In [37]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction import DictVectorizer
from sklearn.metrics import classification_report

import tensorflow.compat.v2 as tf

from tensorflow.python.platform import tf_logging as logging

from keras.models import Sequential
from keras import layers
```

In [ ]:
```
```

In [38]:
```
train_data = []
tags = []
with open('imdb_labelled.txt', 'r') as f:
    train_content = f.read()  # Raw Data without separation
    lines = train_content.split('\n')  # separates rows
    for line in lines:
        if line != "":
            tagged_words = line.split('\t')
            tup = (tagged_words[0],tagged_words[1])
            train_data.append(tup)

with open('amazon_cells_labelled.txt', 'r') as f:
    train_content = f.read()  # Raw Data without separation
    lines = train_content.split('\n')  # separates rows
    for line in lines:
        if line != "":
            tagged_words = line.split('\t')
            tup = (tagged_words[0],tagged_words[1])
            train_data.append(tup)

with open('yelp_labelled.txt', 'r') as f:
    train_content = f.read()  # Raw Data without separation
    lines = train_content.split('\n')  # separates rows
    for line in lines:
        if line != "":
            tagged_words = line.split('\t')
            tup = (tagged_words[0],tagged_words[1])
            train_data.append(tup)

#adds all 3 into training data!
```

In [39]:
```
# train test split
X = []
```

```
for i in range(0, 3000):
    X.append(train_data[i][0])
y = []
for i in range(0, 3000):
    y.append(train_data[i][1])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [40]:
```
## import stuff for word embeddings
from keras.preprocessing.text import Tokenizer
```

In [41]:
```
tk = Tokenizer(num_words=5000)
```

In [42]:
```
enc = tf.keras.layers.TextVectorization(max_tokens=1000)
enc.adapt(X_train)
```

In [43]:
```
tk.fit_on_texts(X_train)
```

In [44]:
```
X2_train = tk.texts_to_sequences(X_train)
X_test = tk.texts_to_sequences(X_test)
```

In [45]:
```
v_size = len(tk.word_index) + 1
```

In [46]:
```
from keras_preprocessing.sequence import pad_sequences
X2_train = pad_sequences(X2_train, padding='post', maxlen=100)
X_test = pad_sequences(X_test, padding='post', maxlen=100)
```

In [47]:
```
# we have to edit y train and test to be proper
y2_train = []
y2_test = []
for i in range(0, 2400):
    if (y_train[i] == '1'):
        y2_train.append(1)
    else:
        y2_train.append(0)

for i in range(0, 600):
    if (y_test[i] == '1'):
        y2_test.append(1)
    else:
        y2_test.append(0)
```

In [59]:
```
em_dim = 100
```

In [60]:
```
in_dim = X2_train.shape[1]
```

In [71]:
```
md2 = Sequential()
```

In [62]:
```
X2_train = np.asarray(X2_train)
y2_train = np.asarray(y2_train)
X_test = np.asarray(X_test)
y2_test = np.asarray(y2_test)
```

In [72]:
```
md2.add(layers.Embedding(v_size, em_dim, input_length=100)) #Build layers of model
md2.add(layers.SimpleRNN(128))
md2.add(layers.Dense(10, activation='relu'))
```

```
md2.add(layers.Dense(1, activation='sigmoid'))
md2.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [73]:
```
fit = md2.fit(X2_train, y2_train, epochs = 4, verbose = False, validation_data=(X_test
```

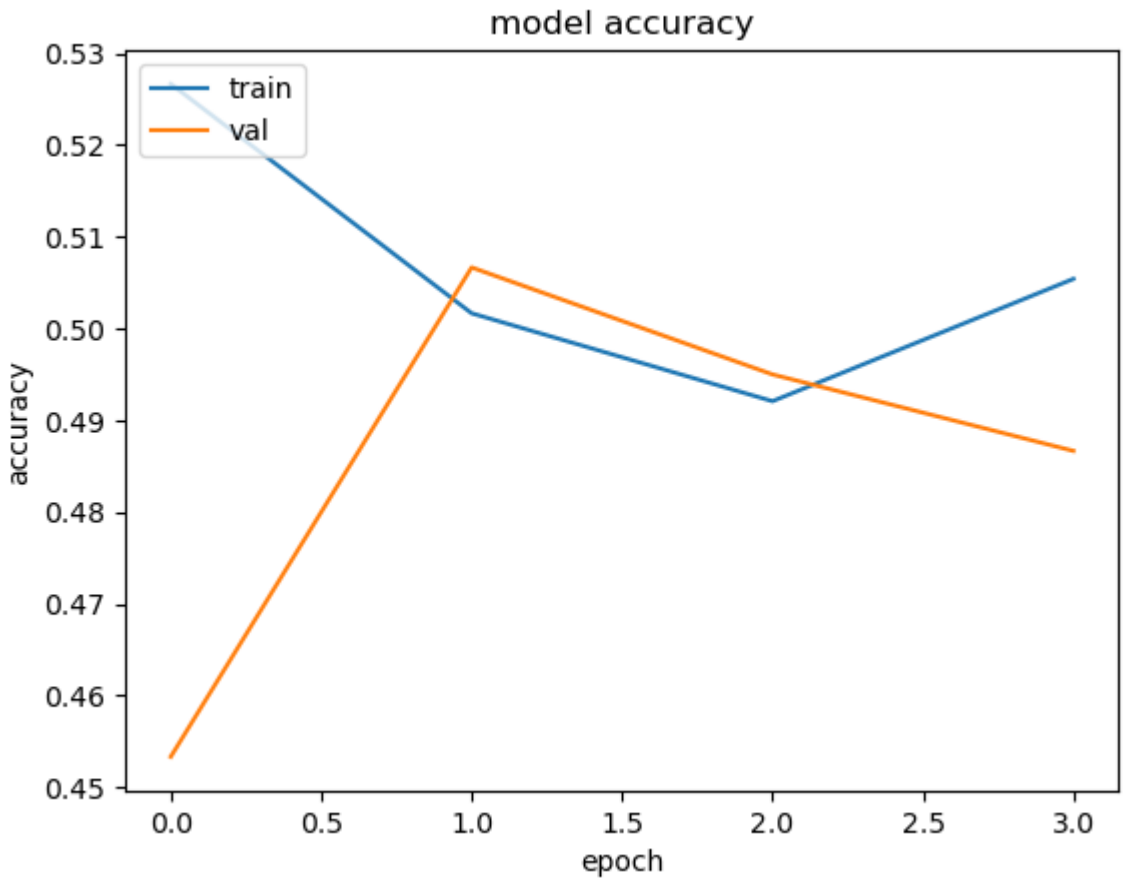
In [74]:
```
md2.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 100, 100)          471700

 simple_rnn_3 (SimpleRNN)    (None, 128)               29312

 dense_10 (Dense)            (None, 10)                1290

 dense_11 (Dense)            (None, 1)                 11

=================================================================
Total params: 502,313
Trainable params: 502,313
Non-trainable params: 0
_________________________________________________________________
```

In [75]:
```
loss, accuracy = md2.evaluate(X2_train, y2_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = md2.evaluate(X_test, y2_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
Training Accuracy: 0.5058
Testing Accuracy:  0.4867
```

In [76]:
```
plt.plot(fit.history['accuracy'])
plt.plot(fit.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```


model accuracy
train
val
accuracy
epoch
0.53
0.52
0.51
0.50
0.49
0.48
0.47
0.46
0.45
0.0
0.5
1.0
1.5
2.0
2.5
3.0

In [ ]: