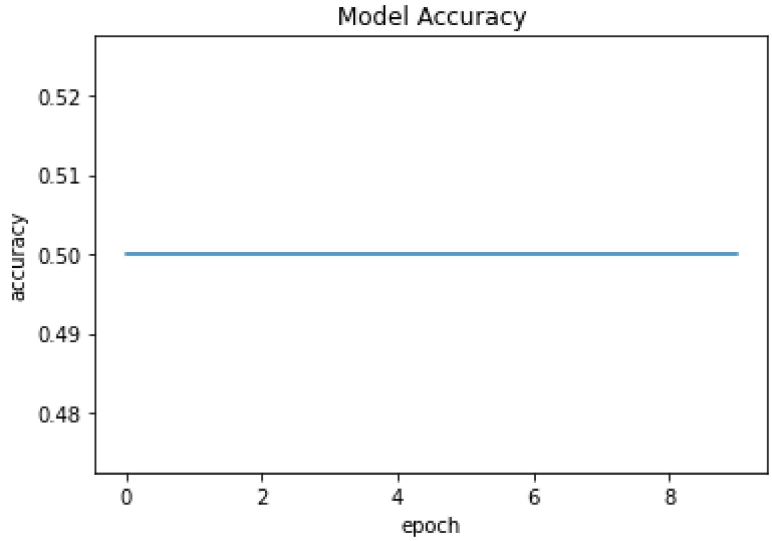
```
Epoch 1/10

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 - 2s - loss: 0.6948 - accuracy: 0.5000 - 2s/epoch - 2s/step
Epoch 2/10
1/1 - 0s - loss: 0.6985 - accuracy: 0.5000 - 36ms/epoch - 36ms/step
Epoch 3/10
1/1 - 0s - loss: 0.6939 - accuracy: 0.5000 - 33ms/epoch - 33ms/step
Epoch 4/10
1/1 - 0s - loss: 0.6940 - accuracy: 0.5000 - 34ms/epoch - 34ms/step
Epoch 5/10
1/1 - 0s - loss: 0.6937 - accuracy: 0.5000 - 34ms/epoch - 34ms/step
Epoch 6/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 34ms/epoch - 34ms/step
Epoch 7/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 41ms/epoch - 41ms/step
Epoch 8/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 33ms/epoch - 33ms/step
Epoch 9/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 44ms/epoch - 44ms/step
Epoch 10/10
1/1 - 0s - loss: 0.6932 - accuracy: 0.5000 - 35ms/epoch - 35ms/step
```

In [92]:
```
model2.evaluate(val_padded,y_val)
```

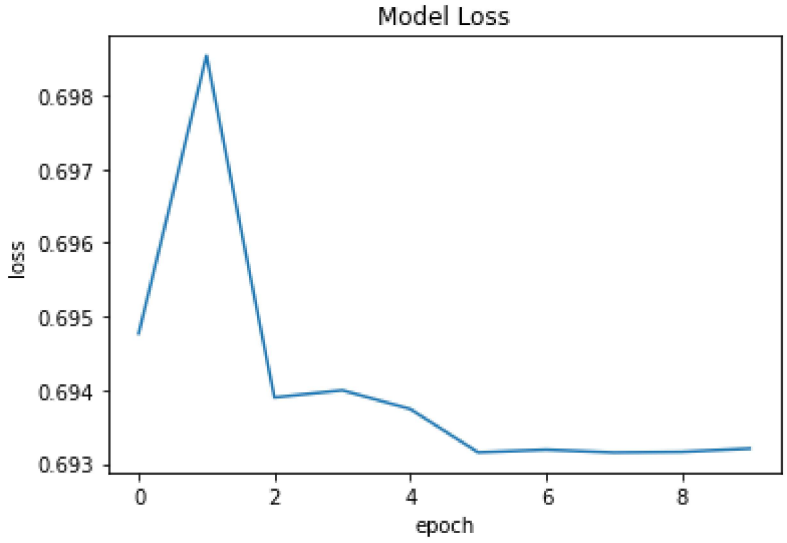
```
1/1 [==============================] - 1s 601ms/step - loss: 0.6932 - accuracy:
0.5000
```

Out[92]:
```
[0.6931962966918945, 0.5]
```

In [93]:
```
plt.plot(history2.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [94]:
```
plt.plot(history2.history['loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```



In [95]:
```
res = model2.predict(trail_pad)
label = ['positive','negative']
print(res,label[np.argmax(trail_pad)>50])
```

```
[[0.5049576]] positive

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarn
ing: In future, it will be an error for 'np.bool_' scalars to be interpreted as
an index
  This is separate from the ipykernel package so we can avoid doing imports unt
il
```

In [ ]: