Requirements

In [74]:

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
import re
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_sco
re, f1_score
```

QUESTION 1

In [247]:

```
dataset = pd.read_csv("urdu-sentiment-corpus-v1.tsv", sep = '\t')
print(dataset.head(10))
```

```
                                               Tweet Class
0  میں نے ایٹم بم بنایا ھے ....او بھائی ایٹم بمب...     P
1  چندے سے انقلاب اور عمران خان وزیر اعظم نہیں بن...    N
2                          ٹویٹر کا خیال کیسے آیا ؟     O
3  سرچ انجن گوگل کے نائب صدر نے فضا میں ، 130,000...     P
4  ابھی تک اسکی لہریں کبھی کبھی آ جاتی ہیں یار : أ...     P
5  گندی زبان اور گٹر جیسے دماغ والے جاھل جیالے ہ...     N
6  قاتل بھی تم مقتول بھی تم,ظالم بھی ہم اور مظلوم...     N
7  نور بغداد کی گلیوں کا ہے ہر ایک کرن اس کی مدین...     P
8  یہ لفظوں کی شرارت ھے سنبھل کر کچھ بھی لِکھنا...     P
9  سمارٹ فون کے عادیوں کے لیے ڈجٹل ڈیٹاکس کیپ متع...     P
```

Preprocessing

In [248]:

```
def sentence_preprocessing(sentences):
    # Preprocessing steps
    preprocessed_sentences = []

    for sentence in sentences:
        # Normalization: Convert to lowercase
        sentence = sentence.lower()

        # Removing Noise: Remove special characters and symbols
        sentence = re.sub(r'[^\w\s]', '', sentence)
        sentence = re.sub(r'\b\d+\b', '', sentence)

        with open('stopwords-ur.txt', 'r', encoding='utf-8') as f:
            stopwords_urdu = f.read().splitlines()
        sentence = ' '.join(word for word in sentence.split() if word not in stopwords_u
rdu)

        preprocessed_sentences.append(sentence)

    return preprocessed_sentences


sentences = dataset['Tweet'].tolist()
processed_sentences = sentence_preprocessing(sentences)
```

```
# Display the preprocessed sentences
for i, sentence in enumerate(processed_sentences[:5]):
    print(f"Original: {sentences[i]}")
    print(f"Processed: {sentence}")
    print()
```

```
Original: میں نے ایٹم بم بنایا ہے ....او بھائی ایٹم بمب کوٹ لکھپت والی اتفاق فیکٹری میں ن
ہیں بنتا.ایٹم بم کھوٹہ کی ایٹمی...
Processed: میں نے ایٹم بم بنایا ہے او بھائی ایٹم بمب کوٹ لکھپت والی اتفاق فیکٹری میں نہیں
بنتاایٹم بم کھوٹہ ایٹمی

Original: چندے سے انقلاب اور عمران خان وزیر اعظم نہیں بن سکتے
Processed: چندے سے انقلاب عمران خان وزیر اعظم نہیں بن سکتے

Original: ٹویٹر کا خیال کیسے آیا ؟
Processed: ٹویٹر کا خیال کیسے آیا

Original: سرچ انجن گوگل کے نائب صدر نے فضا میں ، 130,000 فٹ کی بلندی پر چھلانگ لگا کر عالم
ی ریکارڈ قائم کرلیا. چھلانگ کی...
Processed: سرچ انجن گوگل نائب صدر نے فضا میں فٹ بلندی چھلانگ لگا کر عالمی ریکارڈ قائم کرلی
ا چھلانگ

Original: ابھی تک اسکی لہریں کبھی کبھی آ جاتی ہیں یار :أ ْ
Processed: ابھی اسکی لہریں کبھی کبھی آ جاتی یار أ
```

In [249]:

```
dataset['Class'] = dataset['Class'].replace('O','P')
```

Train Test Split

In [257]:

```
labels = dataset['Class'].tolist()

# Separate out the sentences and labels into training and test sets
training_size = int(len(sentences) * 0.75)

training_sentences = processed_sentences[0:training_size]
testing_sentences = processed_sentences[training_size:]
training_labels = labels[0:training_size]
testing_labels = labels[training_size:]

# Make labels into numpy arrays for use with the network later
training_labels_final = np.array(training_labels)
testing_labels_final = np.array(testing_labels)
```

In [259]:

```
# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit label encoder on the training labels
label_encoder.fit(training_labels)

# Encode training labels
training_labels_final = label_encoder.transform(training_labels)

# Encode testing labels
testing_labels_final = label_encoder.transform(testing_labels)
```

Tokenization

In [68]:

```
embedding_dim = 16
trunc_type='post'
```

```
padding_type='post'
oov_tok = "<OOV>"

tokenizer = Tokenizer(oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
vocab_size = len(tokenizer.word_index) + 1  # Add 1 for padding token (if used)
print("Vocabulary size:", vocab_size)
training_sequences = tokenizer.texts_to_sequences(training_sentences)
max_pad_len = max([len(seq) for seq in training_sequences])
print("Max paded sequence length: ",max_pad_len)
training_padded = pad_sequences(training_sequences, maxlen=max_pad_len, padding=padding_t
ype, truncating=trunc_type)

testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences, maxlen=max_pad_len, padding=padding_ty
pe, truncating=trunc_type)
```

```
Vocabulary size: 4086
Max paded sequence length:  30
```

RNN (2 Layers and 0.3 Dropout)

In [75]:

```
model_rnn_1 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.SimpleRNN(units=16, activation='tanh', return_sequences=True),   # Fi
rst layer
    tf.keras.layers.SimpleRNN(units=32, activation='tanh'),   # Second layer
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_rnn_1.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_rnn_1.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_5 (Embedding)     (None, 30, 16)            65376     
                                                                 
 simple_rnn_10 (SimpleRNN)   (None, 30, 16)            528       
                                                                 
 simple_rnn_11 (SimpleRNN)   (None, 32)                1568      
                                                                 
 dropout_3 (Dropout)         (None, 32)                0         
                                                                 
 dense_5 (Dense)             (None, 1)                 33        
                                                                 
=================================================================
Total params: 67,505
Trainable params: 67,505
Non-trainable params: 0
_________________________________________________________________
```

In [80]:

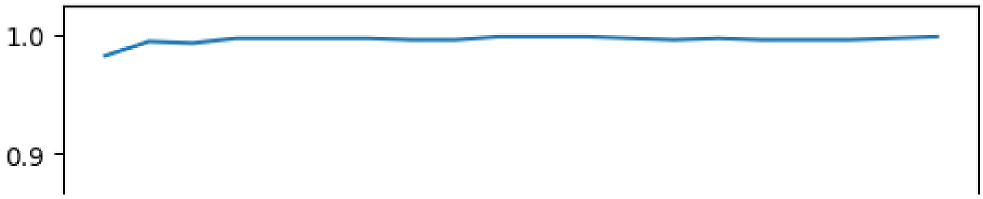
```
num_epochs = 20
history_rnn_1 = model_rnn_1.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

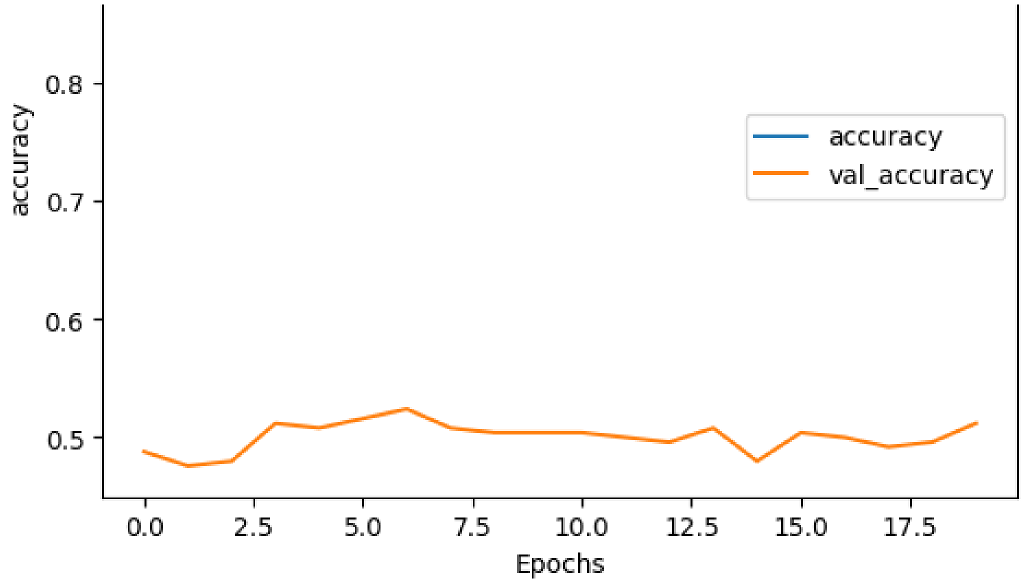
```
Epoch 1/20
24/24 [==============================] - 1s 23ms/step - loss: 0.0270 - accuracy: 0.9827 -
val_loss: 2.4999 - val_accuracy: 0.4880
Epoch 2/20
24/24 [==============================] - 1s 21ms/step - loss: -0.0036 - accuracy: 0.9947
- val_loss: 2.2728 - val_accuracy: 0.4760
Epoch 3/20
24/24 [==============================] - 0s 18ms/step - loss: 0.0031 - accuracy: 0.9933 -
val_loss: 2.1635 - val_accuracy: 0.4800
```

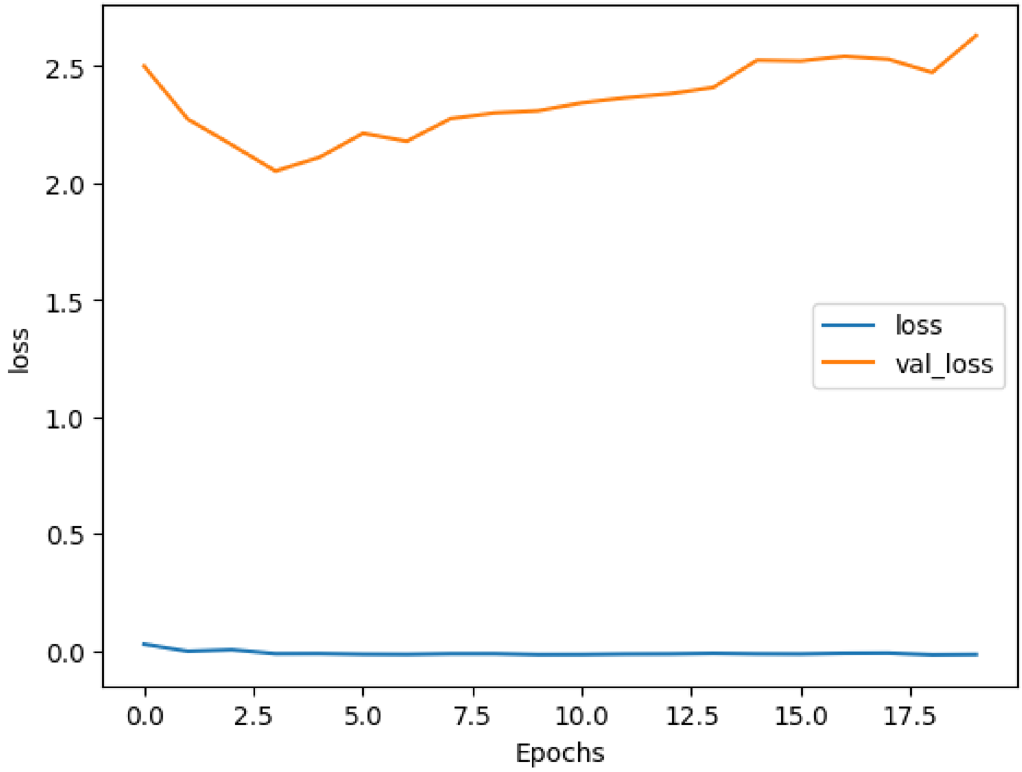
```
val_loss: 2.1055 - val_accuracy: 0.4800
Epoch 4/20
24/24 [==============================] - 0s 16ms/step - loss: -0.0136 - accuracy: 0.9973
- val_loss: 2.0510 - val_accuracy: 0.5120
Epoch 5/20
24/24 [==============================] - 0s 15ms/step - loss: -0.0130 - accuracy: 0.9973
- val_loss: 2.1094 - val_accuracy: 0.5080
Epoch 6/20
24/24 [==============================] - 0s 15ms/step - loss: -0.0160 - accuracy: 0.9973
- val_loss: 2.2122 - val_accuracy: 0.5160
Epoch 7/20
24/24 [==============================] - 0s 16ms/step - loss: -0.0167 - accuracy: 0.9973
- val_loss: 2.1785 - val_accuracy: 0.5240
Epoch 8/20
24/24 [==============================] - 0s 15ms/step - loss: -0.0137 - accuracy: 0.9960
- val_loss: 2.2753 - val_accuracy: 0.5080
Epoch 9/20
24/24 [==============================] - 0s 16ms/step - loss: -0.0136 - accuracy: 0.9960
- val_loss: 2.2995 - val_accuracy: 0.5040
Epoch 10/20
24/24 [==============================] - 0s 15ms/step - loss: -0.0178 - accuracy: 0.9987
- val_loss: 2.3089 - val_accuracy: 0.5040
Epoch 11/20
24/24 [==============================] - 0s 15ms/step - loss: -0.0174 - accuracy: 0.9987
- val_loss: 2.3428 - val_accuracy: 0.5040
Epoch 12/20
24/24 [==============================] - 0s 16ms/step - loss: -0.0151 - accuracy: 0.9987
- val_loss: 2.3647 - val_accuracy: 0.5000
Epoch 13/20
24/24 [==============================] - 0s 16ms/step - loss: -0.0145 - accuracy: 0.9973
- val_loss: 2.3812 - val_accuracy: 0.4960
Epoch 14/20
24/24 [==============================] - 0s 13ms/step - loss: -0.0119 - accuracy: 0.9960
- val_loss: 2.4087 - val_accuracy: 0.5080
Epoch 15/20
24/24 [==============================] - 0s 12ms/step - loss: -0.0142 - accuracy: 0.9973
- val_loss: 2.5248 - val_accuracy: 0.4800
Epoch 16/20
24/24 [==============================] - 0s 11ms/step - loss: -0.0148 - accuracy: 0.9960
- val_loss: 2.5216 - val_accuracy: 0.5040
Epoch 17/20
24/24 [==============================] - 0s 13ms/step - loss: -0.0115 - accuracy: 0.9960
- val_loss: 2.5423 - val_accuracy: 0.5000
Epoch 18/20
24/24 [==============================] - 0s 13ms/step - loss: -0.0107 - accuracy: 0.9960
- val_loss: 2.5290 - val_accuracy: 0.4920
Epoch 19/20
24/24 [==============================] - 0s 13ms/step - loss: -0.0187 - accuracy: 0.9973
- val_loss: 2.4729 - val_accuracy: 0.4960
Epoch 20/20
24/24 [==============================] - 0s 12ms/step - loss: -0.0173 - accuracy: 0.9987
- val_loss: 2.6298 - val_accuracy: 0.5120
```

In [82]:

```python
def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()

plot_graphs(history_rnn_1, "accuracy")
plot_graphs(history_rnn_1, "loss")
```



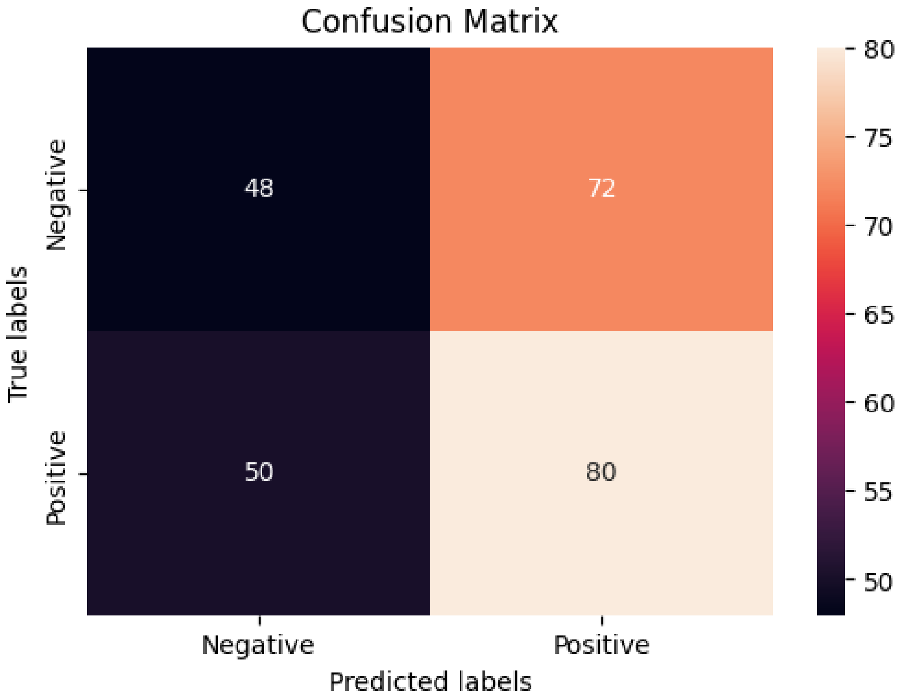




In [85]:

```
# Predict labels for testing data
predicted_labels = (model_rnn_1.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 0s 7ms/step
Accuracy: 0.512
Precision: 0.5263157894736842
Recall: 0.6153846153846154
F1-score: 0.5673758865248227
```



RNN (2 Layers and 0.7 Dropout)

In [86]:

```
model_rnn_2 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.SimpleRNN(units=16, activation='tanh', return_sequences=True),   # Fi
rst layer
    tf.keras.layers.SimpleRNN(units=32, activation='tanh'),   # Second layer
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_rnn_2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_rnn_2.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 30, 16)            65376

 simple_rnn_12 (SimpleRNN)   (None, 30, 16)            528

 simple_rnn_13 (SimpleRNN)   (None, 32)                1568

 dropout_4 (Dropout)         (None, 32)                0

 dense_6 (Dense)             (None, 1)                 33

=================================================================
Total params: 67,505
Trainable params: 67,505
Non-trainable params: 0
_________________________________________________________________
```

In [87]:

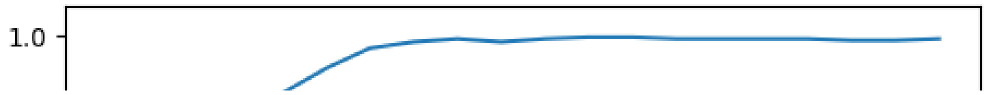
```
num_epochs = 20
```

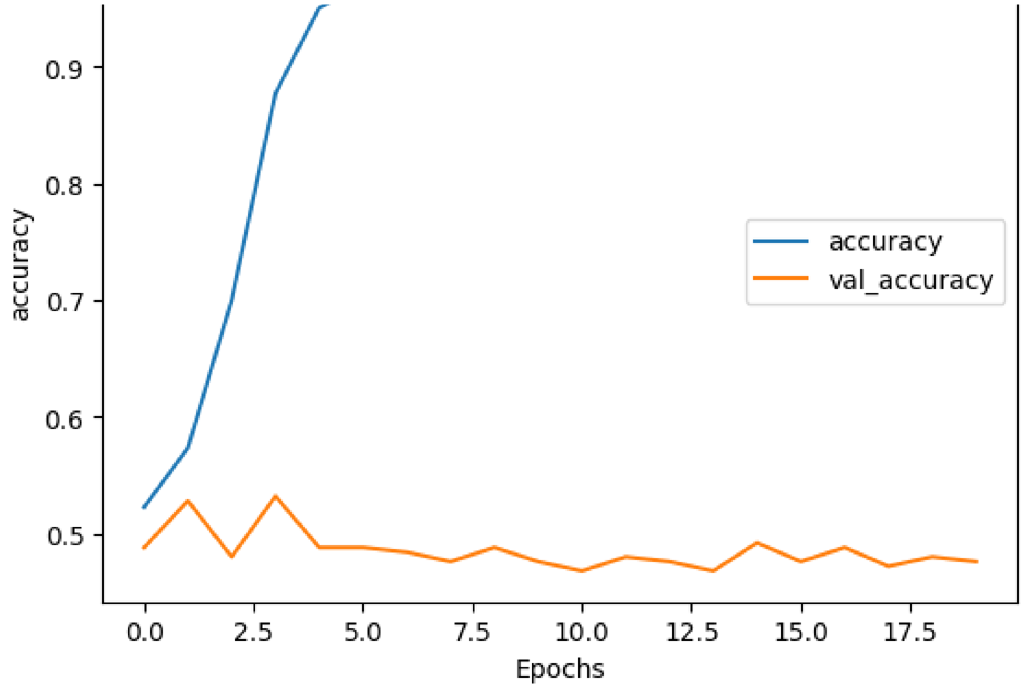
```
history_rnn_2 = model_rnn_2.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

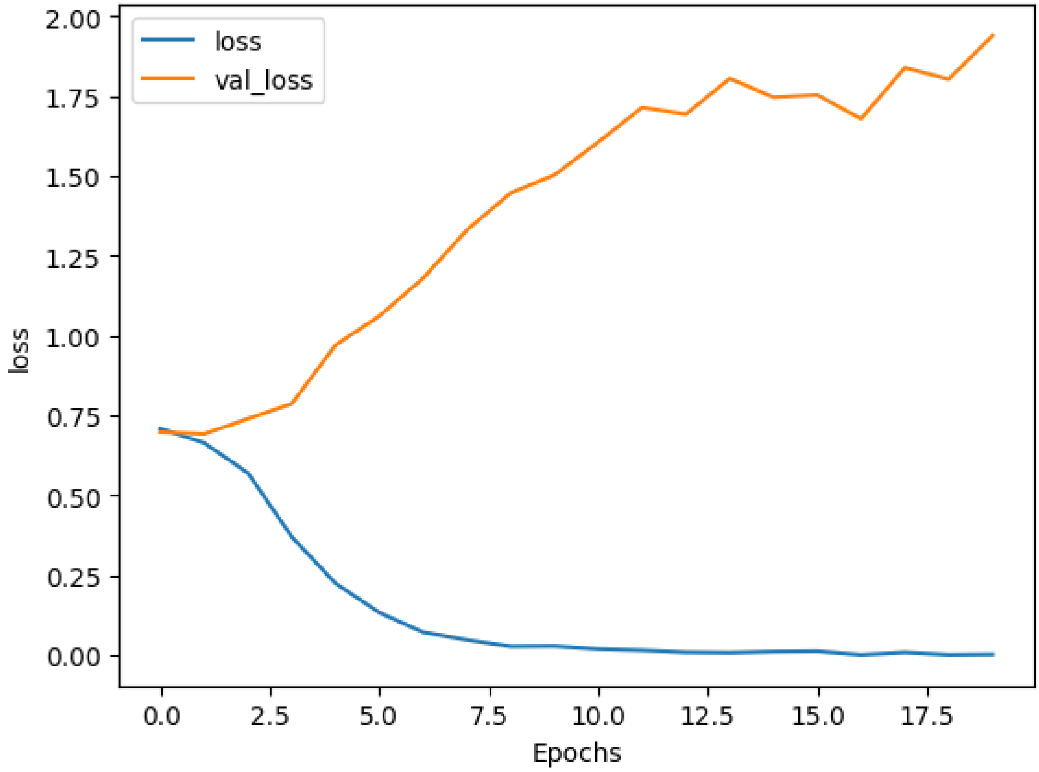
```
Epoch 1/20
24/24 [==============================] - 4s 36ms/step - loss: 0.7093 - accuracy: 0.5227 -
val_loss: 0.6978 - val_accuracy: 0.4880
Epoch 2/20
24/24 [==============================] - 0s 17ms/step - loss: 0.6641 - accuracy: 0.5733 -
val_loss: 0.6924 - val_accuracy: 0.5280
Epoch 3/20
24/24 [==============================] - 0s 15ms/step - loss: 0.5700 - accuracy: 0.7000 -
val_loss: 0.7397 - val_accuracy: 0.4800
Epoch 4/20
24/24 [==============================] - 0s 16ms/step - loss: 0.3703 - accuracy: 0.8773 -
val_loss: 0.7865 - val_accuracy: 0.5320
Epoch 5/20
24/24 [==============================] - 0s 14ms/step - loss: 0.2244 - accuracy: 0.9507 -
val_loss: 0.9702 - val_accuracy: 0.4880
Epoch 6/20
24/24 [==============================] - 0s 15ms/step - loss: 0.1327 - accuracy: 0.9720 -
val_loss: 1.0613 - val_accuracy: 0.4880
Epoch 7/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0714 - accuracy: 0.9893 -
val_loss: 1.1799 - val_accuracy: 0.4840
Epoch 8/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0470 - accuracy: 0.9947 -
val_loss: 1.3307 - val_accuracy: 0.4760
Epoch 9/20
24/24 [==============================] - 0s 13ms/step - loss: 0.0265 - accuracy: 0.9973 -
val_loss: 1.4464 - val_accuracy: 0.4880
Epoch 10/20
24/24 [==============================] - 0s 11ms/step - loss: 0.0275 - accuracy: 0.9947 -
val_loss: 1.5033 - val_accuracy: 0.4760
Epoch 11/20
24/24 [==============================] - 0s 12ms/step - loss: 0.0183 - accuracy: 0.9973 -
val_loss: 1.6061 - val_accuracy: 0.4680
Epoch 12/20
24/24 [==============================] - 0s 12ms/step - loss: 0.0145 - accuracy: 0.9987 -
val_loss: 1.7142 - val_accuracy: 0.4800
Epoch 13/20
24/24 [==============================] - 0s 12ms/step - loss: 0.0085 - accuracy: 0.9987 -
val_loss: 1.6934 - val_accuracy: 0.4760
Epoch 14/20
24/24 [==============================] - 0s 13ms/step - loss: 0.0071 - accuracy: 0.9973 -
val_loss: 1.8049 - val_accuracy: 0.4680
Epoch 15/20
24/24 [==============================] - 0s 12ms/step - loss: 0.0106 - accuracy: 0.9973 -
val_loss: 1.7466 - val_accuracy: 0.4920
Epoch 16/20
24/24 [==============================] - 0s 12ms/step - loss: 0.0118 - accuracy: 0.9973 -
val_loss: 1.7536 - val_accuracy: 0.4760
Epoch 17/20
24/24 [==============================] - 0s 12ms/step - loss: 3.4325e-04 - accuracy: 0.99
73 - val_loss: 1.6789 - val_accuracy: 0.4880
Epoch 18/20
24/24 [==============================] - 0s 12ms/step - loss: 0.0081 - accuracy: 0.9960 -
val_loss: 1.8389 - val_accuracy: 0.4720
Epoch 19/20
24/24 [==============================] - 0s 12ms/step - loss: 4.5338e-04 - accuracy: 0.99
60 - val_loss: 1.8028 - val_accuracy: 0.4800
Epoch 20/20
24/24 [==============================] - 0s 14ms/step - loss: 0.0017 - accuracy: 0.9973 -
val_loss: 1.9390 - val_accuracy: 0.4760
```

In [88]:

```
plot_graphs(history_rnn_2, "accuracy")
plot_graphs(history_rnn_2, "loss")
```



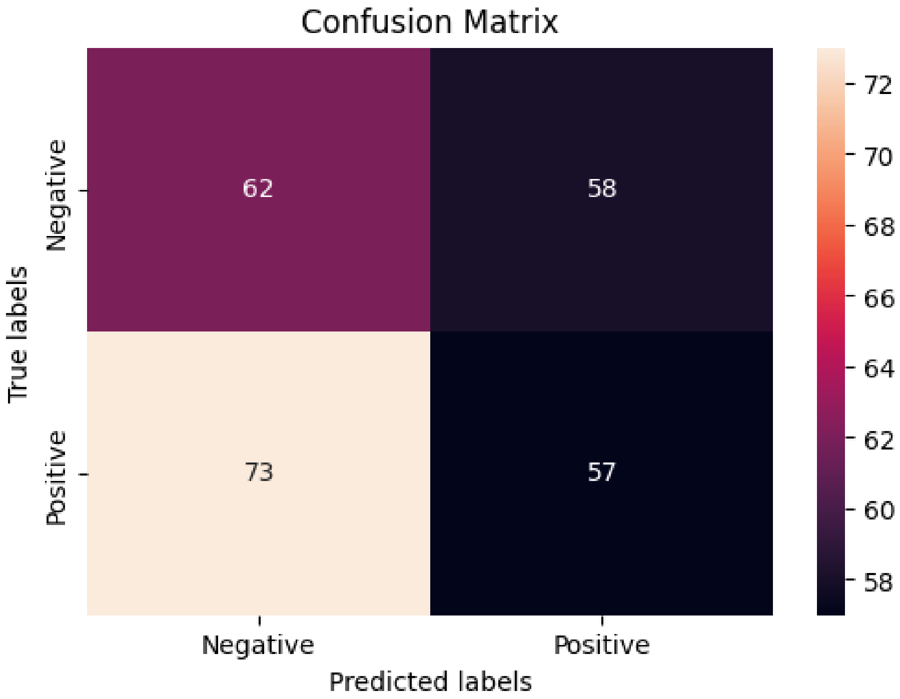




In [89]:

```
# Predict labels for testing data
predicted_labels = (model_rnn_2.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 0s 8ms/step
Accuracy: 0.476
Precision: 0.4956521739130435
Recall: 0.43846153846153846
F1-score: 0.46530612244897956
```



RNN (3 Layers and 0.3 Dropout)

In [97]:

```
model_rnn_3 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.SimpleRNN(units=16, activation='tanh', return_sequences=True),   # Fi
rst layer
    tf.keras.layers.SimpleRNN(units=32, activation='tanh', return_sequences=True),   # Se
cond layer
    tf.keras.layers.SimpleRNN(units=16, activation='tanh'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_rnn_3.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_rnn_3.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_10 (Embedding)    (None, 30, 16)            65376     
                                                                 
 simple_rnn_23 (SimpleRNN)   (None, 30, 16)            528       
                                                                 
 simple_rnn_24 (SimpleRNN)   (None, 30, 32)            1568      
                                                                 
 simple_rnn_25 (SimpleRNN)   (None, 16)                784       
                                                                 
 dropout_8 (Dropout)         (None, 16)                0         
                                                                 
 dense_10 (Dense)            (None, 1)                 17        
                                                                 
=================================================================
Total params: 68,273
Total params: 68,273
Trainable params: 68,273
Non-trainable params: 0
_________________________________________________________________
```

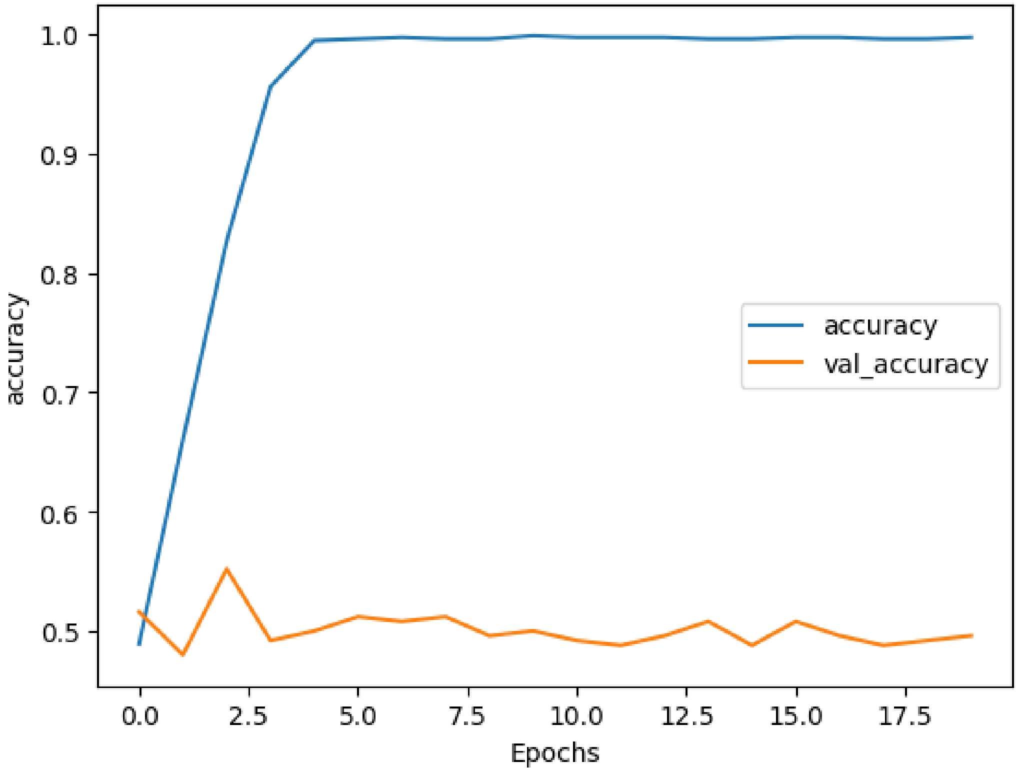
In [98]:

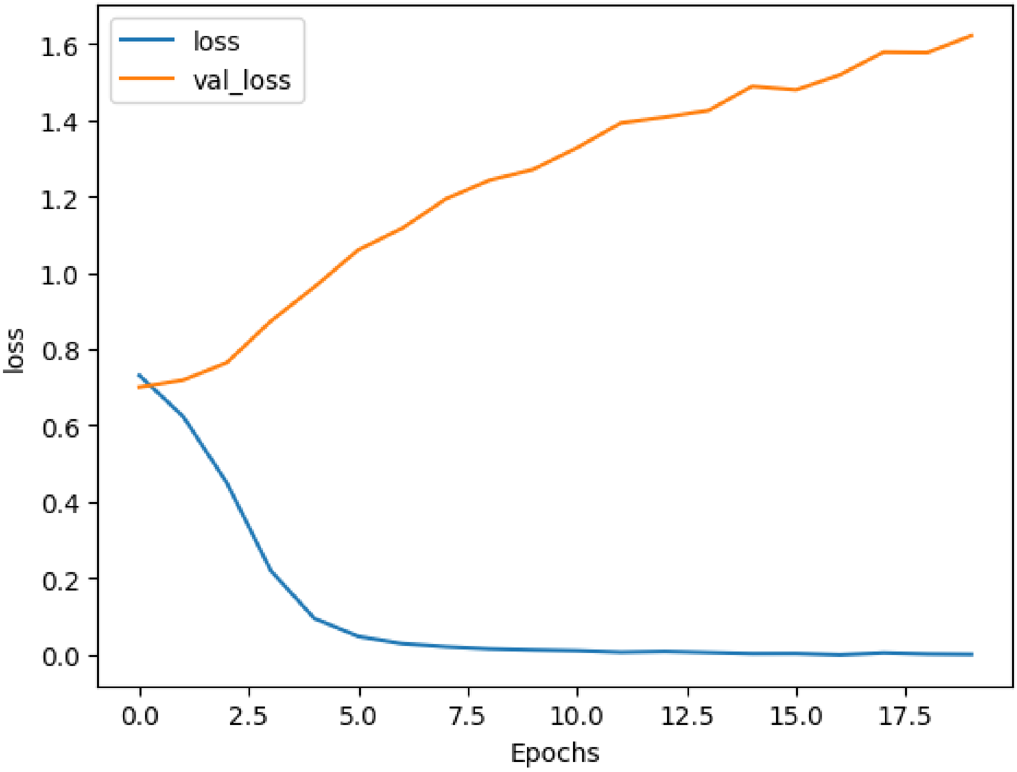
```
num_epochs = 20
history_rnn_3 = model_rnn_3.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

```
Epoch 1/20
24/24 [==============================] - 5s 51ms/step - loss: 0.7315 - accuracy: 0.4893 -
val_loss: 0.7004 - val_accuracy: 0.5160
Epoch 2/20
24/24 [==============================] - 0s 20ms/step - loss: 0.6238 - accuracy: 0.6600 -
val_loss: 0.7193 - val_accuracy: 0.4800
Epoch 3/20
24/24 [==============================] - 0s 20ms/step - loss: 0.4496 - accuracy: 0.8267 -
val_loss: 0.7647 - val_accuracy: 0.5520
Epoch 4/20
24/24 [==============================] - 1s 21ms/step - loss: 0.2203 - accuracy: 0.9560 -
val_loss: 0.8732 - val_accuracy: 0.4920
Epoch 5/20
24/24 [==============================] - 0s 17ms/step - loss: 0.0945 - accuracy: 0.9947 -
val_loss: 0.9631 - val_accuracy: 0.5000
Epoch 6/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0476 - accuracy: 0.9960 -
val_loss: 1.0601 - val_accuracy: 0.5120
Epoch 7/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0290 - accuracy: 0.9973 -
val_loss: 1.1163 - val_accuracy: 0.5080
Epoch 8/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0206 - accuracy: 0.9960 -
val_loss: 1.1935 - val_accuracy: 0.5120
Epoch 9/20
24/24 [==============================] - 0s 17ms/step - loss: 0.0150 - accuracy: 0.9960 -
val_loss: 1.2427 - val_accuracy: 0.4960
Epoch 10/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0123 - accuracy: 0.9987 -
val_loss: 1.2711 - val_accuracy: 0.5000
Epoch 11/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0101 - accuracy: 0.9973 -
val_loss: 1.3278 - val_accuracy: 0.4920
Epoch 12/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0062 - accuracy: 0.9973 -
val_loss: 1.3926 - val_accuracy: 0.4880
Epoch 13/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0081 - accuracy: 0.9973 -
val_loss: 1.4077 - val_accuracy: 0.4960
Epoch 14/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0054 - accuracy: 0.9960 -
val_loss: 1.4253 - val_accuracy: 0.5080
Epoch 15/20
24/24 [==============================] - 0s 17ms/step - loss: 0.0027 - accuracy: 0.9960 -
val_loss: 1.4886 - val_accuracy: 0.4880
Epoch 16/20
24/24 [==============================] - 0s 18ms/step - loss: 0.0030 - accuracy: 0.9973 -
val_loss: 1.4791 - val_accuracy: 0.5080
Epoch 17/20
24/24 [==============================] - 0s 16ms/step - loss: -6.5542e-04 - accuracy: 0.9
973 - val_loss: 1.5187 - val_accuracy: 0.4960
Epoch 18/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0047 - accuracy: 0.9960 -
val_loss: 1.5782 - val_accuracy: 0.4880
Epoch 19/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0017 - accuracy: 0.9960 -
val_loss: 1.5771 - val_accuracy: 0.4920
Epoch 20/20
24/24 [==============================] - 0s 19ms/step - loss: 6.8491e-04 - accuracy: 0.99
73 - val_loss: 1.6210 - val_accuracy: 0.4960
```

In [99]:

```
plot_graphs(history_rnn_3, "accuracy")
plot_graphs(history_rnn_3, "loss")
```

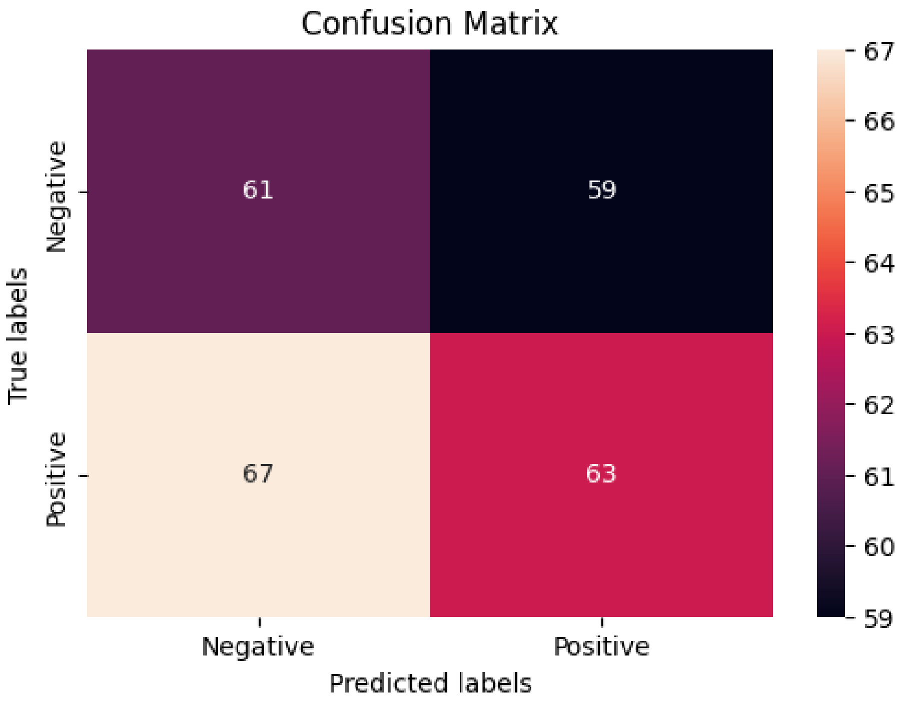




In [100]:

```
# Predict labels for testing data
predicted_labels = (model_rnn_3.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=["Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 1s 10ms/step
Accuracy: 0.496
Precision: 0.5163934426229508
Recall: 0.4846153846153846
F1-score: 0.5
```



RNN (3 Layers and 0.7 Dropout)

In [102]:

```
model_rnn_4 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.SimpleRNN(units=16, activation='tanh', return_sequences=True),   # First layer
    tf.keras.layers.SimpleRNN(units=32, activation='tanh', return_sequences=True),   # Second layer
    tf.keras.layers.SimpleRNN(units=16, activation='tanh'),
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_rnn_4.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_rnn_4.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, 30, 16)            65376

 simple_rnn_26 (SimpleRNN)   (None, 30, 16)            528

 simple_rnn_27 (SimpleRNN)   (None, 30, 32)            1568
 simple_rnn_28 (SimpleRNN)   (None, 16)                784       
                                                                 
 dropout_9 (Dropout)         (None, 16)                0         
                                                                 
 dense_11 (Dense)            (None, 1)                 17        
                                                                 
=================================================================
Total params: 68,273
Trainable params: 68,273
Non-trainable params: 0
_________________________________________________________________
```

In [103]:

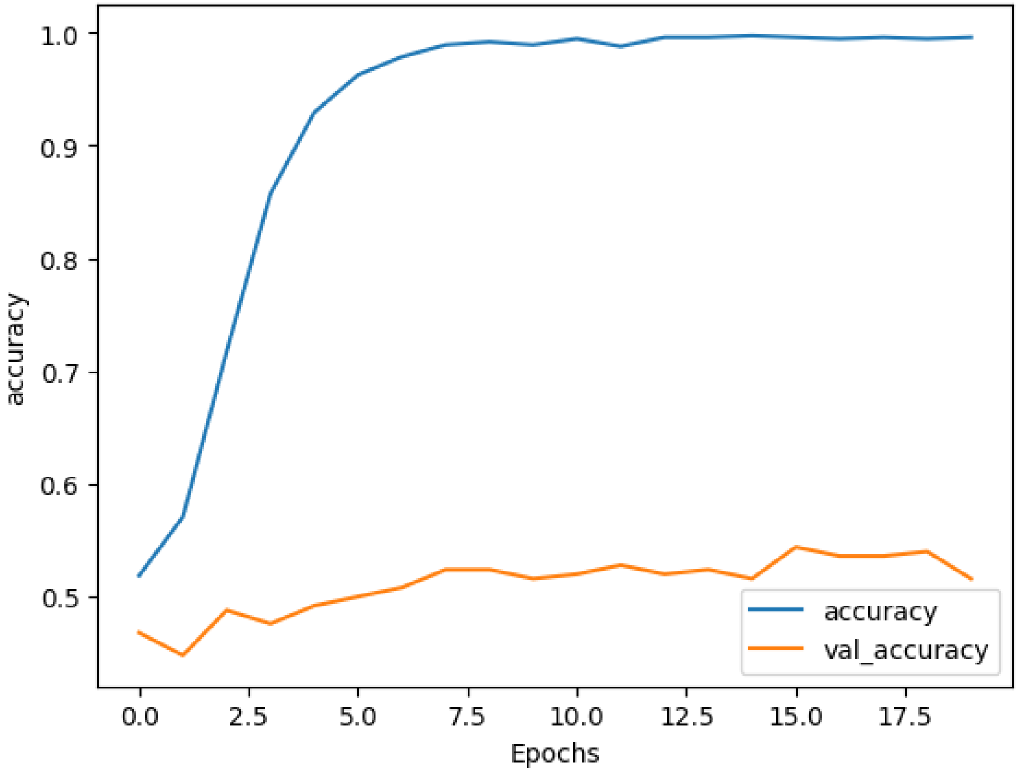
```
num_epochs = 20
history_rnn_4 = model_rnn_4.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

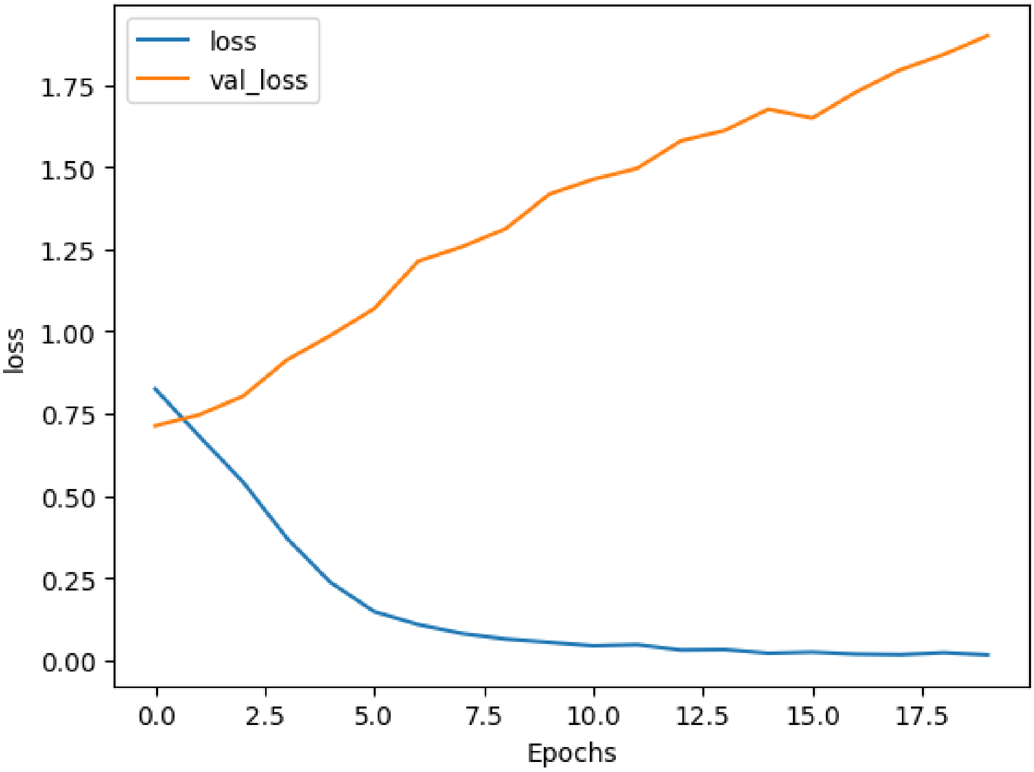
```
Epoch 1/20
24/24 [==============================] - 6s 46ms/step - loss: 0.8250 - accuracy: 0.5187 -
val_loss: 0.7131 - val_accuracy: 0.4680
Epoch 2/20
24/24 [==============================] - 0s 19ms/step - loss: 0.6816 - accuracy: 0.5707 -
val_loss: 0.7466 - val_accuracy: 0.4480
Epoch 3/20
24/24 [==============================] - 0s 16ms/step - loss: 0.5428 - accuracy: 0.7173 -
val_loss: 0.8027 - val_accuracy: 0.4880
Epoch 4/20
24/24 [==============================] - 0s 15ms/step - loss: 0.3720 - accuracy: 0.8573 -
val_loss: 0.9128 - val_accuracy: 0.4760
Epoch 5/20
24/24 [==============================] - 0s 16ms/step - loss: 0.2368 - accuracy: 0.9293 -
val_loss: 0.9877 - val_accuracy: 0.4920
Epoch 6/20
24/24 [==============================] - 0s 16ms/step - loss: 0.1481 - accuracy: 0.9627 -
val_loss: 1.0694 - val_accuracy: 0.5000
Epoch 7/20
24/24 [==============================] - 0s 16ms/step - loss: 0.1088 - accuracy: 0.9787 -
val_loss: 1.2132 - val_accuracy: 0.5080
Epoch 8/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0813 - accuracy: 0.9893 -
val_loss: 1.2575 - val_accuracy: 0.5240
Epoch 9/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0647 - accuracy: 0.9920 -
val_loss: 1.3119 - val_accuracy: 0.5240
Epoch 10/20
24/24 [==============================] - 0s 18ms/step - loss: 0.0549 - accuracy: 0.9893 -
val_loss: 1.4176 - val_accuracy: 0.5160
Epoch 11/20
24/24 [==============================] - 0s 18ms/step - loss: 0.0442 - accuracy: 0.9947 -
val_loss: 1.4626 - val_accuracy: 0.5200
Epoch 12/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0476 - accuracy: 0.9880 -
val_loss: 1.4957 - val_accuracy: 0.5280
Epoch 13/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0315 - accuracy: 0.9960 -
val_loss: 1.5794 - val_accuracy: 0.5200
Epoch 14/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0327 - accuracy: 0.9960 -
val_loss: 1.6109 - val_accuracy: 0.5240
Epoch 15/20
24/24 [==============================] - 0s 19ms/step - loss: 0.0216 - accuracy: 0.9973 -
val_loss: 1.6759 - val_accuracy: 0.5160
Epoch 16/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0250 - accuracy: 0.9960 -
val_loss: 1.6488 - val_accuracy: 0.5440
Epoch 17/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0190 - accuracy: 0.9947 -
val_loss: 1.7276 - val_accuracy: 0.5360
Epoch 18/20
24/24 [==============================] - 0s 15ms/step - loss: 0.0172 - accuracy: 0.9960 -
val_loss: 1.7948 - val_accuracy: 0.5360
```

```
val_loss: 1.7948 - val_accuracy: 0.5360
Epoch 19/20
24/24 [==============================] - 0s 16ms/step - loss: 0.0234 - accuracy: 0.9947 -
val_loss: 1.8419 - val_accuracy: 0.5400
Epoch 20/20
24/24 [==============================] - 0s 19ms/step - loss: 0.0166 - accuracy: 0.9960 -
val_loss: 1.8989 - val_accuracy: 0.5160
```

In [105]:

```
plot_graphs(history_rnn_4, "accuracy")
plot_graphs(history_rnn_4, "loss")
```



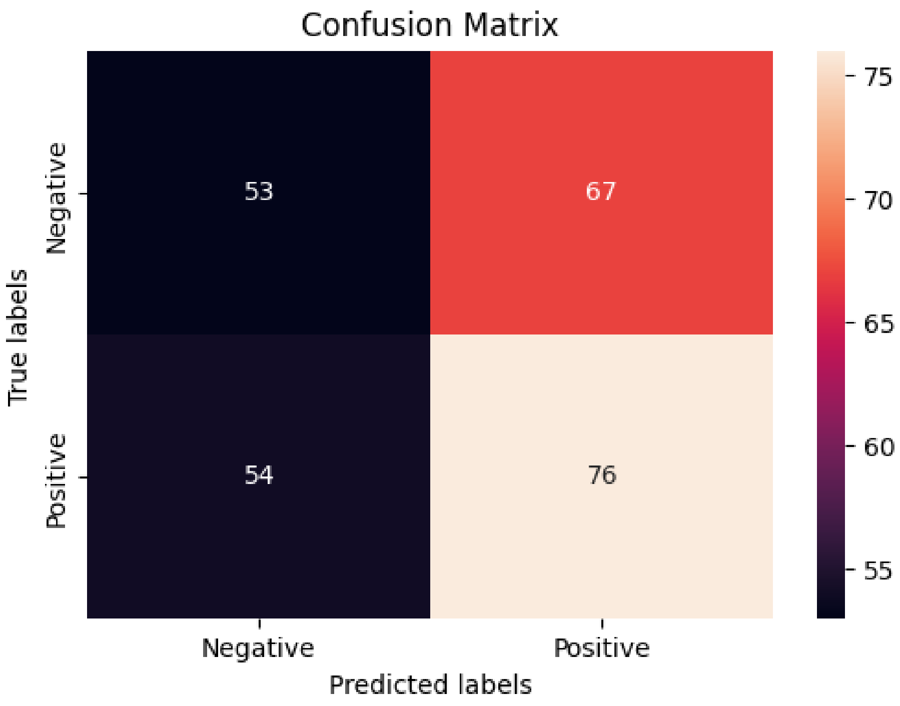


In [107]:

```
# Predict labels for testing data
predicted_labels = (model_rnn_4.predict(testing_padded) > 0.5).astype("int32")
```

```
# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 1s 7ms/step
Accuracy: 0.516
Precision: 0.5314685314685315
Recall: 0.5846153846153846
F1-score: 0.5567765567765569
```



GRU (2 Layers and 0.3 Dropout)

In [118]:

```
model_gru_1 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.GRU(units=64, return_sequences=True),  # First layer
    tf.keras.layers.GRU(units=128),  # Second layer
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_gru_1.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_gru_1.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_14 (Embedding)    (None, 30, 16)            65376
 embedding_14 (Embedding)    (None, 30, 16)            65376

 gru_4 (GRU)                 (None, 30, 64)            15744

 gru_5 (GRU)                 (None, 128)               74496

 dropout_12 (Dropout)        (None, 128)               0

 dense_14 (Dense)            (None, 1)                 129

=================================================================
Total params: 155,745
Trainable params: 155,745
Non-trainable params: 0
_________________________________________________________________
```

In [119]:

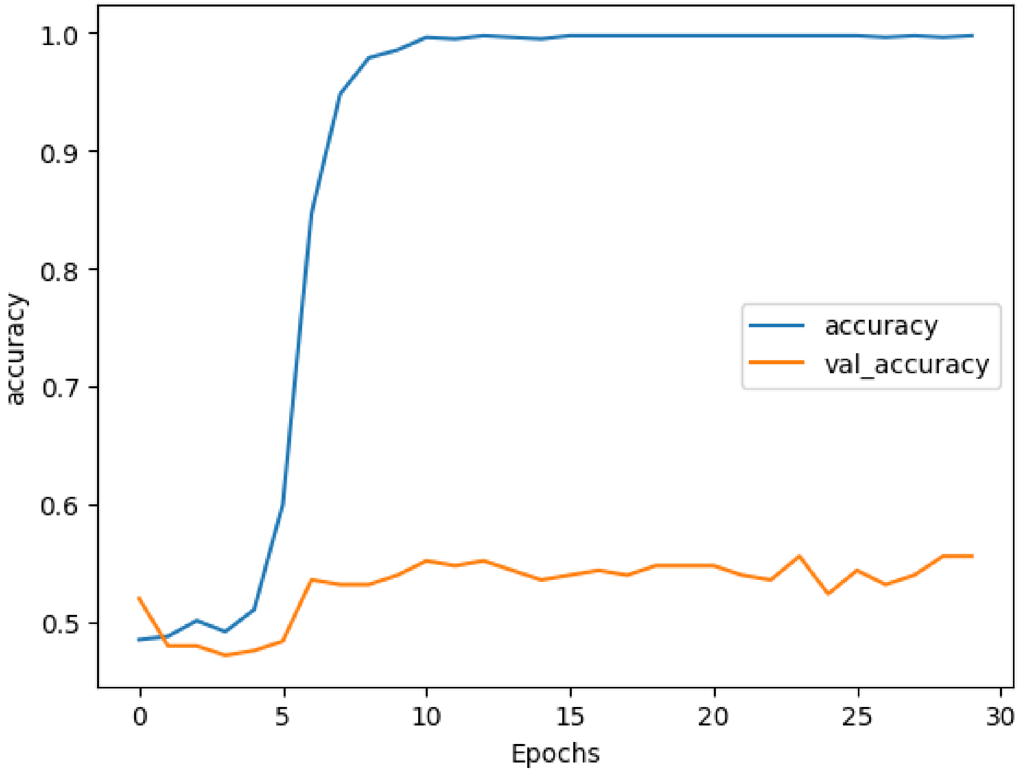
```
num_epochs = 30
history_gru_1 = model_gru_1.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

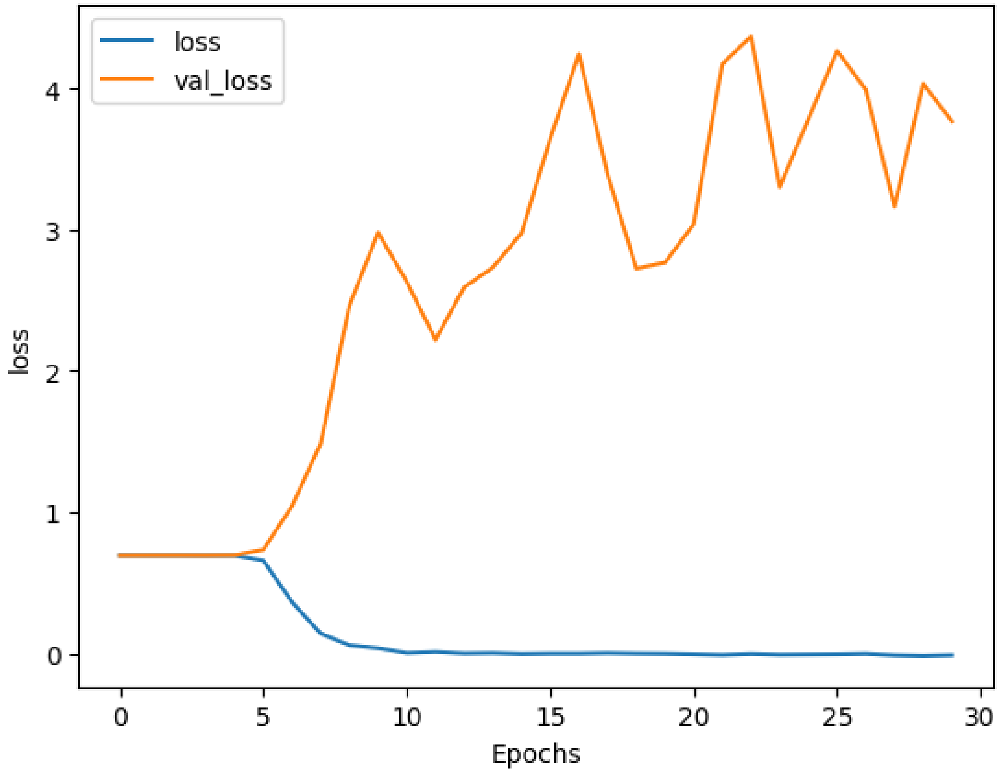
```
Epoch 1/30
24/24 [==============================] - 8s 74ms/step - loss: 0.6944 - accuracy: 0.4853 -
val_loss: 0.6931 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 1s 38ms/step - loss: 0.6938 - accuracy: 0.4880 -
val_loss: 0.6937 - val_accuracy: 0.4800
Epoch 3/30
24/24 [==============================] - 1s 39ms/step - loss: 0.6938 - accuracy: 0.5013 -
val_loss: 0.6939 - val_accuracy: 0.4800
Epoch 4/30
24/24 [==============================] - 1s 38ms/step - loss: 0.6933 - accuracy: 0.4920 -
val_loss: 0.6935 - val_accuracy: 0.4720
Epoch 5/30
24/24 [==============================] - 1s 39ms/step - loss: 0.6933 - accuracy: 0.5107 -
val_loss: 0.6954 - val_accuracy: 0.4760
Epoch 6/30
24/24 [==============================] - 1s 38ms/step - loss: 0.6575 - accuracy: 0.6000 -
val_loss: 0.7351 - val_accuracy: 0.4840
Epoch 7/30
24/24 [==============================] - 1s 39ms/step - loss: 0.3630 - accuracy: 0.8467 -
val_loss: 1.0437 - val_accuracy: 0.5360
Epoch 8/30
24/24 [==============================] - 1s 40ms/step - loss: 0.1402 - accuracy: 0.9480 -
val_loss: 1.4873 - val_accuracy: 0.5320
Epoch 9/30
24/24 [==============================] - 1s 40ms/step - loss: 0.0583 - accuracy: 0.9787 -
val_loss: 2.4639 - val_accuracy: 0.5320
Epoch 10/30
24/24 [==============================] - 1s 39ms/step - loss: 0.0375 - accuracy: 0.9853 -
val_loss: 2.9790 - val_accuracy: 0.5400
Epoch 11/30
24/24 [==============================] - 1s 41ms/step - loss: 0.0040 - accuracy: 0.9960 -
val_loss: 2.6308 - val_accuracy: 0.5520
Epoch 12/30
24/24 [==============================] - 1s 39ms/step - loss: 0.0114 - accuracy: 0.9947 -
val_loss: 2.2210 - val_accuracy: 0.5480
Epoch 13/30
24/24 [==============================] - 1s 38ms/step - loss: 9.2127e-04 - accuracy: 0.99
73 - val_loss: 2.5928 - val_accuracy: 0.5520
Epoch 14/30
24/24 [==============================] - 1s 41ms/step - loss: 0.0033 - accuracy: 0.9960 -
val_loss: 2.7329 - val_accuracy: 0.5440
Epoch 15/30
24/24 [==============================] - 1s 39ms/step - loss: -0.0028 - accuracy: 0.9947
- val_loss: 2.9751 - val_accuracy: 0.5360
Epoch 16/30
24/24 [==============================] - 1s 38ms/step - loss: -3.0109e-04 - accuracy: 0.9
973 - val_loss: 3.6451 - val_accuracy: 0.5400
Epoch 17/30
24/24 [==============================] - 1s 44ms/step - loss: -2.8124e-04 - accuracy: 0.9
973 - val_loss: 4.2427 - val_accuracy: 0.5440
```

```
973 - val_loss: 4.2427 - val_accuracy: 0.5440
Epoch 18/30
24/24 [==============================] - 1s 41ms/step - loss: 0.0031 - accuracy: 0.9973 -
val_loss: 3.3903 - val_accuracy: 0.5400
Epoch 19/30
24/24 [==============================] - 1s 45ms/step - loss: -3.5610e-05 - accuracy: 0.9
973 - val_loss: 2.7244 - val_accuracy: 0.5480
Epoch 20/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0013 - accuracy: 0.9973
- val_loss: 2.7667 - val_accuracy: 0.5480
Epoch 21/30
24/24 [==============================] - 1s 39ms/step - loss: -0.0052 - accuracy: 0.9973
- val_loss: 3.0387 - val_accuracy: 0.5480
Epoch 22/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0096 - accuracy: 0.9973
- val_loss: 4.1727 - val_accuracy: 0.5400
Epoch 23/30
24/24 [==============================] - 1s 38ms/step - loss: -0.0027 - accuracy: 0.9973
- val_loss: 4.3690 - val_accuracy: 0.5360
Epoch 24/30
24/24 [==============================] - 1s 39ms/step - loss: -0.0080 - accuracy: 0.9973
- val_loss: 3.3023 - val_accuracy: 0.5560
Epoch 25/30
24/24 [==============================] - 1s 41ms/step - loss: -0.0065 - accuracy: 0.9973
- val_loss: 3.7838 - val_accuracy: 0.5240
Epoch 26/30
24/24 [==============================] - 1s 38ms/step - loss: -0.0050 - accuracy: 0.9973
- val_loss: 4.2662 - val_accuracy: 0.5440
Epoch 27/30
24/24 [==============================] - 1s 38ms/step - loss: -0.0016 - accuracy: 0.9960
- val_loss: 3.9909 - val_accuracy: 0.5320
Epoch 28/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0123 - accuracy: 0.9973
- val_loss: 3.1631 - val_accuracy: 0.5400
Epoch 29/30
24/24 [==============================] - 1s 38ms/step - loss: -0.0165 - accuracy: 0.9960
- val_loss: 4.0333 - val_accuracy: 0.5560
Epoch 30/30
24/24 [==============================] - 1s 39ms/step - loss: -0.0114 - accuracy: 0.9973
- val_loss: 3.7673 - val_accuracy: 0.5560
```

In [120]:

```
plot_graphs(history_gru_1, "accuracy")
plot_graphs(history_gru_1, "loss")
```

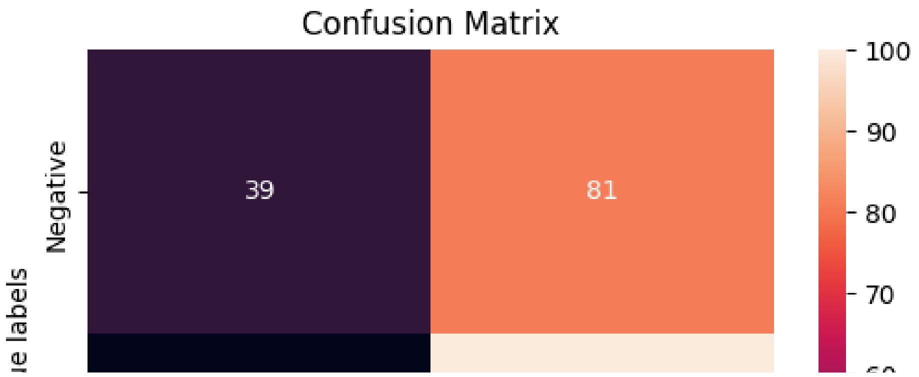


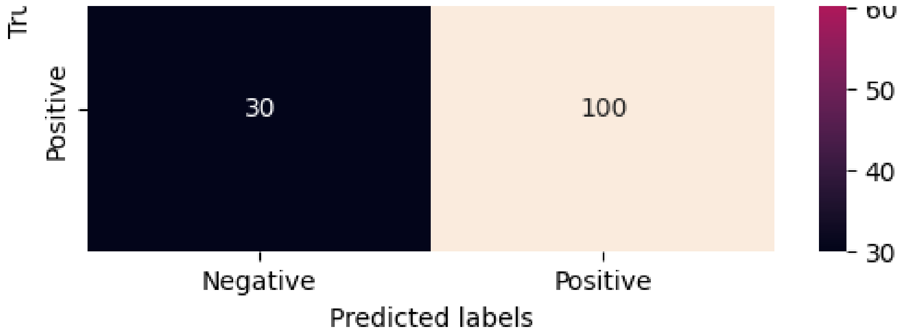


In [121]:

```
# Predict labels for testing data
predicted_labels = (model_gru_1.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 1s 17ms/step
Accuracy: 0.556
Precision: 0.5524861878453039
Recall: 0.7692307692307693
F1-score: 0.6430868167202572
```





GRU (2 Layers and 0.7 Dropout)

In [122]:

```
model_gru_2 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.GRU(units=64, return_sequences=True),  # First layer
    tf.keras.layers.GRU(units=128),  # Second layer
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_gru_2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_gru_2.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_15 (Embedding)    (None, 30, 16)            65376     
                                                                 
 gru_6 (GRU)                 (None, 30, 64)            15744     
                                                                 
 gru_7 (GRU)                 (None, 128)               74496     
                                                                 
 dropout_13 (Dropout)        (None, 128)               0         
                                                                 
 dense_15 (Dense)            (None, 1)                 129       
                                                                 
=================================================================
Total params: 155,745
Trainable params: 155,745
Non-trainable params: 0
_________________________________________________________________
```

In [123]:

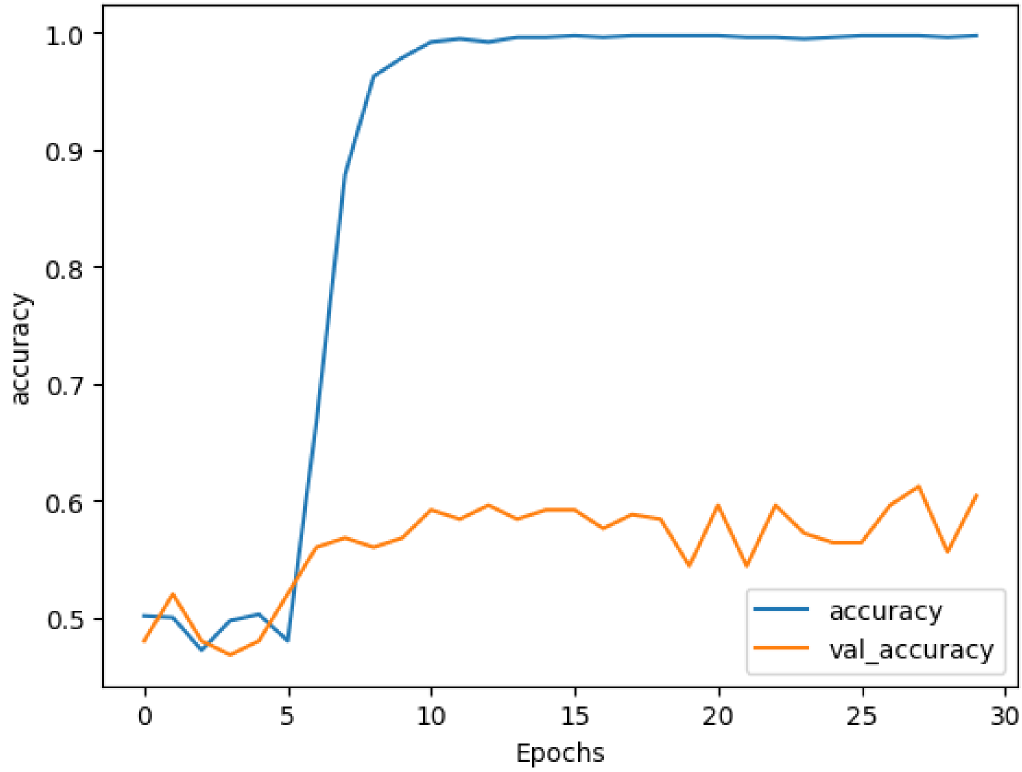
```
num_epochs = 30
history_gru_2 = model_gru_2.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

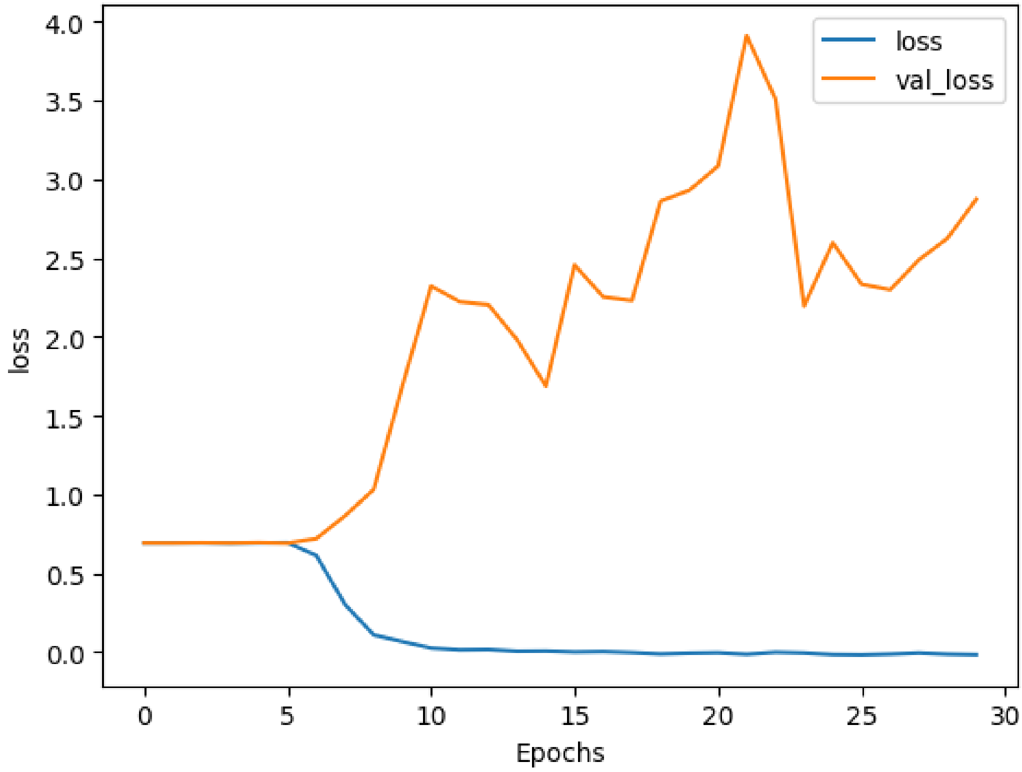
```
Epoch 1/30
24/24 [==============================] - 8s 79ms/step - loss: 0.6934 - accuracy: 0.5013 -
val_loss: 0.6939 - val_accuracy: 0.4800
Epoch 2/30
24/24 [==============================] - 1s 46ms/step - loss: 0.6941 - accuracy: 0.5000 -
val_loss: 0.6932 - val_accuracy: 0.5200
Epoch 3/30
24/24 [==============================] - 1s 39ms/step - loss: 0.6943 - accuracy: 0.4720 -
val_loss: 0.6951 - val_accuracy: 0.4800
Epoch 4/30
24/24 [==============================] - 1s 38ms/step - loss: 0.6924 - accuracy: 0.4973 -
val_loss: 0.6934 - val_accuracy: 0.4680
Epoch 5/30
24/24 [==============================] - 1s 42ms/step - loss: 0.6942 - accuracy: 0.5027 -
val_loss: 0.6960 - val_accuracy: 0.4800
Epoch 6/30
24/24 [==============================] - 1s 38ms/step - loss: 0.6955 - accuracy: 0.4800 -
val_loss: 0.6923 - val_accuracy: 0.5200
```

```
Epoch 7/30
24/24 [==============================] - 1s 38ms/step - loss: 0.6135 - accuracy: 0.6667 -
val_loss: 0.7205 - val_accuracy: 0.5600
Epoch 8/30
24/24 [==============================] - 1s 39ms/step - loss: 0.3032 - accuracy: 0.8787 -
val_loss: 0.8652 - val_accuracy: 0.5680
Epoch 9/30
24/24 [==============================] - 1s 40ms/step - loss: 0.1097 - accuracy: 0.9627 -
val_loss: 1.0322 - val_accuracy: 0.5600
Epoch 10/30
24/24 [==============================] - 1s 38ms/step - loss: 0.0677 - accuracy: 0.9787 -
val_loss: 1.6892 - val_accuracy: 0.5680
Epoch 11/30
24/24 [==============================] - 1s 40ms/step - loss: 0.0267 - accuracy: 0.9920 -
val_loss: 2.3223 - val_accuracy: 0.5920
Epoch 12/30
24/24 [==============================] - 1s 39ms/step - loss: 0.0153 - accuracy: 0.9947 -
val_loss: 2.2229 - val_accuracy: 0.5840
Epoch 13/30
24/24 [==============================] - 1s 39ms/step - loss: 0.0172 - accuracy: 0.9920 -
val_loss: 2.2037 - val_accuracy: 0.5960
Epoch 14/30
24/24 [==============================] - 1s 40ms/step - loss: 0.0060 - accuracy: 0.9960 -
val_loss: 1.9809 - val_accuracy: 0.5840
Epoch 15/30
24/24 [==============================] - 1s 40ms/step - loss: 0.0072 - accuracy: 0.9960 -
val_loss: 1.6882 - val_accuracy: 0.5920
Epoch 16/30
24/24 [==============================] - 1s 41ms/step - loss: 0.0016 - accuracy: 0.9973 -
val_loss: 2.4566 - val_accuracy: 0.5920
Epoch 17/30
24/24 [==============================] - 1s 43ms/step - loss: 0.0040 - accuracy: 0.9960 -
val_loss: 2.2536 - val_accuracy: 0.5760
Epoch 18/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0019 - accuracy: 0.9973
- val_loss: 2.2300 - val_accuracy: 0.5880
Epoch 19/30
24/24 [==============================] - 1s 41ms/step - loss: -0.0105 - accuracy: 0.9973
- val_loss: 2.8608 - val_accuracy: 0.5840
Epoch 20/30
24/24 [==============================] - 1s 38ms/step - loss: -0.0052 - accuracy: 0.9973
- val_loss: 2.9299 - val_accuracy: 0.5440
Epoch 21/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0028 - accuracy: 0.9973
- val_loss: 3.0832 - val_accuracy: 0.5960
Epoch 22/30
24/24 [==============================] - 1s 41ms/step - loss: -0.0125 - accuracy: 0.9960
- val_loss: 3.9099 - val_accuracy: 0.5440
Epoch 23/30
24/24 [==============================] - 1s 39ms/step - loss: 1.4269e-04 - accuracy: 0.99
60 - val_loss: 3.5095 - val_accuracy: 0.5960
Epoch 24/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0040 - accuracy: 0.9947
- val_loss: 2.1952 - val_accuracy: 0.5720
Epoch 25/30
24/24 [==============================] - 1s 41ms/step - loss: -0.0147 - accuracy: 0.9960
- val_loss: 2.5979 - val_accuracy: 0.5640
Epoch 26/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0166 - accuracy: 0.9973
- val_loss: 2.3346 - val_accuracy: 0.5640
Epoch 27/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0116 - accuracy: 0.9973
- val_loss: 2.2990 - val_accuracy: 0.5960
Epoch 28/30
24/24 [==============================] - 1s 42ms/step - loss: -0.0039 - accuracy: 0.9973
- val_loss: 2.4882 - val_accuracy: 0.6120
Epoch 29/30
24/24 [==============================] - 1s 40ms/step - loss: -0.0120 - accuracy: 0.9960
- val_loss: 2.6259 - val_accuracy: 0.5560
Epoch 30/30
24/24 [==============================] - 1s 39ms/step - loss: -0.0156 - accuracy: 0.9973
- val_loss: 2.8728 - val_accuracy: 0.6040
```

In [124]:

```
plot_graphs(history_gru_2, "accuracy")
plot_graphs(history_gru_2, "loss")
```



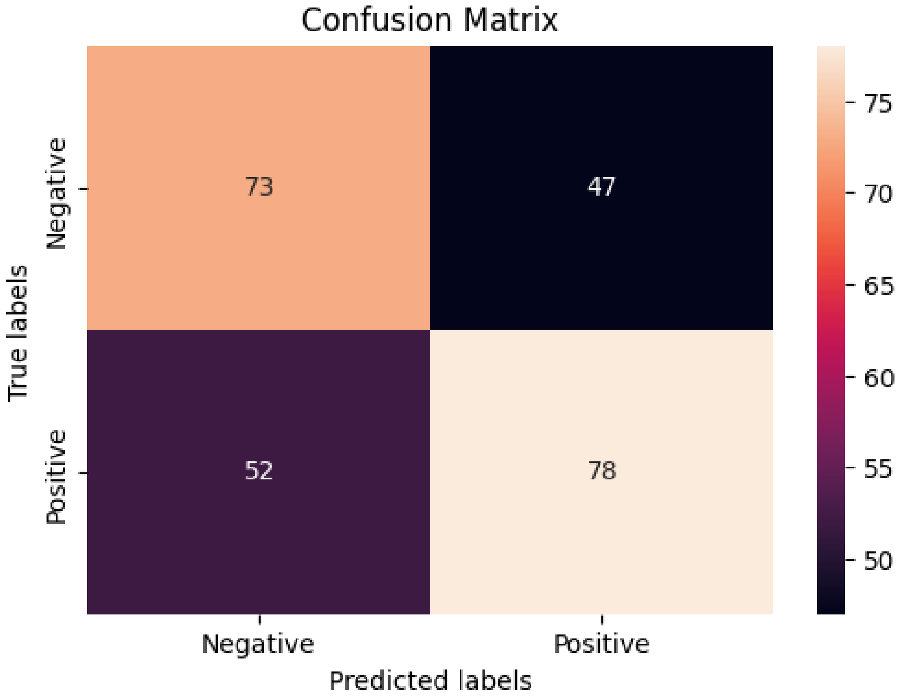


In [125]:

```
# Predict labels for testing data
predicted_labels = (model_gru_2.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
```

```
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 1s 18ms/step
Accuracy: 0.604
Precision: 0.624
Recall: 0.6
F1-score: 0.611764705882353
```



GRU (3 Layers and 0.3 Dropout)

In [127]:

```
model_gru_3 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.GRU(units=64, return_sequences=True),  # First layer
    tf.keras.layers.GRU(units=128,return_sequences=True),  # Second layer
    tf.keras.layers.GRU(units=64),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_gru_3.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_gru_3.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_17 (Embedding)    (None, 30, 16)            65376     
                                                                 
 gru_11 (GRU)                (None, 30, 64)            15744     
                                                                 
 gru_12 (GRU)                (None, 30, 128)           74496     
                                                                 
 gru_13 (GRU)                (None, 64)                37248     
 gru_13 (GRU)                (None, 64)                37248

 dropout_15 (Dropout)        (None, 64)                0

 dense_17 (Dense)            (None, 1)                 65

=================================================================
Total params: 192,929
Trainable params: 192,929
Non-trainable params: 0
_________________________________________________________________
```

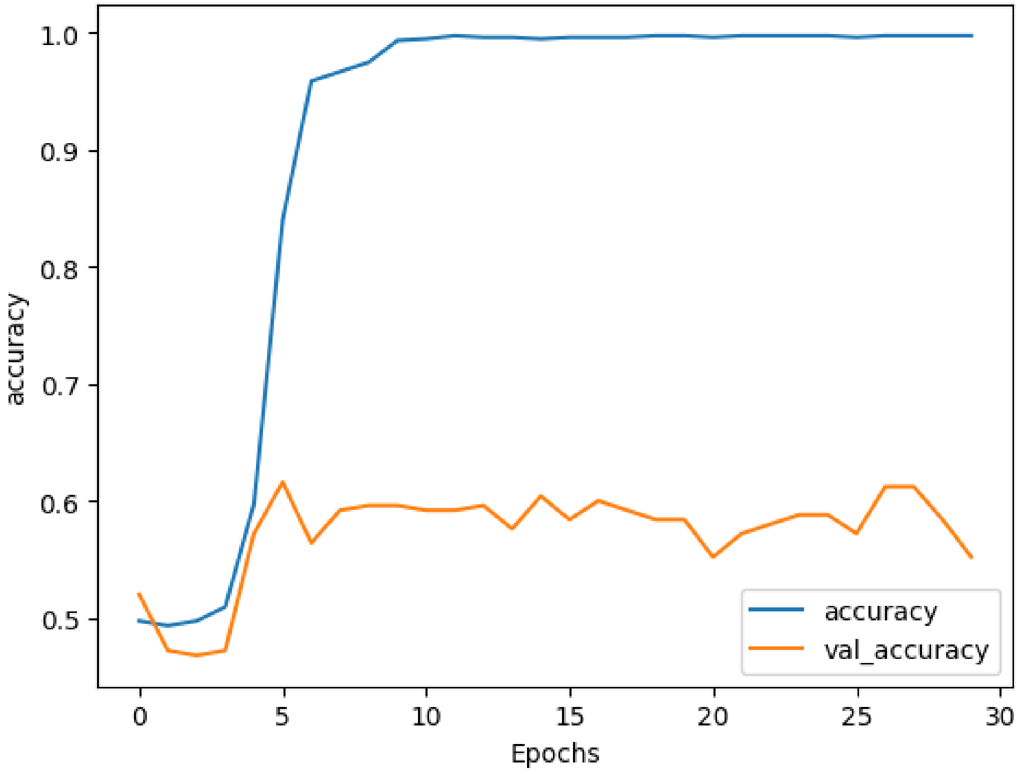
In [128]:

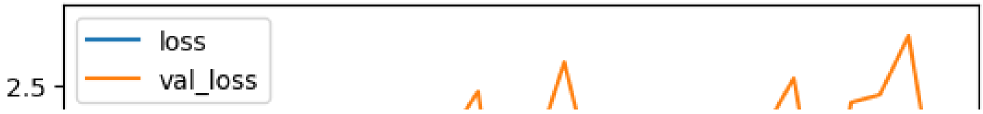
```
num_epochs = 30
history_gru_3 = model_gru_3.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

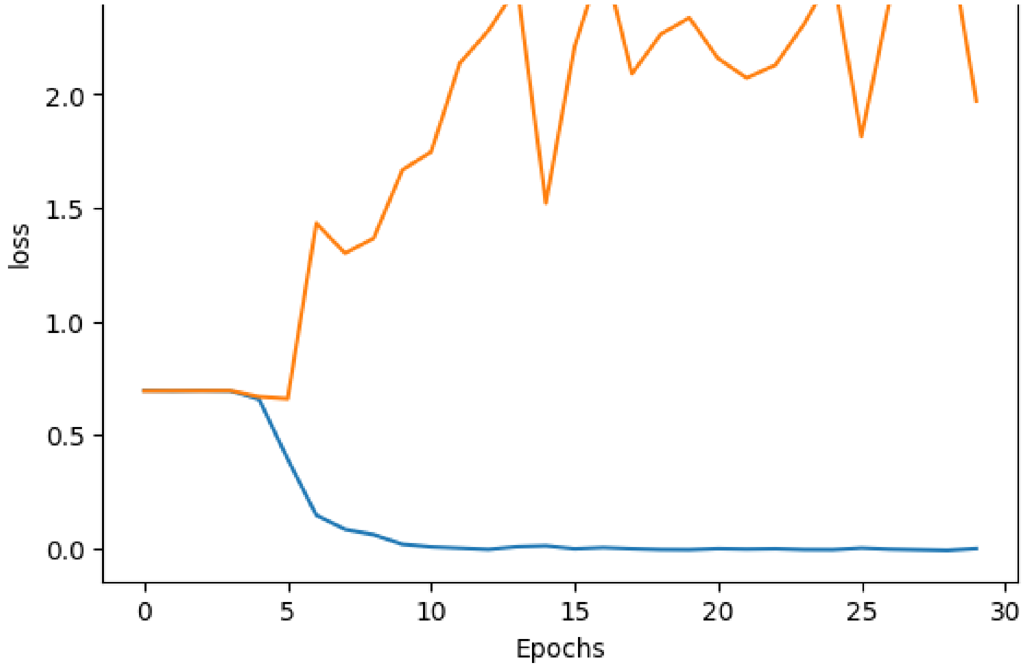
```
Epoch 1/30
24/24 [==============================] - 10s 96ms/step - loss: 0.6944 - accuracy: 0.4973
- val_loss: 0.6927 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 1s 51ms/step - loss: 0.6931 - accuracy: 0.4933 -
val_loss: 0.6933 - val_accuracy: 0.4720
Epoch 3/30
24/24 [==============================] - 1s 49ms/step - loss: 0.6941 - accuracy: 0.4973 -
val_loss: 0.6935 - val_accuracy: 0.4680
Epoch 4/30
24/24 [==============================] - 1s 49ms/step - loss: 0.6927 - accuracy: 0.5093 -
val_loss: 0.6939 - val_accuracy: 0.4720
Epoch 5/30
24/24 [==============================] - 1s 50ms/step - loss: 0.6565 - accuracy: 0.5973 -
val_loss: 0.6668 - val_accuracy: 0.5720
Epoch 6/30
24/24 [==============================] - 1s 50ms/step - loss: 0.3922 - accuracy: 0.8400 -
val_loss: 0.6584 - val_accuracy: 0.6160
Epoch 7/30
24/24 [==============================] - 1s 51ms/step - loss: 0.1447 - accuracy: 0.9587 -
val_loss: 1.4315 - val_accuracy: 0.5640
Epoch 8/30
24/24 [==============================] - 1s 48ms/step - loss: 0.0819 - accuracy: 0.9667 -
val_loss: 1.2997 - val_accuracy: 0.5920
Epoch 9/30
24/24 [==============================] - 1s 49ms/step - loss: 0.0587 - accuracy: 0.9747 -
val_loss: 1.3656 - val_accuracy: 0.5960
Epoch 10/30
24/24 [==============================] - 1s 51ms/step - loss: 0.0165 - accuracy: 0.9933 -
val_loss: 1.6671 - val_accuracy: 0.5960
Epoch 11/30
24/24 [==============================] - 1s 51ms/step - loss: 0.0045 - accuracy: 0.9947 -
val_loss: 1.7460 - val_accuracy: 0.5920
Epoch 12/30
24/24 [==============================] - 1s 49ms/step - loss: -7.1738e-04 - accuracy: 0.9
973 - val_loss: 2.1375 - val_accuracy: 0.5920
Epoch 13/30
24/24 [==============================] - 1s 55ms/step - loss: -0.0062 - accuracy: 0.9960
- val_loss: 2.2812 - val_accuracy: 0.5960
Epoch 14/30
24/24 [==============================] - 1s 58ms/step - loss: 0.0055 - accuracy: 0.9960 -
val_loss: 2.4740 - val_accuracy: 0.5760
Epoch 15/30
24/24 [==============================] - 1s 52ms/step - loss: 0.0092 - accuracy: 0.9947 -
val_loss: 1.5206 - val_accuracy: 0.6040
Epoch 16/30
24/24 [==============================] - 1s 56ms/step - loss: -0.0036 - accuracy: 0.9960
- val_loss: 2.2097 - val_accuracy: 0.5840
Epoch 17/30
24/24 [==============================] - 1s 60ms/step - loss: 0.0016 - accuracy: 0.9960 -
val_loss: 2.6029 - val_accuracy: 0.6000
Epoch 18/30
24/24 [==============================] - 2s 64ms/step - loss: -0.0033 - accuracy: 0.9960
- val_loss: 2.0909 - val_accuracy: 0.5920
Epoch 19/30
Epoch 19/30
24/24 [==============================] - 2s 64ms/step - loss: -0.0068 - accuracy: 0.9973
- val_loss: 2.2646 - val_accuracy: 0.5840
Epoch 20/30
24/24 [==============================] - 1s 62ms/step - loss: -0.0075 - accuracy: 0.9973
- val_loss: 2.3376 - val_accuracy: 0.5840
Epoch 21/30
24/24 [==============================] - 1s 59ms/step - loss: -0.0027 - accuracy: 0.9960
- val_loss: 2.1584 - val_accuracy: 0.5520
Epoch 22/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0049 - accuracy: 0.9973
- val_loss: 2.0729 - val_accuracy: 0.5720
Epoch 23/30
24/24 [==============================] - 1s 52ms/step - loss: -0.0032 - accuracy: 0.9973
- val_loss: 2.1296 - val_accuracy: 0.5800
Epoch 24/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0070 - accuracy: 0.9973
- val_loss: 2.3101 - val_accuracy: 0.5880
Epoch 25/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0074 - accuracy: 0.9973
- val_loss: 2.5323 - val_accuracy: 0.5880
Epoch 26/30
24/24 [==============================] - 1s 50ms/step - loss: -1.9348e-04 - accuracy: 0.9
960 - val_loss: 1.8152 - val_accuracy: 0.5720
Epoch 27/30
24/24 [==============================] - 1s 49ms/step - loss: -0.0053 - accuracy: 0.9973
- val_loss: 2.4249 - val_accuracy: 0.6120
Epoch 28/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0079 - accuracy: 0.9973
- val_loss: 2.4589 - val_accuracy: 0.6120
Epoch 29/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0105 - accuracy: 0.9973
- val_loss: 2.7192 - val_accuracy: 0.5840
Epoch 30/30
24/24 [==============================] - 1s 50ms/step - loss: -0.0022 - accuracy: 0.9973
- val_loss: 1.9717 - val_accuracy: 0.5520
```

In [130]:

```
plot_graphs(history_gru_3, "accuracy")
plot_graphs(history_gru_3, "loss")
```



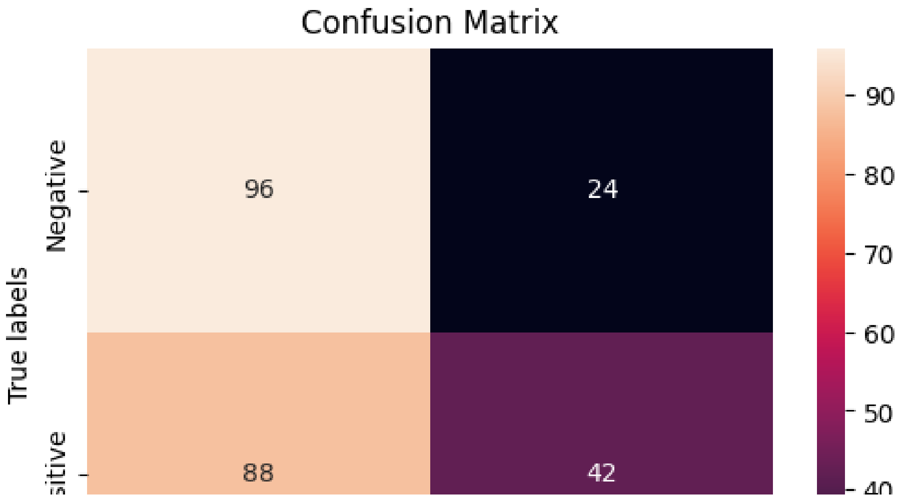




In [132]:

```
# Predict labels for testing data
predicted_labels = (model_gru_3.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 2s 19ms/step
Accuracy: 0.552
Precision: 0.6363636363636364
Recall: 0.3230769230769231
F1-score: 0.4285714285714286
```



GRU (3 Layers and 0.7 Dropout)

In [133]:

```
model_gru_4 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.GRU(units=64, return_sequences=True),   # First layer
    tf.keras.layers.GRU(units=128,return_sequences=True),   # Second layer
    tf.keras.layers.GRU(units=64),
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_gru_4.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_gru_4.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_18 (Embedding)    (None, 30, 16)            65376     
                                                                 
 gru_14 (GRU)                (None, 30, 64)            15744     
                                                                 
 gru_15 (GRU)                (None, 30, 128)           74496     
                                                                 
 gru_16 (GRU)                (None, 64)                37248     
                                                                 
 dropout_16 (Dropout)        (None, 64)                0         
                                                                 
 dense_18 (Dense)            (None, 1)                 65        
                                                                 
=================================================================
Total params: 192,929
Trainable params: 192,929
Non-trainable params: 0
_________________________________________________________________
```

In [134]:

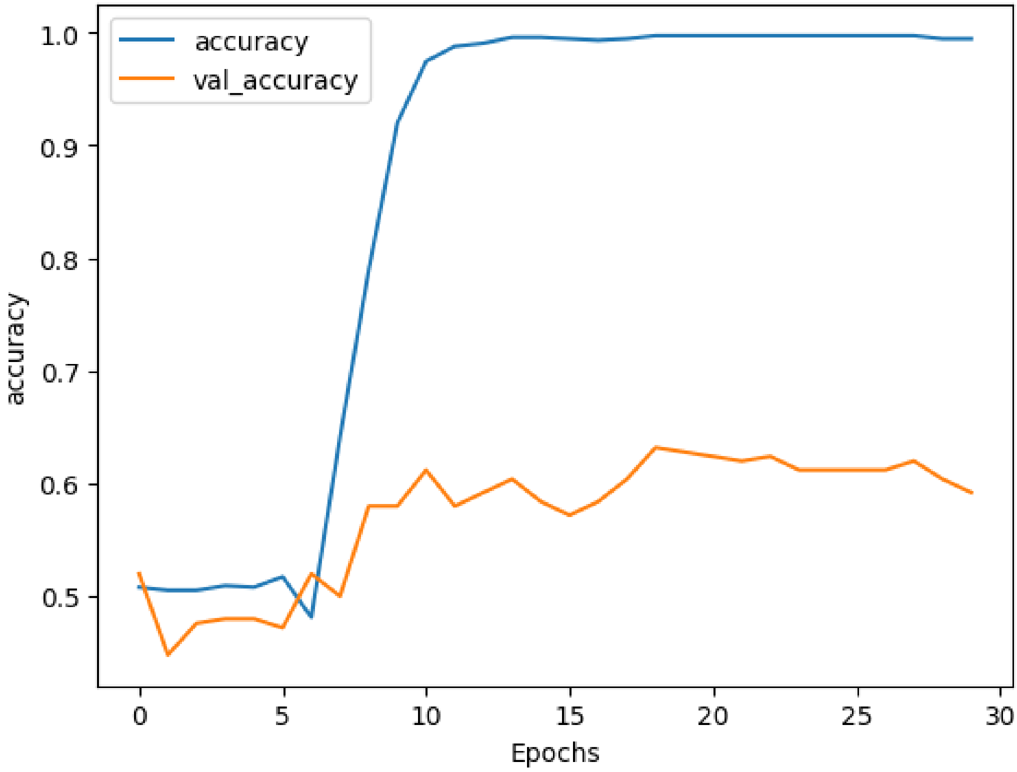
```
num_epochs = 30
history_gru_4 = model_gru_4.fit(training_padded, training_labels_final, epochs=num_epochs
, validation_data=(testing_padded, testing_labels_final))
```

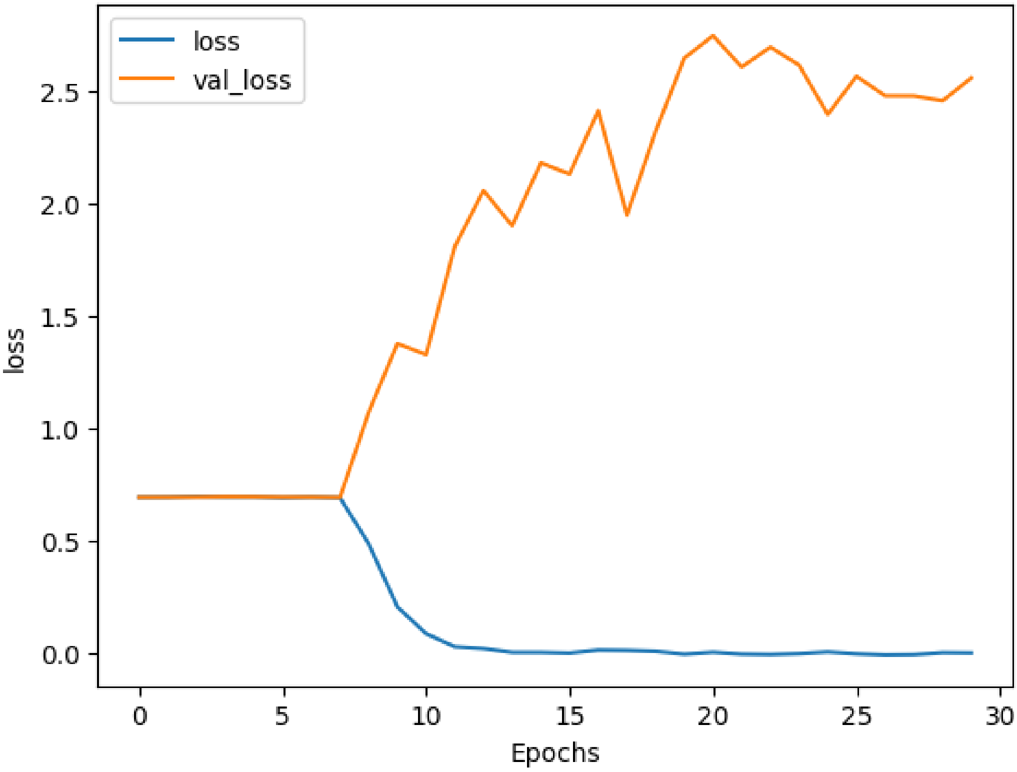
```
Epoch 1/30
24/24 [==============================] - 11s 103ms/step - loss: 0.6930 - accuracy: 0.5080
- val_loss: 0.6930 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 1s 51ms/step - loss: 0.6934 - accuracy: 0.5053 -
val_loss: 0.6933 - val_accuracy: 0.4480
Epoch 3/30
24/24 [==============================] - 1s 54ms/step - loss: 0.6962 - accuracy: 0.5053 -
val_loss: 0.6939 - val_accuracy: 0.4760
Epoch 4/30
24/24 [==============================] - 1s 54ms/step - loss: 0.6943 - accuracy: 0.5093 -
val_loss: 0.6954 - val_accuracy: 0.4800
Epoch 5/30
24/24 [==============================] - 1s 51ms/step - loss: 0.6945 - accuracy: 0.5080 -
val_loss: 0.6949 - val_accuracy: 0.4800
Epoch 6/30
24/24 [==============================] - 1s 51ms/step - loss: 0.6915 - accuracy: 0.5173 -
val_loss: 0.6937 - val_accuracy: 0.4720
Epoch 7/30
```

```
24/24 [==============================] - 1s 56ms/step - loss: 0.6936 - accuracy: 0.4813 -
val_loss: 0.6939 - val_accuracy: 0.5200
Epoch 8/30
24/24 [==============================] - 1s 51ms/step - loss: 0.6914 - accuracy: 0.6413 -
val_loss: 0.6928 - val_accuracy: 0.5000
Epoch 9/30
24/24 [==============================] - 1s 53ms/step - loss: 0.4863 - accuracy: 0.7893 -
val_loss: 1.0716 - val_accuracy: 0.5800
Epoch 10/30
24/24 [==============================] - 1s 54ms/step - loss: 0.2063 - accuracy: 0.9200 -
val_loss: 1.3768 - val_accuracy: 0.5800
Epoch 11/30
24/24 [==============================] - 1s 52ms/step - loss: 0.0869 - accuracy: 0.9747 -
val_loss: 1.3273 - val_accuracy: 0.6120
Epoch 12/30
24/24 [==============================] - 1s 56ms/step - loss: 0.0290 - accuracy: 0.9880 -
val_loss: 1.8080 - val_accuracy: 0.5800
Epoch 13/30
24/24 [==============================] - 1s 52ms/step - loss: 0.0211 - accuracy: 0.9907 -
val_loss: 2.0569 - val_accuracy: 0.5920
Epoch 14/30
24/24 [==============================] - 1s 54ms/step - loss: 0.0040 - accuracy: 0.9960 -
val_loss: 1.9009 - val_accuracy: 0.6040
Epoch 15/30
24/24 [==============================] - 1s 56ms/step - loss: 0.0039 - accuracy: 0.9960 -
val_loss: 2.1798 - val_accuracy: 0.5840
Epoch 16/30
24/24 [==============================] - 1s 52ms/step - loss: 0.0011 - accuracy: 0.9947 -
val_loss: 2.1306 - val_accuracy: 0.5720
Epoch 17/30
24/24 [==============================] - 1s 55ms/step - loss: 0.0143 - accuracy: 0.9933 -
val_loss: 2.4119 - val_accuracy: 0.5840
Epoch 18/30
24/24 [==============================] - 1s 55ms/step - loss: 0.0128 - accuracy: 0.9947 -
val_loss: 1.9478 - val_accuracy: 0.6040
Epoch 19/30
24/24 [==============================] - 1s 55ms/step - loss: 0.0088 - accuracy: 0.9973 -
val_loss: 2.3202 - val_accuracy: 0.6320
Epoch 20/30
24/24 [==============================] - 1s 54ms/step - loss: -0.0037 - accuracy: 0.9973
- val_loss: 2.6459 - val_accuracy: 0.6280
Epoch 21/30
24/24 [==============================] - 1s 51ms/step - loss: 0.0047 - accuracy: 0.9973 -
val_loss: 2.7451 - val_accuracy: 0.6240
Epoch 22/30
24/24 [==============================] - 1s 57ms/step - loss: -0.0034 - accuracy: 0.9973
- val_loss: 2.6063 - val_accuracy: 0.6200
Epoch 23/30
24/24 [==============================] - 1s 57ms/step - loss: -0.0052 - accuracy: 0.9973
- val_loss: 2.6944 - val_accuracy: 0.6240
Epoch 24/30
24/24 [==============================] - 1s 57ms/step - loss: -0.0014 - accuracy: 0.9973
- val_loss: 2.6145 - val_accuracy: 0.6120
Epoch 25/30
24/24 [==============================] - 1s 53ms/step - loss: 0.0061 - accuracy: 0.9973 -
val_loss: 2.3951 - val_accuracy: 0.6120
Epoch 26/30
24/24 [==============================] - 1s 56ms/step - loss: -0.0022 - accuracy: 0.9973
- val_loss: 2.5652 - val_accuracy: 0.6120
Epoch 27/30
24/24 [==============================] - 1s 54ms/step - loss: -0.0071 - accuracy: 0.9973
- val_loss: 2.4781 - val_accuracy: 0.6120
Epoch 28/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0059 - accuracy: 0.9973
- val_loss: 2.4776 - val_accuracy: 0.6200
Epoch 29/30
24/24 [==============================] - 1s 56ms/step - loss: 0.0030 - accuracy: 0.9947 -
val_loss: 2.4559 - val_accuracy: 0.6040
Epoch 30/30
24/24 [==============================] - 1s 57ms/step - loss: 0.0019 - accuracy: 0.9947 -
val_loss: 2.5575 - val_accuracy: 0.5920
```

In [135]:

```
plot_graphs(history_gru_4, "accuracy")
plot_graphs(history_gru_4, "loss")
```

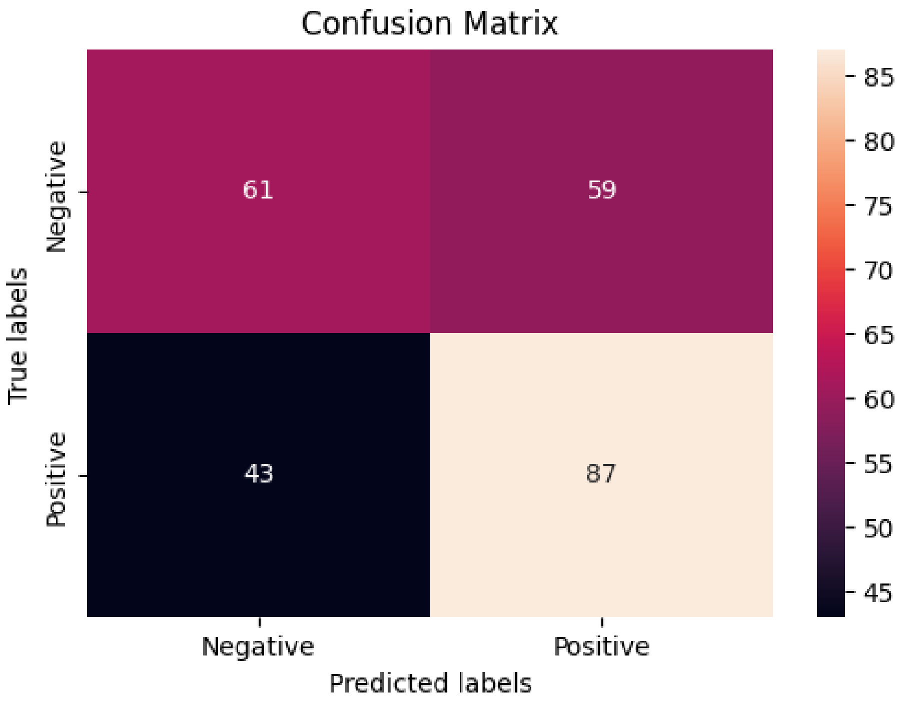




In [136]:

```
# Predict labels for testing data
predicted_labels = (model_gru_4.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 2s 24ms/step
Accuracy: 0.592
Precision: 0.5958904109589042
Recall: 0.6692307692307692
F1-score: 0.6304347826086957
```



LSTM (2 Layers and 0.3 Dropout)

In [139]:

```
model_lstm_1 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.LSTM(units=64, return_sequences=True),   # First layer
    tf.keras.layers.LSTM(units=128),   # Second layer
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_lstm_1.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_lstm_1.summary()
```

```
Model: "sequential_20"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_20 (Embedding)    (None, 30, 16)            65376     
                                                                 
 lstm_2 (LSTM)               (None, 30, 64)            20736     
                                                                 
 lstm_3 (LSTM)               (None, 128)               98816     
                                                                 
 dropout_18 (Dropout)        (None, 128)               0         
                                                                 
 dense_20 (Dense)            (None, 1)                 129       
 dense_20 (Dense)            (None, 1)                 129

=================================================================
Total params: 185,057
Trainable params: 185,057
Non-trainable params: 0
_________________________________________________________________
```

In [140]:

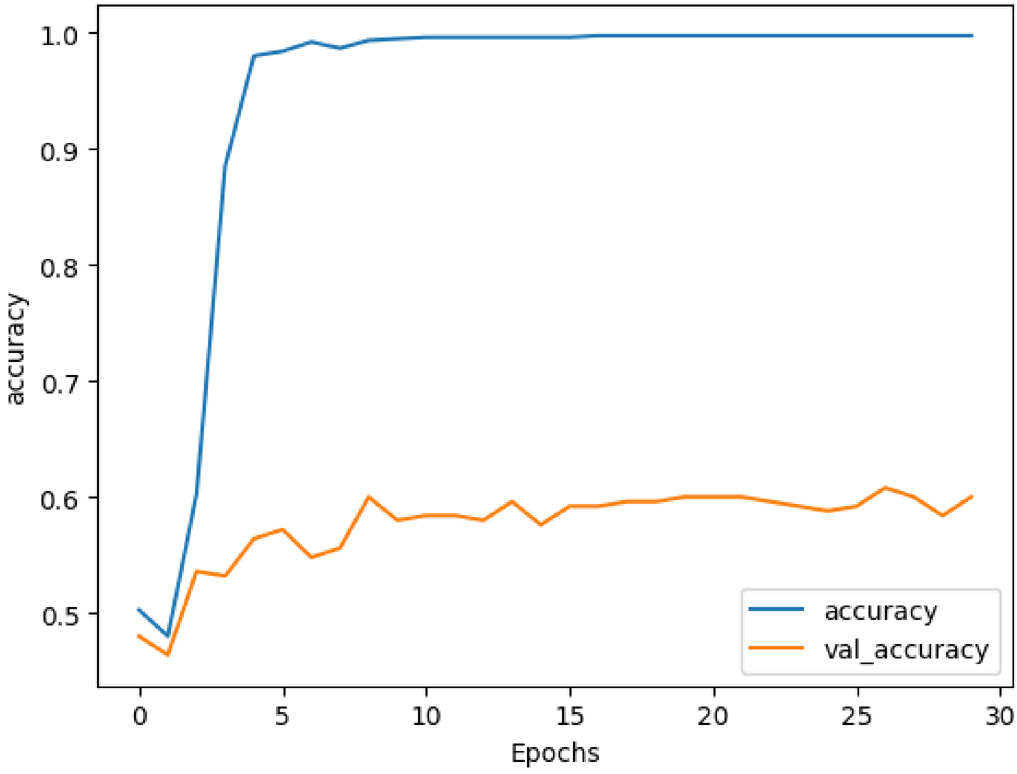
```
num_epochs = 30
history_lstm_1 = model_lstm_1.fit(training_padded, training_labels_final, epochs=num_epoc
hs, validation_data=(testing_padded, testing_labels_final))
```

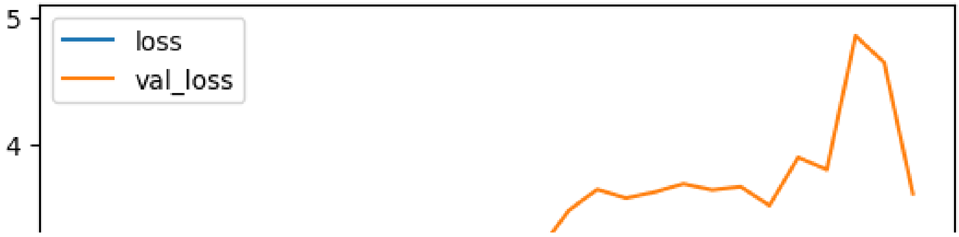
```
Epoch 1/30
24/24 [==============================] - 8s 94ms/step - loss: 0.6936 - accuracy: 0.5027 -
val_loss: 0.6945 - val_accuracy: 0.4800
Epoch 2/30
24/24 [==============================] - 1s 51ms/step - loss: 0.6949 - accuracy: 0.4800 -
val_loss: 0.6933 - val_accuracy: 0.4640
Epoch 3/30
24/24 [==============================] - 1s 47ms/step - loss: 0.6688 - accuracy: 0.6027 -
val_loss: 0.9084 - val_accuracy: 0.5360
Epoch 4/30
24/24 [==============================] - 1s 48ms/step - loss: 0.2945 - accuracy: 0.8853 -
val_loss: 0.8814 - val_accuracy: 0.5320
Epoch 5/30
24/24 [==============================] - 1s 57ms/step - loss: 0.0768 - accuracy: 0.9800 -
val_loss: 2.1008 - val_accuracy: 0.5640
Epoch 6/30
24/24 [==============================] - 1s 49ms/step - loss: 0.0400 - accuracy: 0.9840 -
val_loss: 1.7204 - val_accuracy: 0.5720
Epoch 7/30
24/24 [==============================] - 1s 47ms/step - loss: 0.0238 - accuracy: 0.9920 -
val_loss: 2.8255 - val_accuracy: 0.5480
Epoch 8/30
24/24 [==============================] - 1s 53ms/step - loss: 0.0293 - accuracy: 0.9867 -
val_loss: 1.8702 - val_accuracy: 0.5560
Epoch 9/30
24/24 [==============================] - 1s 49ms/step - loss: 0.0086 - accuracy: 0.9933 -
val_loss: 2.9821 - val_accuracy: 0.6000
Epoch 10/30
24/24 [==============================] - 1s 46ms/step - loss: 0.0119 - accuracy: 0.9947 -
val_loss: 2.3975 - val_accuracy: 0.5800
Epoch 11/30
24/24 [==============================] - 1s 47ms/step - loss: 0.0021 - accuracy: 0.9960 -
val_loss: 2.7729 - val_accuracy: 0.5840
Epoch 12/30
24/24 [==============================] - 1s 53ms/step - loss: 9.3313e-05 - accuracy: 0.99
60 - val_loss: 3.0206 - val_accuracy: 0.5840
Epoch 13/30
24/24 [==============================] - 1s 51ms/step - loss: 0.0016 - accuracy: 0.9960 -
val_loss: 3.0271 - val_accuracy: 0.5800
Epoch 14/30
24/24 [==============================] - 1s 53ms/step - loss: -0.0028 - accuracy: 0.9960
- val_loss: 3.0237 - val_accuracy: 0.5960
Epoch 15/30
24/24 [==============================] - 1s 52ms/step - loss: -0.0066 - accuracy: 0.9960
- val_loss: 2.4549 - val_accuracy: 0.5760
Epoch 16/30
24/24 [==============================] - 1s 49ms/step - loss: -0.0112 - accuracy: 0.9960
- val_loss: 2.6163 - val_accuracy: 0.5920
Epoch 17/30
24/24 [==============================] - 1s 50ms/step - loss: -0.0095 - accuracy: 0.9973
- val_loss: 3.1692 - val_accuracy: 0.5920
Epoch 18/30
24/24 [==============================] - 1s 45ms/step - loss: -0.0132 - accuracy: 0.9973
- val_loss: 3.4774 - val_accuracy: 0.5960
Epoch 19/30
24/24 [==============================] - 1s 50ms/step - loss: -0.0119 - accuracy: 0.9973
- val_loss: 3.6437 - val_accuracy: 0.5960
Epoch 20/30
24/24 [==============================] - 1s 50ms/step - loss: -0.0144 - accuracy: 0.9973
```

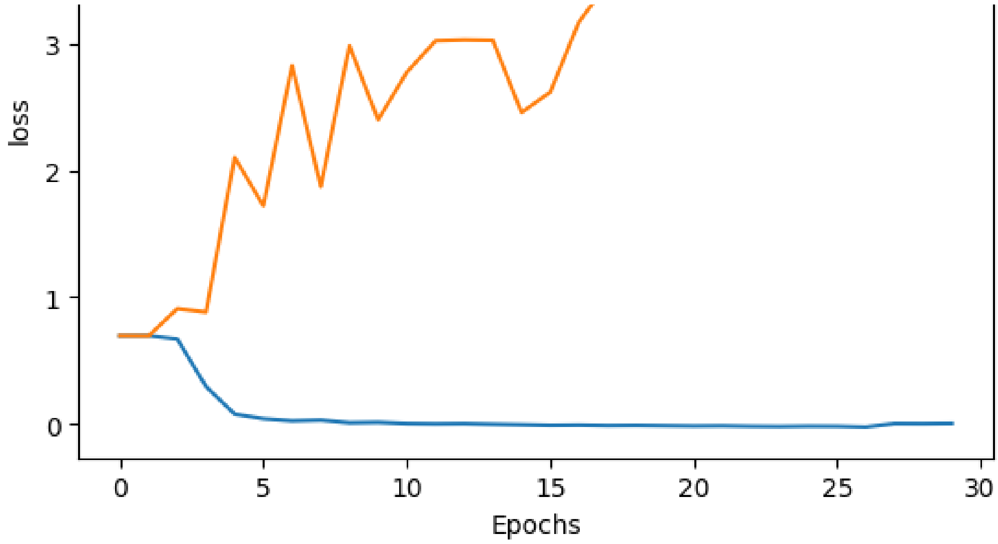
```
24/24 [==============================] - 1s 50ms/step - loss: -0.0144 - accuracy: 0.9973
- val_loss: 3.5750 - val_accuracy: 0.6000
Epoch 21/30
24/24 [==============================] - 1s 48ms/step - loss: -0.0166 - accuracy: 0.9973
- val_loss: 3.6227 - val_accuracy: 0.6000
Epoch 22/30
24/24 [==============================] - 1s 46ms/step - loss: -0.0155 - accuracy: 0.9973
- val_loss: 3.6881 - val_accuracy: 0.6000
Epoch 23/30
24/24 [==============================] - 1s 50ms/step - loss: -0.0188 - accuracy: 0.9973
- val_loss: 3.6392 - val_accuracy: 0.5960
Epoch 24/30
24/24 [==============================] - 1s 46ms/step - loss: -0.0203 - accuracy: 0.9973
- val_loss: 3.6648 - val_accuracy: 0.5920
Epoch 25/30
24/24 [==============================] - 1s 47ms/step - loss: -0.0180 - accuracy: 0.9973
- val_loss: 3.5167 - val_accuracy: 0.5880
Epoch 26/30
24/24 [==============================] - 1s 47ms/step - loss: -0.0189 - accuracy: 0.9973
- val_loss: 3.8954 - val_accuracy: 0.5920
Epoch 27/30
24/24 [==============================] - 1s 45ms/step - loss: -0.0244 - accuracy: 0.9973
- val_loss: 3.8000 - val_accuracy: 0.6080
Epoch 28/30
24/24 [==============================] - 1s 45ms/step - loss: 0.0024 - accuracy: 0.9973 -
val_loss: 4.8547 - val_accuracy: 0.6000
Epoch 29/30
24/24 [==============================] - 1s 47ms/step - loss: 0.0017 - accuracy: 0.9973 -
val_loss: 4.6435 - val_accuracy: 0.5840
Epoch 30/30
24/24 [==============================] - 1s 44ms/step - loss: 0.0038 - accuracy: 0.9973 -
val_loss: 3.6089 - val_accuracy: 0.6000
```

In [142]:

```
plot_graphs(history_lstm_1, "accuracy")
plot_graphs(history_lstm_1, "loss")
```



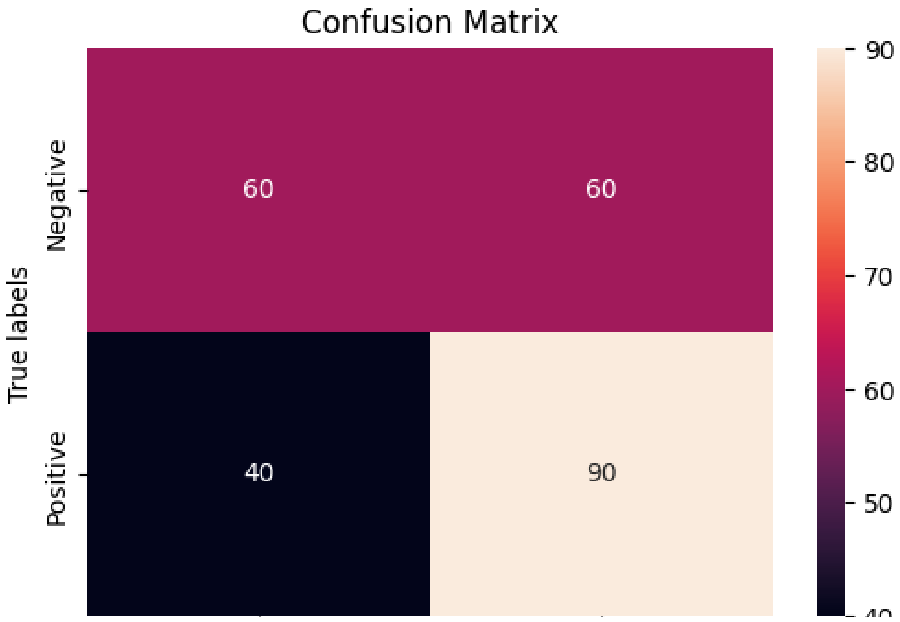




In [143]:

```
# Predict labels for testing data
predicted_labels = (model_lstm_1.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 1s 18ms/step
Accuracy: 0.6
Precision: 0.6
Recall: 0.6923076923076923
F1-score: 0.6428571428571429
```



LSTM (2 Layers and 0.7 Dropout)

In [145]:

```
model_lstm_2 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.LSTM(units=64, return_sequences=True),   # First layer
    tf.keras.layers.LSTM(units=128),   # Second layer
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_lstm_2.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_lstm_2.summary()
```

```
Model: "sequential_22"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_22 (Embedding)    (None, 30, 16)            65376

 lstm_6 (LSTM)               (None, 30, 64)            20736

 lstm_7 (LSTM)               (None, 128)               98816

 dropout_20 (Dropout)        (None, 128)               0

 dense_22 (Dense)            (None, 1)                 129

=================================================================
Total params: 185,057
Trainable params: 185,057
Non-trainable params: 0
_________________________________________________________________
```

In [146]:

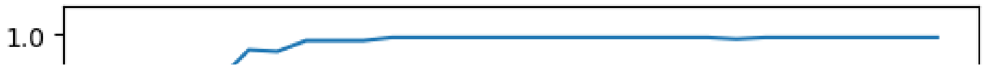
```
num_epochs = 30
history_lstm_2 = model_lstm_2.fit(training_padded, training_labels_final, epochs=num_epoc
hs, validation_data=(testing_padded, testing_labels_final))
```

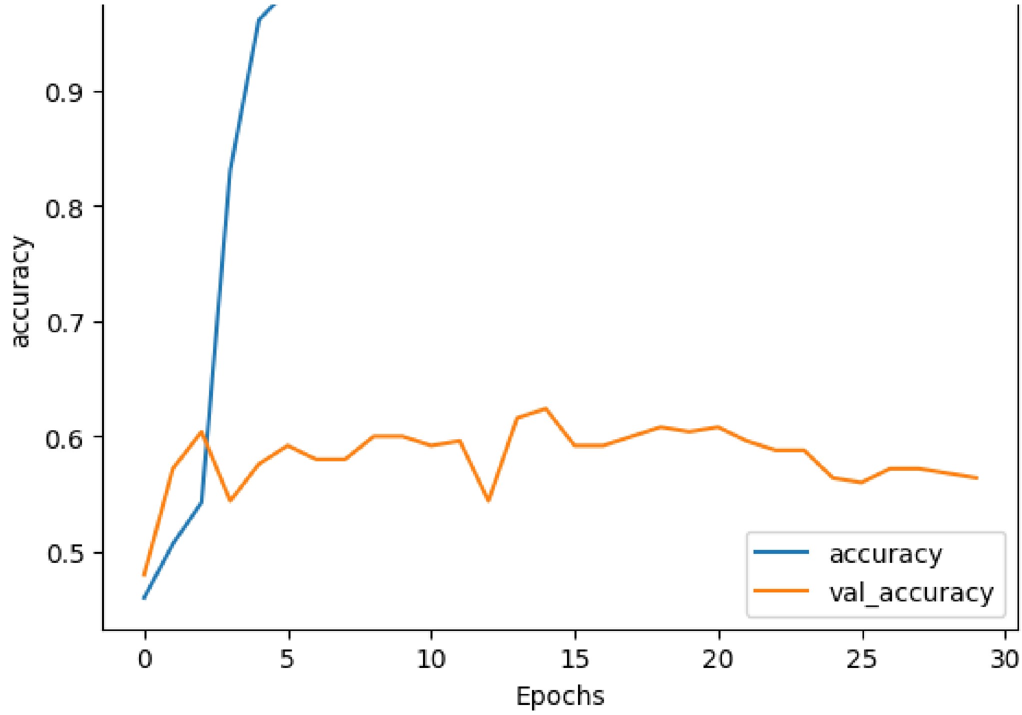
```
Epoch 1/30
24/24 [==============================] - 7s 80ms/step - loss: 0.6944 - accuracy: 0.4600 -
val_loss: 0.6936 - val_accuracy: 0.4800
Epoch 2/30
24/24 [==============================] - 1s 42ms/step - loss: 0.6932 - accuracy: 0.5067 -
val_loss: 0.6929 - val_accuracy: 0.5720
Epoch 3/30
24/24 [==============================] - 1s 41ms/step - loss: 0.6892 - accuracy: 0.5427 -
val_loss: 0.6864 - val_accuracy: 0.6040
Epoch 4/30
24/24 [==============================] - 1s 44ms/step - loss: 0.4426 - accuracy: 0.8307 -
val_loss: 0.8079 - val_accuracy: 0.5440
Epoch 5/30
24/24 [==============================] - 1s 44ms/step - loss: 0.1381 - accuracy: 0.9613 -
val_loss: 1.1067 - val_accuracy: 0.5760
Epoch 6/30
24/24 [==============================] - 1s 43ms/step - loss: 0.0325 - accuracy: 0.9867 -
val_loss: 1.8653 - val_accuracy: 0.5920
Epoch 7/30
24/24 [==============================] - 1s 42ms/step - loss: 0.0389 - accuracy: 0.9853 -
val_loss: 1.9371 - val_accuracy: 0.5800
Epoch 8/30
24/24 [==============================] - 1s 43ms/step - loss: 0.0166 - accuracy: 0.9947 -
val_loss: 1.8829 - val_accuracy: 0.5800
Epoch 9/30
24/24 [==============================] - 1s 44ms/step - loss: 0.0102 - accuracy: 0.9947 -
```

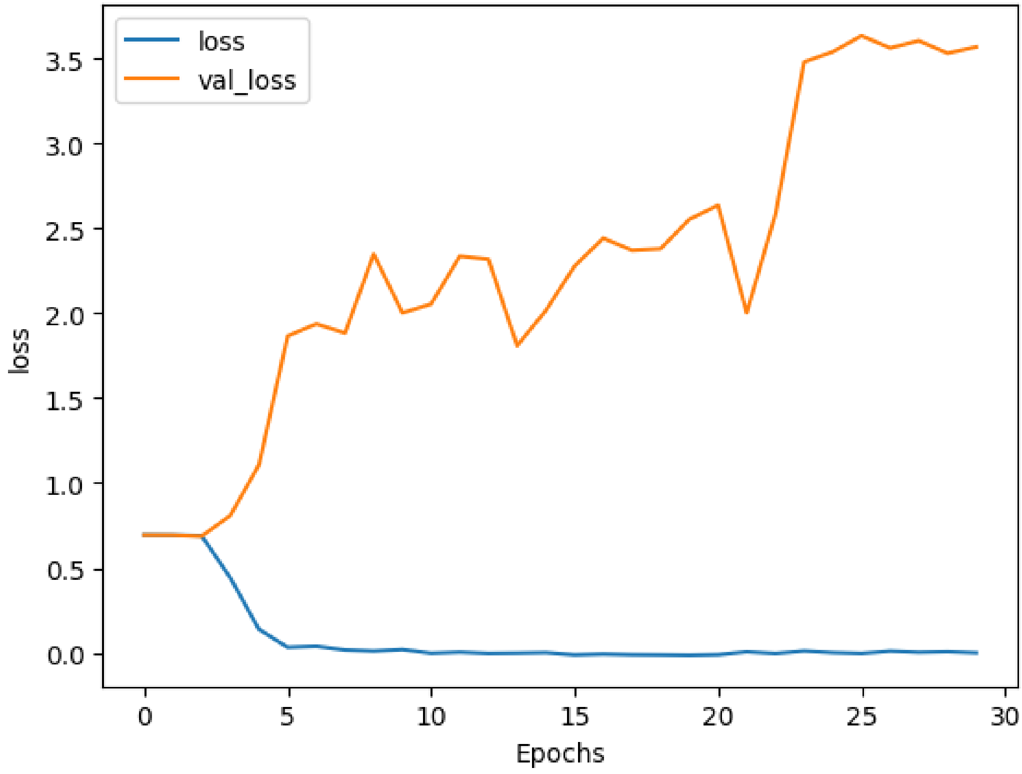
```
val_loss: 2.3494 - val_accuracy: 0.6000
Epoch 10/30
24/24 [==============================] - 1s 43ms/step - loss: 0.0192 - accuracy: 0.9947 -
val_loss: 2.0018 - val_accuracy: 0.6000
Epoch 11/30
24/24 [==============================] - 1s 44ms/step - loss: -0.0026 - accuracy: 0.9973
- val_loss: 2.0540 - val_accuracy: 0.5920
Epoch 12/30
24/24 [==============================] - 1s 46ms/step - loss: 0.0037 - accuracy: 0.9973 -
val_loss: 2.3357 - val_accuracy: 0.5960
Epoch 13/30
24/24 [==============================] - 1s 45ms/step - loss: -0.0037 - accuracy: 0.9973
- val_loss: 2.3173 - val_accuracy: 0.5440
Epoch 14/30
24/24 [==============================] - 1s 45ms/step - loss: -0.0018 - accuracy: 0.9973
- val_loss: 1.8088 - val_accuracy: 0.6160
Epoch 15/30
24/24 [==============================] - 1s 47ms/step - loss: 9.9094e-04 - accuracy: 0.99
73 - val_loss: 2.0163 - val_accuracy: 0.6240
Epoch 16/30
24/24 [==============================] - 1s 46ms/step - loss: -0.0123 - accuracy: 0.9973
- val_loss: 2.2773 - val_accuracy: 0.5920
Epoch 17/30
24/24 [==============================] - 1s 49ms/step - loss: -0.0076 - accuracy: 0.9973
- val_loss: 2.4417 - val_accuracy: 0.5920
Epoch 18/30
24/24 [==============================] - 1s 53ms/step - loss: -0.0115 - accuracy: 0.9973
- val_loss: 2.3709 - val_accuracy: 0.6000
Epoch 19/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0127 - accuracy: 0.9973
- val_loss: 2.3797 - val_accuracy: 0.6080
Epoch 20/30
24/24 [==============================] - 1s 47ms/step - loss: -0.0146 - accuracy: 0.9973
- val_loss: 2.5529 - val_accuracy: 0.6040
Epoch 21/30
24/24 [==============================] - 1s 63ms/step - loss: -0.0115 - accuracy: 0.9973
- val_loss: 2.6361 - val_accuracy: 0.6080
Epoch 22/30
24/24 [==============================] - 1s 56ms/step - loss: 0.0060 - accuracy: 0.9973 -
val_loss: 2.0028 - val_accuracy: 0.5960
Epoch 23/30
24/24 [==============================] - 1s 56ms/step - loss: -0.0041 - accuracy: 0.9960
- val_loss: 2.5822 - val_accuracy: 0.5880
Epoch 24/30
24/24 [==============================] - 1s 53ms/step - loss: 0.0107 - accuracy: 0.9973 -
val_loss: 3.4793 - val_accuracy: 0.5880
Epoch 25/30
24/24 [==============================] - 1s 49ms/step - loss: 9.9316e-04 - accuracy: 0.99
73 - val_loss: 3.5396 - val_accuracy: 0.5640
Epoch 26/30
24/24 [==============================] - 1s 51ms/step - loss: -0.0037 - accuracy: 0.9973
- val_loss: 3.6339 - val_accuracy: 0.5600
Epoch 27/30
24/24 [==============================] - 1s 53ms/step - loss: 0.0098 - accuracy: 0.9973 -
val_loss: 3.5624 - val_accuracy: 0.5720
Epoch 28/30
24/24 [==============================] - 1s 52ms/step - loss: 0.0032 - accuracy: 0.9973 -
val_loss: 3.6047 - val_accuracy: 0.5720
Epoch 29/30
24/24 [==============================] - 1s 50ms/step - loss: 0.0065 - accuracy: 0.9973 -
val_loss: 3.5313 - val_accuracy: 0.5680
Epoch 30/30
24/24 [==============================] - 1s 49ms/step - loss: 2.9854e-05 - accuracy: 0.99
73 - val_loss: 3.5682 - val_accuracy: 0.5640
```

In [148]:

```
plot_graphs(history_lstm_2, "accuracy")
plot_graphs(history_lstm_2, "loss")
```





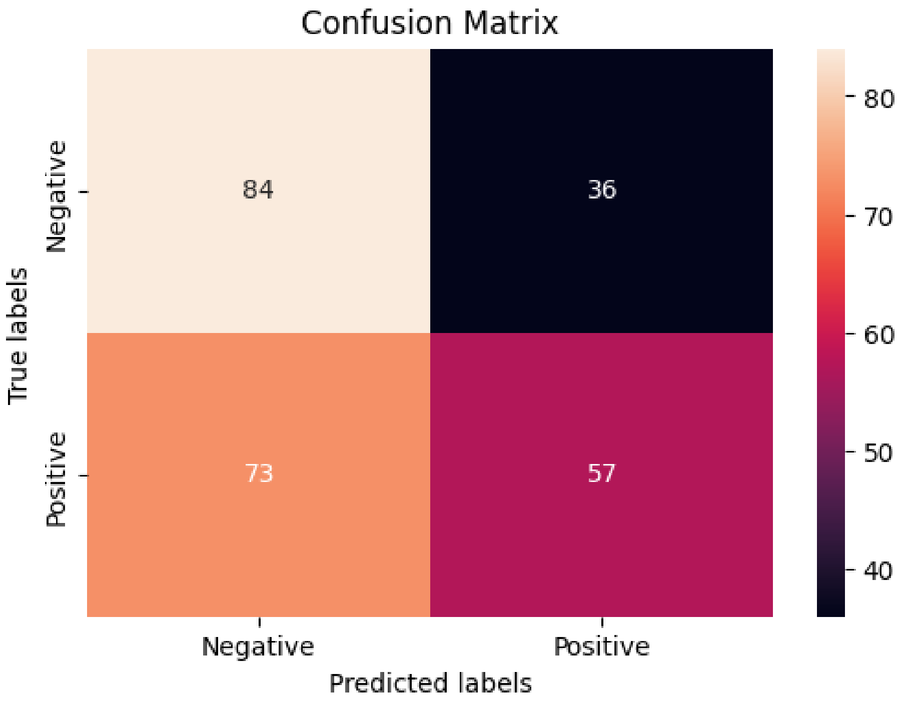


In [149]:

```
# Predict labels for testing data
predicted_labels = (model_lstm_2.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
```

```
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 1s 25ms/step
Accuracy: 0.564
Precision: 0.6129032258064516
Recall: 0.43846153846153846
F1-score: 0.5112107623318385
```



LSTM (3 Layers and 0.3 Dropout)

In [150]:

```
model_lstm_3 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.LSTM(units=64, return_sequences=True),   # First layer
     tf.keras.layers.LSTM(units=128, return_sequences=True),
    tf.keras.layers.LSTM(units=64),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_lstm_3.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_lstm_3.summary()
```

```
Model: "sequential_23"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_23 (Embedding)    (None, 30, 16)            65376     
                                                                 
 lstm_8 (LSTM)               (None, 30, 64)            20736     
                                                                 
 lstm_9 (LSTM)               (None, 30, 128)           98816     
                                                                 
 lstm_10 (LSTM)              (None, 64)                49408     
                                                                 
 dropout_21 (Dropout)        (None, 64)                0         
                                                                 
 dense_23 (Dense)            (None, 1)                 65        
                                                                 
=================================================================
Total params: 234,401
Trainable params: 234,401
Trainable params: 234,401
Non-trainable params: 0
_________________________________________________________________
```

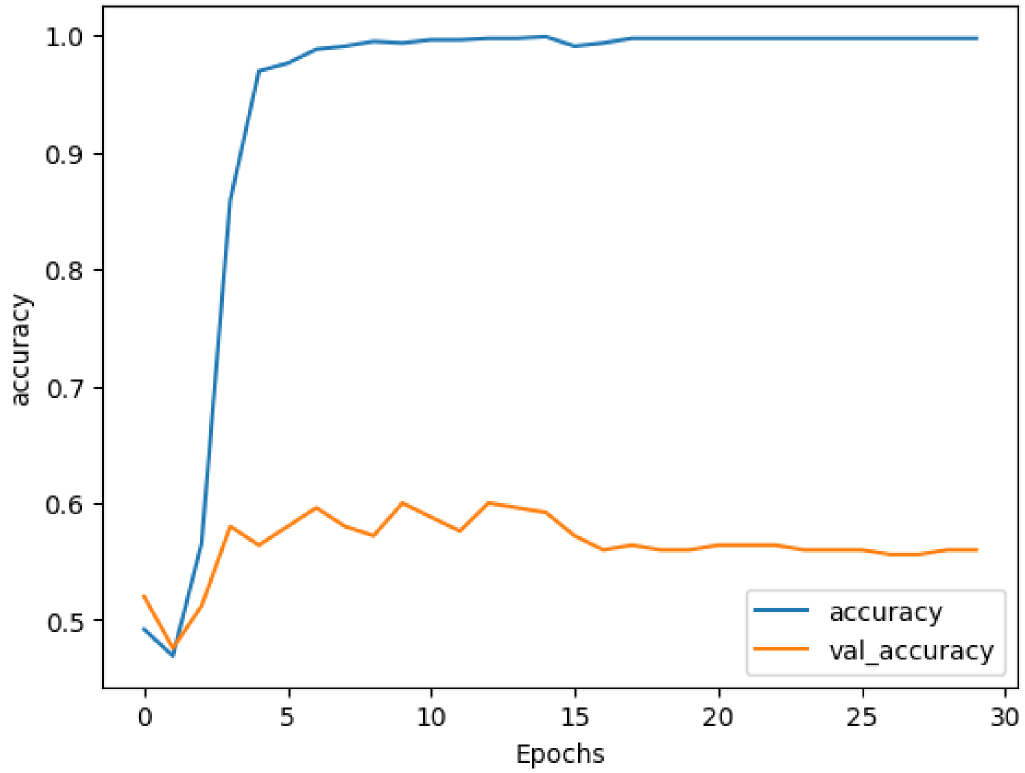
In [151]:

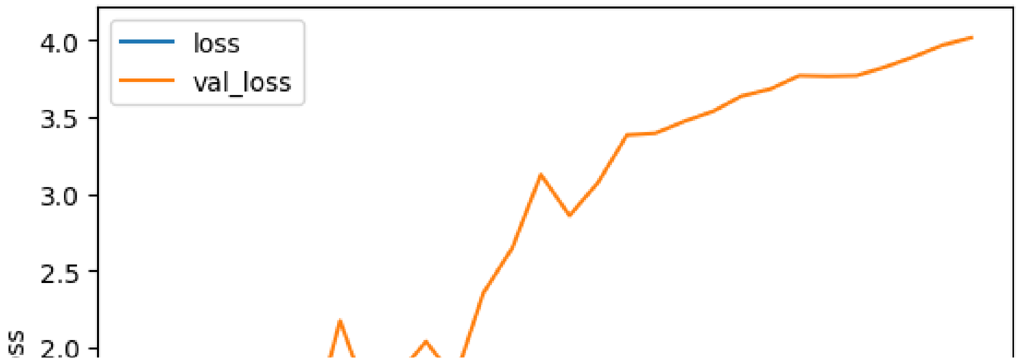
```
num_epochs = 30
history_lstm_3 = model_lstm_3.fit(training_padded, training_labels_final, epochs=num_epoc
hs, validation_data=(testing_padded, testing_labels_final))
```

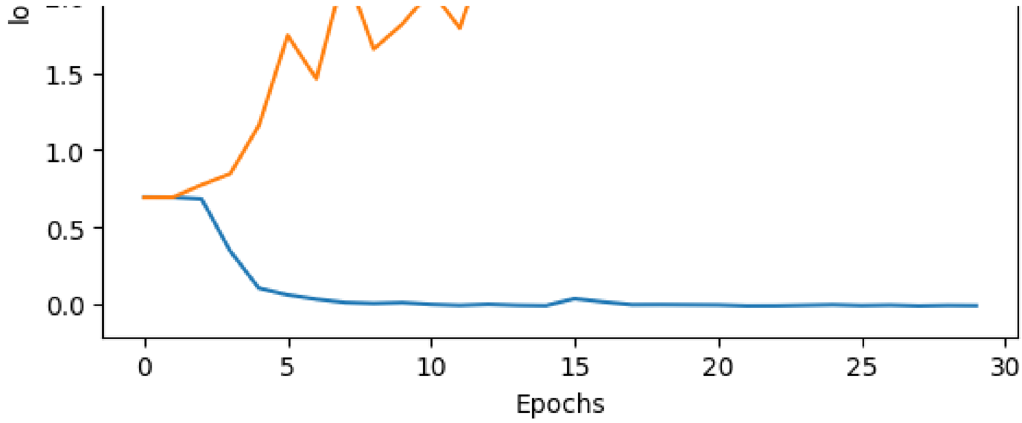
```
Epoch 1/30
24/24 [==============================] - 11s 127ms/step - loss: 0.6946 - accuracy: 0.4920
- val_loss: 0.6924 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 1s 57ms/step - loss: 0.6932 - accuracy: 0.4693 -
val_loss: 0.6934 - val_accuracy: 0.4760
Epoch 3/30
24/24 [==============================] - 2s 66ms/step - loss: 0.6820 - accuracy: 0.5653 -
val_loss: 0.7735 - val_accuracy: 0.5120
Epoch 4/30
24/24 [==============================] - 1s 62ms/step - loss: 0.3429 - accuracy: 0.8587 -
val_loss: 0.8461 - val_accuracy: 0.5800
Epoch 5/30
24/24 [==============================] - 2s 66ms/step - loss: 0.1013 - accuracy: 0.9693 -
val_loss: 1.1610 - val_accuracy: 0.5640
Epoch 6/30
24/24 [==============================] - 2s 69ms/step - loss: 0.0576 - accuracy: 0.9760 -
val_loss: 1.7468 - val_accuracy: 0.5800
Epoch 7/30
24/24 [==============================] - 1s 61ms/step - loss: 0.0300 - accuracy: 0.9880 -
val_loss: 1.4640 - val_accuracy: 0.5960
Epoch 8/30
24/24 [==============================] - 2s 70ms/step - loss: 0.0083 - accuracy: 0.9907 -
val_loss: 2.1766 - val_accuracy: 0.5800
Epoch 9/30
24/24 [==============================] - 2s 66ms/step - loss: 0.0024 - accuracy: 0.9947 -
val_loss: 1.6574 - val_accuracy: 0.5720
Epoch 10/30
24/24 [==============================] - 2s 73ms/step - loss: 0.0075 - accuracy: 0.9933 -
val_loss: 1.8204 - val_accuracy: 0.6000
Epoch 11/30
24/24 [==============================] - 2s 66ms/step - loss: -0.0039 - accuracy: 0.9960
- val_loss: 2.0404 - val_accuracy: 0.5880
Epoch 12/30
24/24 [==============================] - 1s 63ms/step - loss: -0.0098 - accuracy: 0.9960
- val_loss: 1.7940 - val_accuracy: 0.5760
Epoch 13/30
24/24 [==============================] - 1s 58ms/step - loss: -0.0032 - accuracy: 0.9973
- val_loss: 2.3573 - val_accuracy: 0.6000
Epoch 14/30
24/24 [==============================] - 1s 62ms/step - loss: -0.0094 - accuracy: 0.9973
- val_loss: 2.6478 - val_accuracy: 0.5960
Epoch 15/30
24/24 [==============================] - 1s 61ms/step - loss: -0.0118 - accuracy: 0.9987
- val_loss: 3.1272 - val_accuracy: 0.5920
Epoch 16/30
24/24 [==============================] - 1s 61ms/step - loss: 0.0336 - accuracy: 0.9907 -
val_loss: 2.8597 - val_accuracy: 0.5720
Epoch 17/30
24/24 [==============================] - 2s 64ms/step - loss: 0.0122 - accuracy: 0.9933 -
val_loss: 3.0772 - val_accuracy: 0.5600
Epoch 18/30
24/24 [==============================] - 1s 61ms/step - loss: -0.0059 - accuracy: 0.9973
- val_loss: 3.3843 - val_accuracy: 0.5640
Epoch 19/30
24/24 [==============================] - 1s 62ms/step - loss: -0.0053 - accuracy: 0.9973
- val_loss: 3.3962 - val_accuracy: 0.5600
Epoch 20/30
24/24 [==============================] - 2s 69ms/step - loss: -0.0064 - accuracy: 0.9973
- val_loss: 3.4739 - val_accuracy: 0.5600
Epoch 21/30
24/24 [==============================] - 2s 64ms/step - loss: -0.0074 - accuracy: 0.9973
- val_loss: 3.5391 - val_accuracy: 0.5640
- val_loss: 3.5391 - val_accuracy: 0.5640
Epoch 22/30
24/24 [==============================] - 2s 63ms/step - loss: -0.0130 - accuracy: 0.9973
- val_loss: 3.6378 - val_accuracy: 0.5640
Epoch 23/30
24/24 [==============================] - 1s 61ms/step - loss: -0.0126 - accuracy: 0.9973
- val_loss: 3.6839 - val_accuracy: 0.5640
Epoch 24/30
24/24 [==============================] - 2s 68ms/step - loss: -0.0094 - accuracy: 0.9973
- val_loss: 3.7714 - val_accuracy: 0.5600
Epoch 25/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0060 - accuracy: 0.9973
- val_loss: 3.7676 - val_accuracy: 0.5600
Epoch 26/30
24/24 [==============================] - 1s 63ms/step - loss: -0.0109 - accuracy: 0.9973
- val_loss: 3.7712 - val_accuracy: 0.5600
Epoch 27/30
24/24 [==============================] - 2s 68ms/step - loss: -0.0082 - accuracy: 0.9973
- val_loss: 3.8271 - val_accuracy: 0.5560
Epoch 28/30
24/24 [==============================] - 2s 66ms/step - loss: -0.0131 - accuracy: 0.9973
- val_loss: 3.8941 - val_accuracy: 0.5560
Epoch 29/30
24/24 [==============================] - 2s 66ms/step - loss: -0.0100 - accuracy: 0.9973
- val_loss: 3.9699 - val_accuracy: 0.5600
Epoch 30/30
24/24 [==============================] - 1s 62ms/step - loss: -0.0113 - accuracy: 0.9973
- val_loss: 4.0178 - val_accuracy: 0.5600
```

In [153]:

```
plot_graphs(history_lstm_3, "accuracy")
plot_graphs(history_lstm_3, "loss")
```



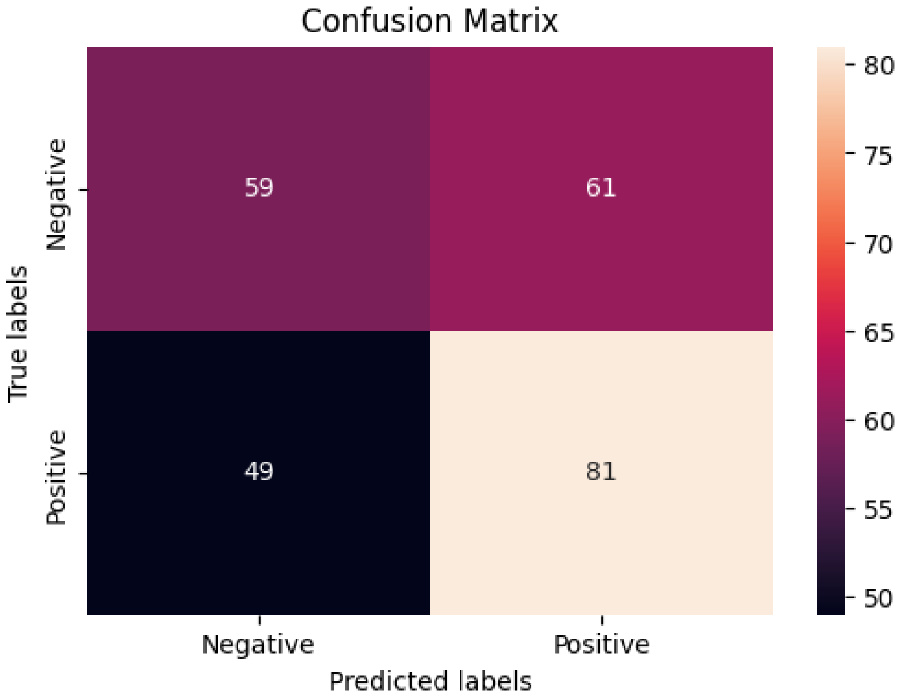




In [154]:

```
# Predict labels for testing data
predicted_labels = (model_lstm_3.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 2s 27ms/step
Accuracy: 0.56
Precision: 0.5704225352112676
Recall: 0.6230769230769231
F1-score: 0.5955882352941178
```



LSTM (3 Layers and 0.7 Dropout)

In [152]:

```
model_lstm_4 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.LSTM(units=64, return_sequences=True),   # First layer
     tf.keras.layers.LSTM(units=128, return_sequences=True),
    tf.keras.layers.LSTM(units=64),
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_lstm_4.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model_lstm_4.summary()
```

```
Model: "sequential_24"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_24 (Embedding)    (None, 30, 16)            65376     
                                                                 
 lstm_11 (LSTM)              (None, 30, 64)            20736     
                                                                 
 lstm_12 (LSTM)              (None, 30, 128)           98816     
                                                                 
 lstm_13 (LSTM)              (None, 64)                49408     
                                                                 
 dropout_22 (Dropout)        (None, 64)                0         
                                                                 
 dense_24 (Dense)            (None, 1)                 65        
                                                                 
=================================================================
Total params: 234,401
Trainable params: 234,401
Non-trainable params: 0
_________________________________________________________________
```

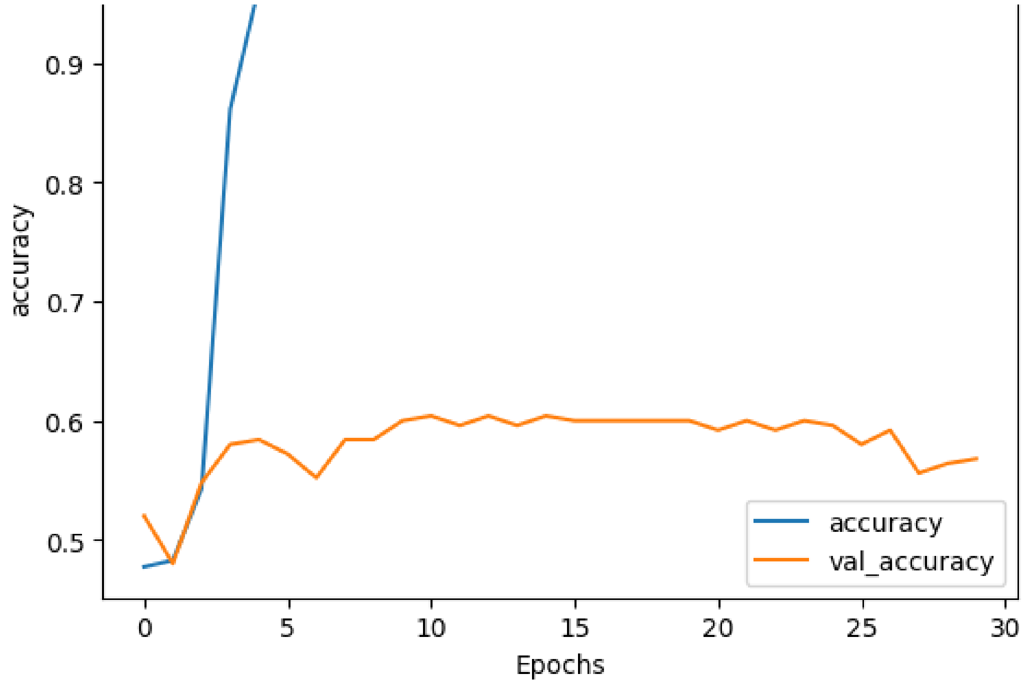
In [155]:

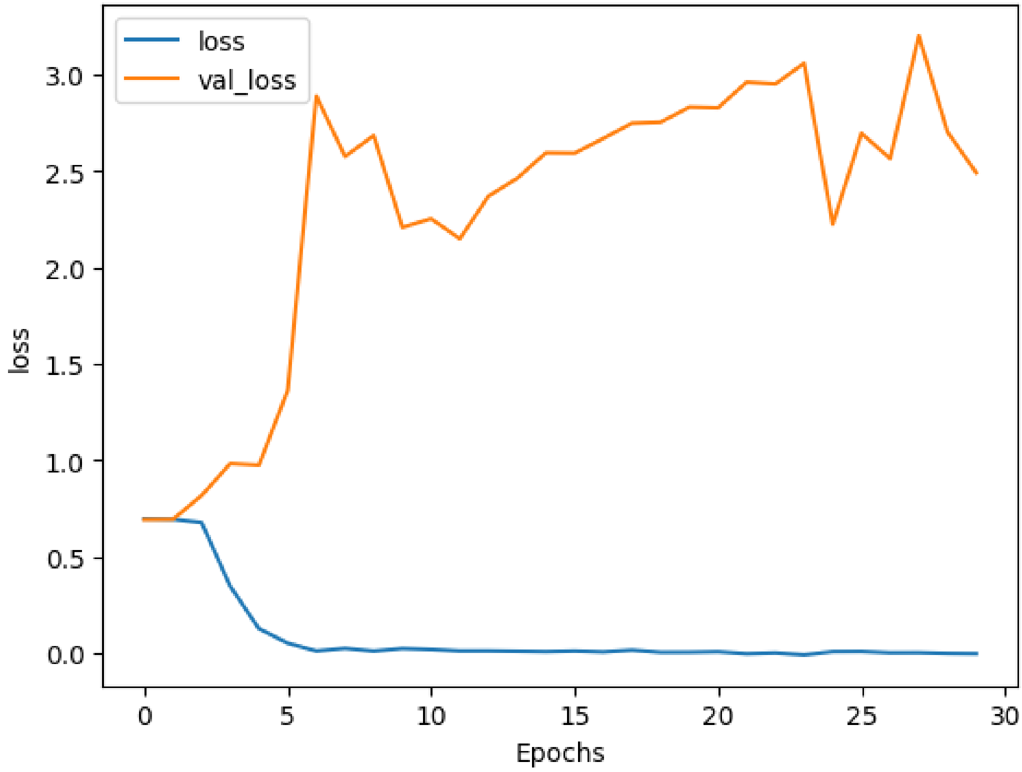
```
num_epochs = 30
history_lstm_4 = model_lstm_4.fit(training_padded, training_labels_final, epochs=num_epoc
hs, validation_data=(testing_padded, testing_labels_final))
```

```
Epoch 1/30
24/24 [==============================] - 11s 126ms/step - loss: 0.6967 - accuracy: 0.4773
- val_loss: 0.6926 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 2s 66ms/step - loss: 0.6939 - accuracy: 0.4827 -
val_loss: 0.6939 - val_accuracy: 0.4800
Epoch 3/30
24/24 [==============================] - 2s 66ms/step - loss: 0.6769 - accuracy: 0.5427 -
val_loss: 0.8168 - val_accuracy: 0.5480
Epoch 4/30
24/24 [==============================] - 2s 65ms/step - loss: 0.3470 - accuracy: 0.8613 -
val_loss: 0.9829 - val_accuracy: 0.5800
Epoch 5/30
24/24 [==============================] - 1s 62ms/step - loss: 0.1269 - accuracy: 0.9640 -
val_loss: 0.9747 - val_accuracy: 0.5840
Epoch 6/30
24/24 [==============================] - 2s 64ms/step - loss: 0.0524 - accuracy: 0.9853 -
val_loss: 1.3614 - val_accuracy: 0.5720
Epoch 7/30
24/24 [==============================] - 1s 62ms/step - loss: 0.0115 - accuracy: 0.9973 -
val_loss: 2.8877 - val_accuracy: 0.5520
Epoch 8/30
24/24 [==============================] - 2s 66ms/step - loss: 0.0252 - accuracy: 0.9947 -
val_loss: 2.5739 - val_accuracy: 0.5840
Epoch 9/30
24/24 [==============================] - 2s 67ms/step - loss: 0.0108 - accuracy: 0.9960 -
val_loss: 2.6829 - val_accuracy: 0.5840
Epoch 10/30
Epoch 10/30
24/24 [==============================] - 1s 63ms/step - loss: 0.0243 - accuracy: 0.9947 -
val_loss: 2.2067 - val_accuracy: 0.6000
Epoch 11/30
24/24 [==============================] - 2s 66ms/step - loss: 0.0195 - accuracy: 0.9960 -
val_loss: 2.2518 - val_accuracy: 0.6040
Epoch 12/30
24/24 [==============================] - 2s 63ms/step - loss: 0.0116 - accuracy: 0.9947 -
val_loss: 2.1474 - val_accuracy: 0.5960
Epoch 13/30
24/24 [==============================] - 2s 64ms/step - loss: 0.0117 - accuracy: 0.9960 -
val_loss: 2.3684 - val_accuracy: 0.6040
Epoch 14/30
24/24 [==============================] - 2s 74ms/step - loss: 0.0099 - accuracy: 0.9960 -
val_loss: 2.4613 - val_accuracy: 0.5960
Epoch 15/30
24/24 [==============================] - 2s 65ms/step - loss: 0.0079 - accuracy: 0.9960 -
val_loss: 2.5926 - val_accuracy: 0.6040
Epoch 16/30
24/24 [==============================] - 2s 67ms/step - loss: 0.0110 - accuracy: 0.9960 -
val_loss: 2.5911 - val_accuracy: 0.6000
Epoch 17/30
24/24 [==============================] - 2s 75ms/step - loss: 0.0065 - accuracy: 0.9960 -
val_loss: 2.6673 - val_accuracy: 0.6000
Epoch 18/30
24/24 [==============================] - 2s 71ms/step - loss: 0.0157 - accuracy: 0.9960 -
val_loss: 2.7463 - val_accuracy: 0.6000
Epoch 19/30
24/24 [==============================] - 2s 67ms/step - loss: 0.0047 - accuracy: 0.9960 -
val_loss: 2.7516 - val_accuracy: 0.6000
Epoch 20/30
24/24 [==============================] - 2s 70ms/step - loss: 0.0050 - accuracy: 0.9960 -
val_loss: 2.8301 - val_accuracy: 0.6000
Epoch 21/30
24/24 [==============================] - 2s 70ms/step - loss: 0.0075 - accuracy: 0.9960 -
val_loss: 2.8263 - val_accuracy: 0.5920
Epoch 22/30
24/24 [==============================] - 2s 67ms/step - loss: -0.0024 - accuracy: 0.9960
- val_loss: 2.9592 - val_accuracy: 0.6000
Epoch 23/30
24/24 [==============================] - 2s 70ms/step - loss: 0.0018 - accuracy: 0.9960 -
val_loss: 2.9499 - val_accuracy: 0.5920
Epoch 24/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0086 - accuracy: 0.9973
- val_loss: 3.0571 - val_accuracy: 0.6000
Epoch 25/30
24/24 [==============================] - 2s 72ms/step - loss: 0.0081 - accuracy: 0.9960 -
val_loss: 2.2221 - val_accuracy: 0.5960
Epoch 26/30
24/24 [==============================] - 2s 69ms/step - loss: 0.0088 - accuracy: 0.9973 -
val_loss: 2.6941 - val_accuracy: 0.5800
Epoch 27/30
24/24 [==============================] - 2s 69ms/step - loss: 0.0023 - accuracy: 0.9973 -
val_loss: 2.5631 - val_accuracy: 0.5920
Epoch 28/30
24/24 [==============================] - 2s 74ms/step - loss: 0.0027 - accuracy: 0.9960 -
val_loss: 3.1997 - val_accuracy: 0.5560
Epoch 29/30
24/24 [==============================] - 2s 76ms/step - loss: -8.0418e-04 - accuracy: 0.9
960 - val_loss: 2.7006 - val_accuracy: 0.5640
Epoch 30/30
24/24 [==============================] - 2s 72ms/step - loss: -0.0021 - accuracy: 0.9973
- val_loss: 2.4899 - val_accuracy: 0.5680
```

In [156]:

```
plot_graphs(history_lstm_4, "accuracy")
plot_graphs(history_lstm_4, "loss")
```

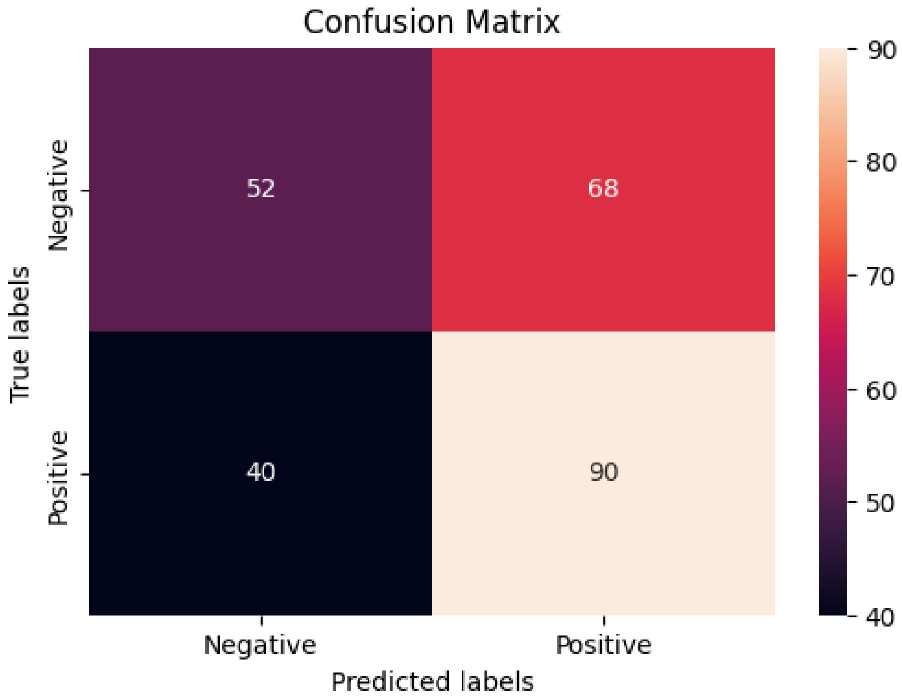




In [158]:

```
# Predict labels for testing data
predicted_labels = (model_lstm_4.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 3s 24ms/step
Accuracy: 0.568
Precision: 0.569620253164557
Recall: 0.6923076923076923
F1-score: 0.625
```



Bi-LSTM (2 Layers and 0.3 Dropout)

In [159]:

```
model_bilstm_1 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=64, return_sequences=True))
,  # First BiLSTM layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=128)),  # Second BiLSTM lay
er
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_bilstm_1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'
])
model_bilstm_1.summary()
```

```
Model: "sequential_25"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_25 (Embedding)    (None, 30, 16)            65376     
                                                                 
 bidirectional (Bidirectiona  (None, 30, 128)          41472     
 l)                                                              
                                                                 
 bidirectional_1 (Bidirectio  (None, 256)              263168    
 nal)                                                            
                                                                 
 dropout_23 (Dropout)        (None, 256)               0         
                                                                 
 dense_25 (Dense)            (None, 1)                 257       
                                                                 
=================================================================
Total params: 370,273
Total params: 370,273
Trainable params: 370,273
Non-trainable params: 0
_________________________________________________________________
```

In [160]:

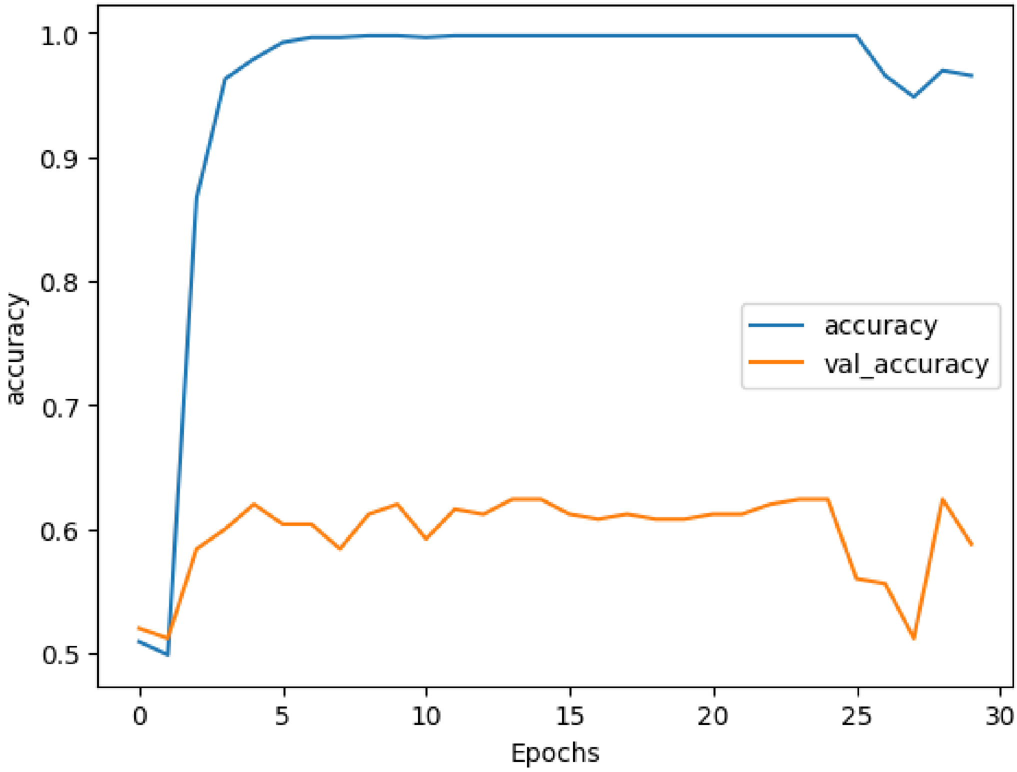
```
num_epochs = 30
history_bilstm_1 = model_bilstm_1.fit(training_padded, training_labels_final, epochs=num_
epochs, validation_data=(testing_padded, testing_labels_final))
```

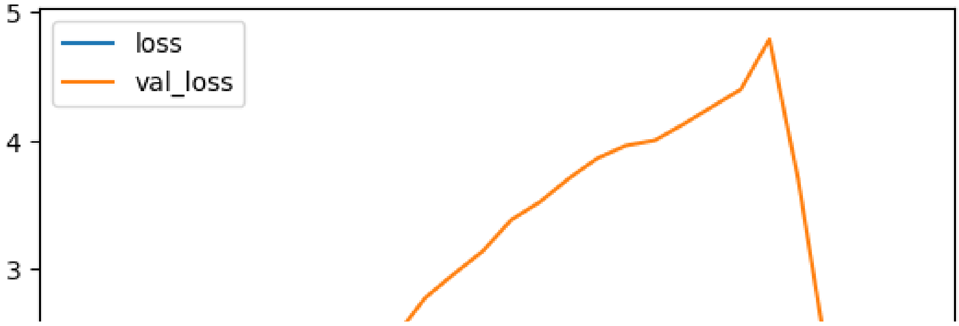
```
Epoch 1/30
24/24 [==============================] - 12s 148ms/step - loss: 0.6942 - accuracy: 0.5093
- val_loss: 0.6917 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 2s 68ms/step - loss: 0.6893 - accuracy: 0.4987 -
val_loss: 0.6913 - val_accuracy: 0.5120
Epoch 3/30
24/24 [==============================] - 2s 70ms/step - loss: 0.4366 - accuracy: 0.8667 -
val_loss: 0.8825 - val_accuracy: 0.5840
Epoch 4/30
24/24 [==============================] - 2s 72ms/step - loss: 0.1109 - accuracy: 0.9627 -
val_loss: 0.9275 - val_accuracy: 0.6000
Epoch 5/30
24/24 [==============================] - 2s 69ms/step - loss: 0.0632 - accuracy: 0.9787 -
val_loss: 1.1421 - val_accuracy: 0.6200
Epoch 6/30
24/24 [==============================] - 2s 68ms/step - loss: 0.0176 - accuracy: 0.9920 -
val_loss: 1.9641 - val_accuracy: 0.6040
Epoch 7/30
24/24 [==============================] - 2s 71ms/step - loss: 0.0080 - accuracy: 0.9960 -
val_loss: 1.5784 - val_accuracy: 0.6040
Epoch 8/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0030 - accuracy: 0.9960
- val_loss: 2.2944 - val_accuracy: 0.5840
Epoch 9/30
24/24 [==============================] - 2s 71ms/step - loss: -0.0036 - accuracy: 0.9973
- val_loss: 2.0815 - val_accuracy: 0.6120
Epoch 10/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0094 - accuracy: 0.9973
- val_loss: 2.4836 - val_accuracy: 0.6200
Epoch 11/30
24/24 [==============================] - 2s 70ms/step - loss: 0.0150 - accuracy: 0.9960 -
val_loss: 1.7484 - val_accuracy: 0.5920
Epoch 12/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0078 - accuracy: 0.9973
- val_loss: 2.4869 - val_accuracy: 0.6160
Epoch 13/30
24/24 [==============================] - 2s 69ms/step - loss: -0.0125 - accuracy: 0.9973
- val_loss: 2.7762 - val_accuracy: 0.6120
Epoch 14/30
24/24 [==============================] - 2s 72ms/step - loss: -0.0157 - accuracy: 0.9973
- val_loss: 2.9613 - val_accuracy: 0.6240
Epoch 15/30
24/24 [==============================] - 2s 68ms/step - loss: -0.0197 - accuracy: 0.9973
- val_loss: 3.1373 - val_accuracy: 0.6240
Epoch 16/30
24/24 [==============================] - 2s 68ms/step - loss: -0.0244 - accuracy: 0.9973
- val_loss: 3.3823 - val_accuracy: 0.6120
Epoch 17/30
24/24 [==============================] - 2s 69ms/step - loss: -0.0251 - accuracy: 0.9973
- val_loss: 3.5218 - val_accuracy: 0.6080
Epoch 18/30
24/24 [==============================] - 2s 69ms/step - loss: -0.0308 - accuracy: 0.9973
- val_loss: 3.7030 - val_accuracy: 0.6120
Epoch 19/30
24/24 [==============================] - 2s 72ms/step - loss: -0.0311 - accuracy: 0.9973
- val_loss: 3.8614 - val_accuracy: 0.6080
Epoch 20/30
24/24 [==============================] - 2s 71ms/step - loss: -0.0351 - accuracy: 0.9973
- val_loss: 3.9601 - val_accuracy: 0.6080
Epoch 21/30
24/24 [==============================] - 2s 71ms/step - loss: -0.0344 - accuracy: 0.9973
```

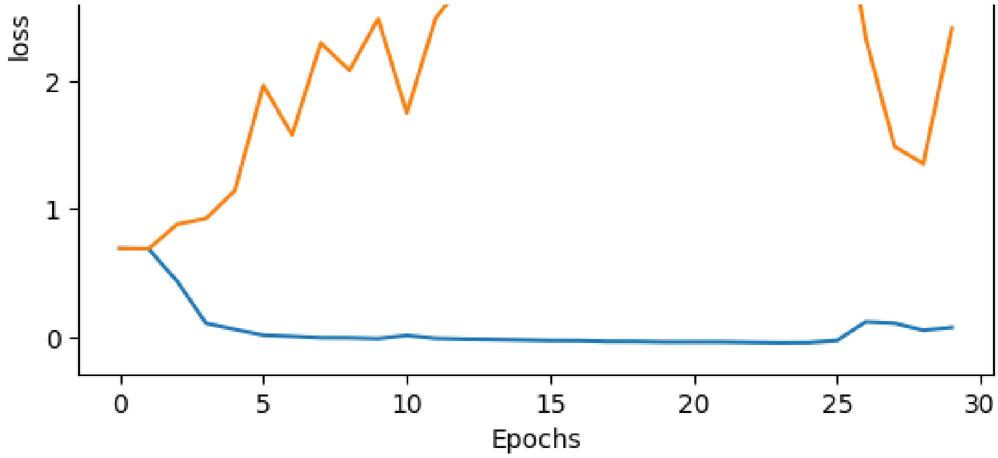
```
24/24 [==============================] - 2s 71ms/step - loss: -0.0344 - accuracy: 0.9973
- val_loss: 3.9999 - val_accuracy: 0.6120
Epoch 22/30
24/24 [==============================] - 2s 67ms/step - loss: -0.0349 - accuracy: 0.9973
- val_loss: 4.1256 - val_accuracy: 0.6120
Epoch 23/30
24/24 [==============================] - 2s 72ms/step - loss: -0.0384 - accuracy: 0.9973
- val_loss: 4.2619 - val_accuracy: 0.6200
Epoch 24/30
24/24 [==============================] - 2s 74ms/step - loss: -0.0412 - accuracy: 0.9973
- val_loss: 4.3963 - val_accuracy: 0.6240
Epoch 25/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0398 - accuracy: 0.9973
- val_loss: 4.7867 - val_accuracy: 0.6240
Epoch 26/30
24/24 [==============================] - 2s 72ms/step - loss: -0.0236 - accuracy: 0.9973
- val_loss: 3.6995 - val_accuracy: 0.5600
Epoch 27/30
24/24 [==============================] - 2s 72ms/step - loss: 0.1226 - accuracy: 0.9653 -
val_loss: 2.3287 - val_accuracy: 0.5560
Epoch 28/30
24/24 [==============================] - 2s 72ms/step - loss: 0.1114 - accuracy: 0.9480 -
val_loss: 1.4886 - val_accuracy: 0.5120
Epoch 29/30
24/24 [==============================] - 2s 77ms/step - loss: 0.0562 - accuracy: 0.9693 -
val_loss: 1.3533 - val_accuracy: 0.6240
Epoch 30/30
24/24 [==============================] - 2s 69ms/step - loss: 0.0782 - accuracy: 0.9653 -
val_loss: 2.4096 - val_accuracy: 0.5880
```

In [161]:

```
plot_graphs(history_bilstm_1, "accuracy")
plot_graphs(history_bilstm_1, "loss")
```



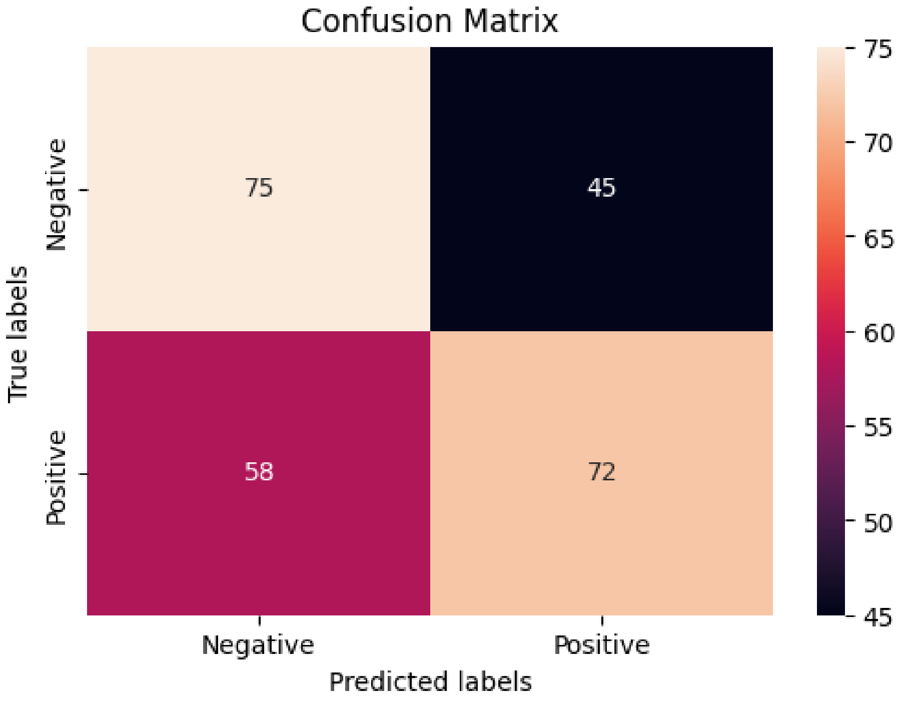




In [162]:

```
# Predict labels for testing data
predicted_labels = (model_bilstm_1.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 2s 27ms/step
Accuracy: 0.588
Precision: 0.6153846153846154
Recall: 0.5538461538461539
F1-score: 0.5829959514170041
```



Bi-LSTM (2 Layers and 0.7 Dropout)

In [163]:

```
model_bilstm_2 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=64, return_sequences=True))
,  # First BiLSTM layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=128)),  # Second BiLSTM lay
er
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_bilstm_2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'
])
model_bilstm_2.summary()
```

```
Model: "sequential_26"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_26 (Embedding)    (None, 30, 16)            65376     
                                                                 
 bidirectional_2 (Bidirectio  (None, 30, 128)          41472     
 nal)                                                            
                                                                 
 bidirectional_3 (Bidirectio  (None, 256)              263168    
 nal)                                                            
                                                                 
 dropout_24 (Dropout)        (None, 256)               0         
                                                                 
 dense_26 (Dense)            (None, 1)                 257       
                                                                 
=================================================================
Total params: 370,273
Trainable params: 370,273
Non-trainable params: 0
_________________________________________________________________
```

In [164]:

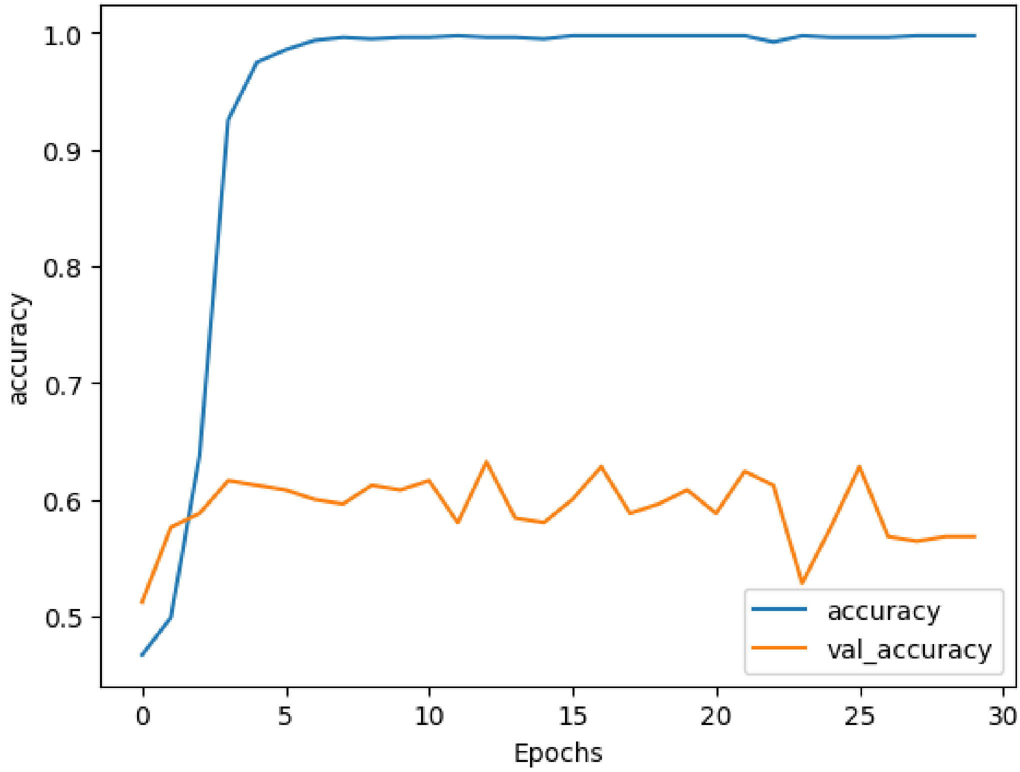
```
num_epochs = 30
history_bilstm_2 = model_bilstm_2.fit(training_padded, training_labels_final, epochs=num_
epochs, validation_data=(testing_padded, testing_labels_final))
```

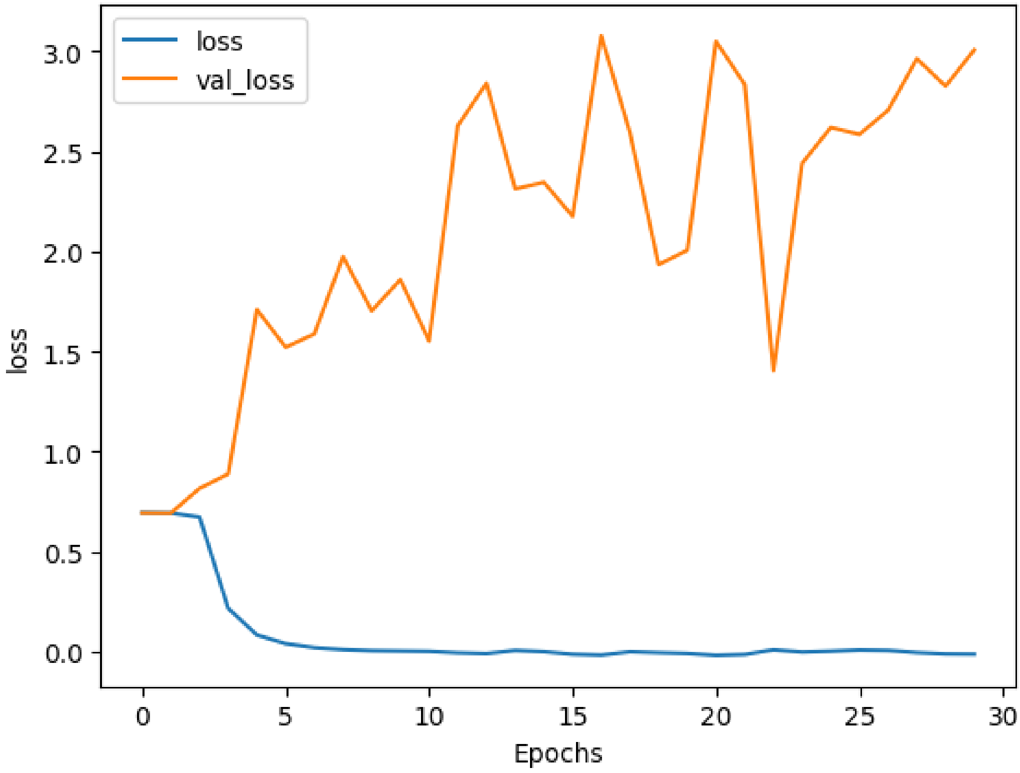
```
Epoch 1/30
24/24 [==============================] - 13s 158ms/step - loss: 0.6949 - accuracy: 0.4667
- val_loss: 0.6930 - val_accuracy: 0.5120
Epoch 2/30
24/24 [==============================] - 2s 71ms/step - loss: 0.6932 - accuracy: 0.4987 -
val_loss: 0.6922 - val_accuracy: 0.5760
Epoch 3/30
24/24 [==============================] - 2s 77ms/step - loss: 0.6720 - accuracy: 0.6373 -
val_loss: 0.8155 - val_accuracy: 0.5880
Epoch 4/30
24/24 [==============================] - 2s 75ms/step - loss: 0.2158 - accuracy: 0.9253 -
val_loss: 0.8877 - val_accuracy: 0.6160
Epoch 5/30
24/24 [==============================] - 2s 75ms/step - loss: 0.0823 - accuracy: 0.9747 -
val_loss: 1.7096 - val_accuracy: 0.6120
Epoch 6/30
24/24 [==============================] - 2s 77ms/step - loss: 0.0394 - accuracy: 0.9853 -
val_loss: 1.5213 - val_accuracy: 0.6080
Epoch 7/30
24/24 [==============================] - 2s 73ms/step - loss: 0.0193 - accuracy: 0.9933 -
val_loss: 1.5874 - val_accuracy: 0.6000
Epoch 8/30
24/24 [==============================] - 2s 79ms/step - loss: 0.0095 - accuracy: 0.9960 -
val_loss: 1.9744 - val_accuracy: 0.5960
```

```
Epoch 9/30
24/24 [==============================] - 2s 79ms/step - loss: 0.0039 - accuracy: 0.9947 -
val_loss: 1.7032 - val_accuracy: 0.6120
Epoch 10/30
24/24 [==============================] - 2s 73ms/step - loss: 0.0022 - accuracy: 0.9960 -
val_loss: 1.8595 - val_accuracy: 0.6080
Epoch 11/30
24/24 [==============================] - 2s 73ms/step - loss: 4.8597e-04 - accuracy: 0.99
60 - val_loss: 1.5525 - val_accuracy: 0.6160
Epoch 12/30
24/24 [==============================] - 2s 72ms/step - loss: -0.0066 - accuracy: 0.9973
- val_loss: 2.6294 - val_accuracy: 0.5800
Epoch 13/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0103 - accuracy: 0.9960
- val_loss: 2.8411 - val_accuracy: 0.6320
Epoch 14/30
24/24 [==============================] - 2s 71ms/step - loss: 0.0050 - accuracy: 0.9960 -
val_loss: 2.3132 - val_accuracy: 0.5840
Epoch 15/30
24/24 [==============================] - 2s 71ms/step - loss: -0.0010 - accuracy: 0.9947
- val_loss: 2.3460 - val_accuracy: 0.5800
Epoch 16/30
24/24 [==============================] - 2s 68ms/step - loss: -0.0136 - accuracy: 0.9973
- val_loss: 2.1768 - val_accuracy: 0.6000
Epoch 17/30
24/24 [==============================] - 2s 69ms/step - loss: -0.0177 - accuracy: 0.9973
- val_loss: 3.0787 - val_accuracy: 0.6280
Epoch 18/30
24/24 [==============================] - 2s 70ms/step - loss: -0.0014 - accuracy: 0.9973
- val_loss: 2.5944 - val_accuracy: 0.5880
Epoch 19/30
24/24 [==============================] - 2s 77ms/step - loss: -0.0059 - accuracy: 0.9973
- val_loss: 1.9342 - val_accuracy: 0.5960
Epoch 20/30
24/24 [==============================] - 2s 76ms/step - loss: -0.0096 - accuracy: 0.9973
- val_loss: 2.0063 - val_accuracy: 0.6080
Epoch 21/30
24/24 [==============================] - 2s 74ms/step - loss: -0.0190 - accuracy: 0.9973
- val_loss: 3.0504 - val_accuracy: 0.5880
Epoch 22/30
24/24 [==============================] - 2s 68ms/step - loss: -0.0148 - accuracy: 0.9973
- val_loss: 2.8360 - val_accuracy: 0.6240
Epoch 23/30
24/24 [==============================] - 2s 70ms/step - loss: 0.0083 - accuracy: 0.9920 -
val_loss: 1.4050 - val_accuracy: 0.6120
Epoch 24/30
24/24 [==============================] - 2s 68ms/step - loss: -0.0029 - accuracy: 0.9973
- val_loss: 2.4408 - val_accuracy: 0.5280
Epoch 25/30
24/24 [==============================] - 2s 72ms/step - loss: 0.0013 - accuracy: 0.9960 -
val_loss: 2.6200 - val_accuracy: 0.5760
Epoch 26/30
24/24 [==============================] - 2s 69ms/step - loss: 0.0073 - accuracy: 0.9960 -
val_loss: 2.5855 - val_accuracy: 0.6280
Epoch 27/30
24/24 [==============================] - 2s 77ms/step - loss: 0.0051 - accuracy: 0.9960 -
val_loss: 2.7071 - val_accuracy: 0.5680
Epoch 28/30
24/24 [==============================] - 2s 73ms/step - loss: -0.0055 - accuracy: 0.9973
- val_loss: 2.9639 - val_accuracy: 0.5640
Epoch 29/30
24/24 [==============================] - 2s 69ms/step - loss: -0.0119 - accuracy: 0.9973
- val_loss: 2.8270 - val_accuracy: 0.5680
Epoch 30/30
24/24 [==============================] - 2s 76ms/step - loss: -0.0130 - accuracy: 0.9973
- val_loss: 3.0081 - val_accuracy: 0.5680
```

In [165]:

```
plot_graphs(history_bilstm_2, "accuracy")
plot_graphs(history_bilstm_2, "loss")
```



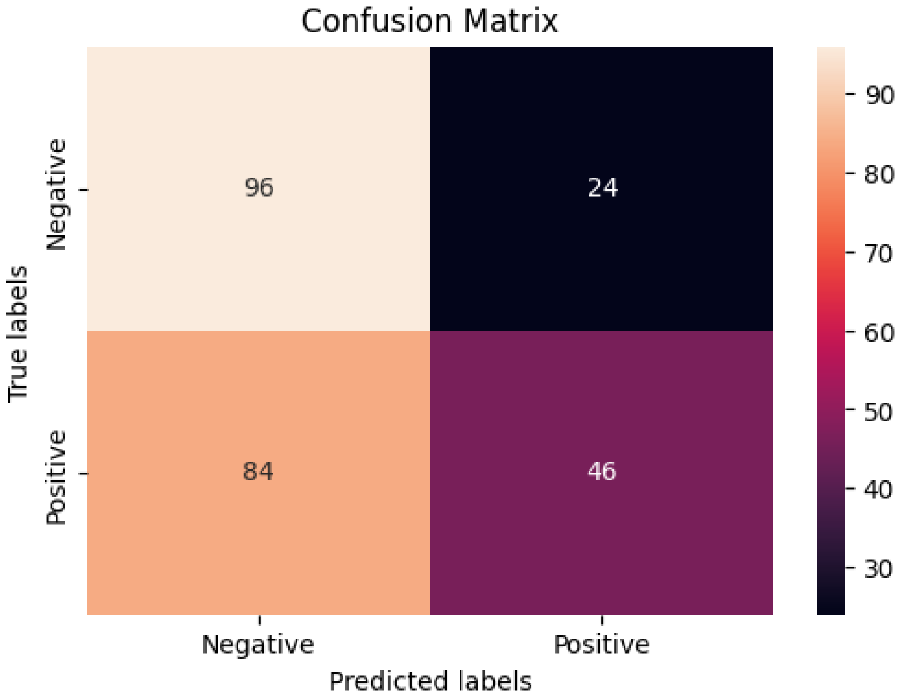


In [166]:

```
# Predict labels for testing data
predicted_labels = (model_bilstm_2.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
```

```
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 2s 22ms/step
Accuracy: 0.568
Precision: 0.6571428571428571
Recall: 0.35384615384615387
F1-score: 0.4600000000000001
```



Bi-LSTM (3 Layers and 0.3 Dropout)

In [168]:

```
model_bilstm_3 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=64, return_sequences=True))
,  # First BiLSTM layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=128, return_sequences=True)
),  # Second BiLSTM layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=64)),  # Third BiLSTM layer
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_bilstm_3.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'
])
model_bilstm_3.summary()
```

```
Model: "sequential_28"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_28 (Embedding)    (None, 30, 16)            65376     
                                                                 
 bidirectional_7 (Bidirectio  (None, 30, 128)          41472     
 nal)                                                            
                                                                 
 bidirectional_8 (Bidirectio  (None, 30, 256)          263168    
 nal)                                                            
                                                                 
 bidirectional_9 (Bidirectio  (None, 128)              164352    
 nal)
 nal)

 dropout_26 (Dropout)        (None, 128)               0

 dense_28 (Dense)            (None, 1)                 129

=================================================================
Total params: 534,497
Trainable params: 534,497
Non-trainable params: 0
_________________________________________________________________
```

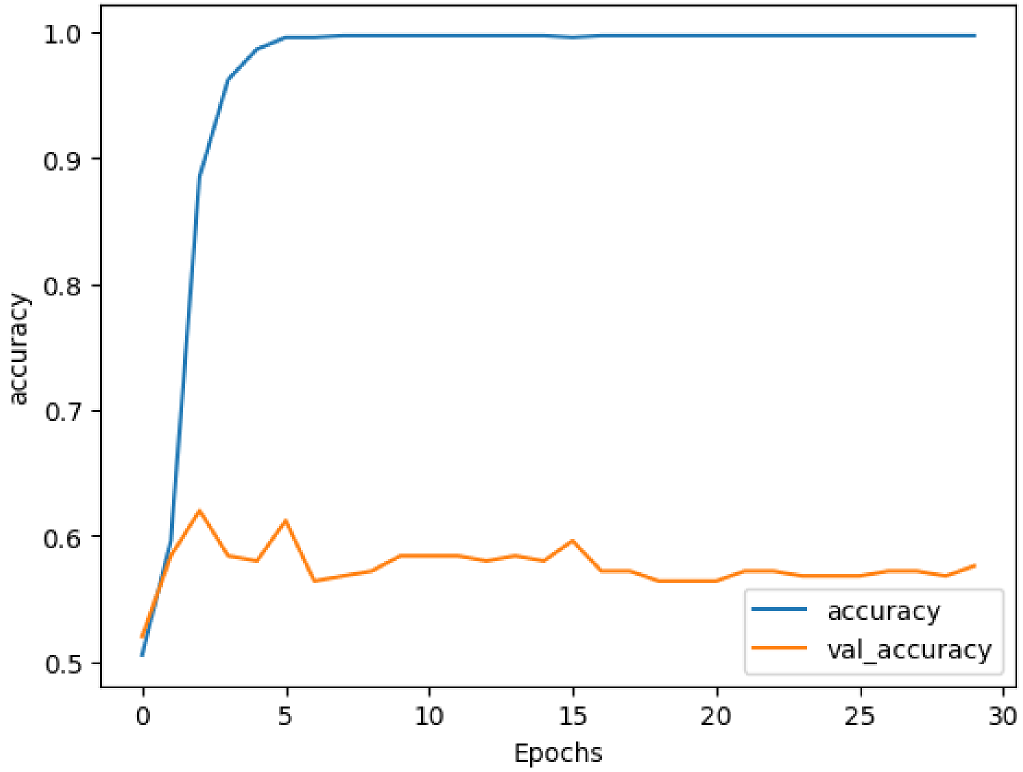
In [169]:

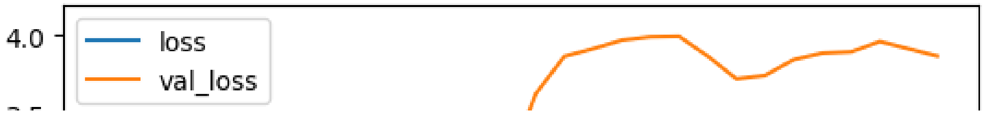
```
num_epochs = 30
history_bilstm_3 = model_bilstm_3.fit(training_padded, training_labels_final, epochs=num_
epochs, validation_data=(testing_padded, testing_labels_final))
```

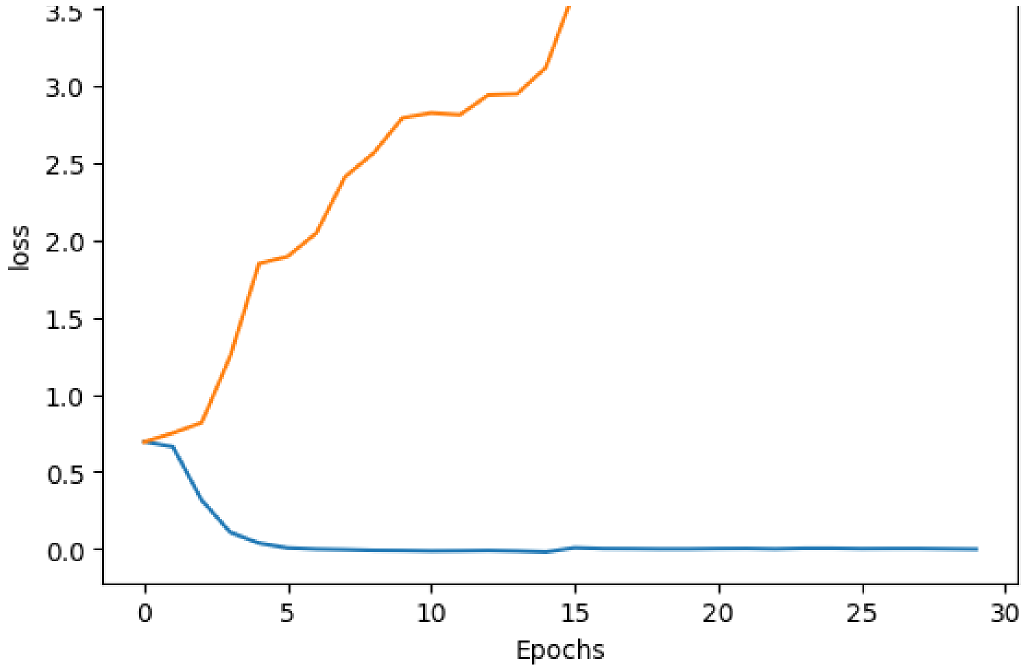
```
Epoch 1/30
24/24 [==============================] - 19s 208ms/step - loss: 0.6943 - accuracy: 0.5053
- val_loss: 0.6924 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 2s 95ms/step - loss: 0.6621 - accuracy: 0.5960 -
val_loss: 0.7500 - val_accuracy: 0.5840
Epoch 3/30
24/24 [==============================] - 2s 99ms/step - loss: 0.3164 - accuracy: 0.8853 -
val_loss: 0.8168 - val_accuracy: 0.6200
Epoch 4/30
24/24 [==============================] - 2s 94ms/step - loss: 0.1066 - accuracy: 0.9627 -
val_loss: 1.2482 - val_accuracy: 0.5840
Epoch 5/30
24/24 [==============================] - 2s 104ms/step - loss: 0.0367 - accuracy: 0.9867
- val_loss: 1.8480 - val_accuracy: 0.5800
Epoch 6/30
24/24 [==============================] - 2s 103ms/step - loss: 0.0055 - accuracy: 0.9960
- val_loss: 1.8948 - val_accuracy: 0.6120
Epoch 7/30
24/24 [==============================] - 2s 98ms/step - loss: -0.0020 - accuracy: 0.9960
- val_loss: 2.0465 - val_accuracy: 0.5640
Epoch 8/30
24/24 [==============================] - 2s 95ms/step - loss: -0.0051 - accuracy: 0.9973
- val_loss: 2.4114 - val_accuracy: 0.5680
Epoch 9/30
24/24 [==============================] - 2s 93ms/step - loss: -0.0096 - accuracy: 0.9973
- val_loss: 2.5654 - val_accuracy: 0.5720
Epoch 10/30
24/24 [==============================] - 2s 94ms/step - loss: -0.0108 - accuracy: 0.9973
- val_loss: 2.7930 - val_accuracy: 0.5840
Epoch 11/30
24/24 [==============================] - 2s 96ms/step - loss: -0.0136 - accuracy: 0.9973
- val_loss: 2.8254 - val_accuracy: 0.5840
Epoch 12/30
24/24 [==============================] - 2s 96ms/step - loss: -0.0126 - accuracy: 0.9973
- val_loss: 2.8133 - val_accuracy: 0.5840
Epoch 13/30
24/24 [==============================] - 2s 98ms/step - loss: -0.0106 - accuracy: 0.9973
- val_loss: 2.9417 - val_accuracy: 0.5800
Epoch 14/30
24/24 [==============================] - 2s 96ms/step - loss: -0.0142 - accuracy: 0.9973
- val_loss: 2.9488 - val_accuracy: 0.5840
Epoch 15/30
24/24 [==============================] - 2s 101ms/step - loss: -0.0201 - accuracy: 0.9973
- val_loss: 3.1203 - val_accuracy: 0.5800
Epoch 16/30
24/24 [==============================] - 2s 94ms/step - loss: 0.0065 - accuracy: 0.9960 -
val_loss: 3.6208 - val_accuracy: 0.5960
Epoch 17/30
24/24 [==============================] - 2s 97ms/step - loss: 9.6908e-04 - accuracy: 0.99
73 - val_loss: 3.8642 - val_accuracy: 0.5720
Epoch 18/30
24/24 [==============================] - 2s 102ms/step - loss: 2.9994e-04 - accuracy: 0.9
973 - val_loss: 3.9136 - val_accuracy: 0.5720
Epoch 19/30
Epoch 19/30
24/24 [==============================] - 2s 93ms/step - loss: -0.0013 - accuracy: 0.9973
- val_loss: 3.9683 - val_accuracy: 0.5640
Epoch 20/30
24/24 [==============================] - 3s 107ms/step - loss: -0.0012 - accuracy: 0.9973
- val_loss: 3.9901 - val_accuracy: 0.5640
Epoch 21/30
24/24 [==============================] - 2s 99ms/step - loss: 0.0010 - accuracy: 0.9973 -
val_loss: 3.9933 - val_accuracy: 0.5640
Epoch 22/30
24/24 [==============================] - 2s 93ms/step - loss: 0.0019 - accuracy: 0.9973 -
val_loss: 3.8661 - val_accuracy: 0.5720
Epoch 23/30
24/24 [==============================] - 3s 109ms/step - loss: -0.0011 - accuracy: 0.9973
- val_loss: 3.7194 - val_accuracy: 0.5720
Epoch 24/30
24/24 [==============================] - 3s 112ms/step - loss: 0.0030 - accuracy: 0.9973
- val_loss: 3.7413 - val_accuracy: 0.5680
Epoch 25/30
24/24 [==============================] - 2s 99ms/step - loss: 0.0030 - accuracy: 0.9973 -
val_loss: 3.8437 - val_accuracy: 0.5680
Epoch 26/30
24/24 [==============================] - 2s 100ms/step - loss: 4.1843e-04 - accuracy: 0.9
973 - val_loss: 3.8854 - val_accuracy: 0.5680
Epoch 27/30
24/24 [==============================] - 3s 110ms/step - loss: 0.0012 - accuracy: 0.9973
- val_loss: 3.8938 - val_accuracy: 0.5720
Epoch 28/30
24/24 [==============================] - 3s 110ms/step - loss: 0.0013 - accuracy: 0.9973
- val_loss: 3.9605 - val_accuracy: 0.5720
Epoch 29/30
24/24 [==============================] - 3s 106ms/step - loss: -7.1097e-04 - accuracy: 0.
9973 - val_loss: 3.9137 - val_accuracy: 0.5680
Epoch 30/30
24/24 [==============================] - 3s 106ms/step - loss: -0.0025 - accuracy: 0.9973
- val_loss: 3.8664 - val_accuracy: 0.5760
```

In [170]:

```
plot_graphs(history_bilstm_3, "accuracy")
plot_graphs(history_bilstm_3, "loss")
```



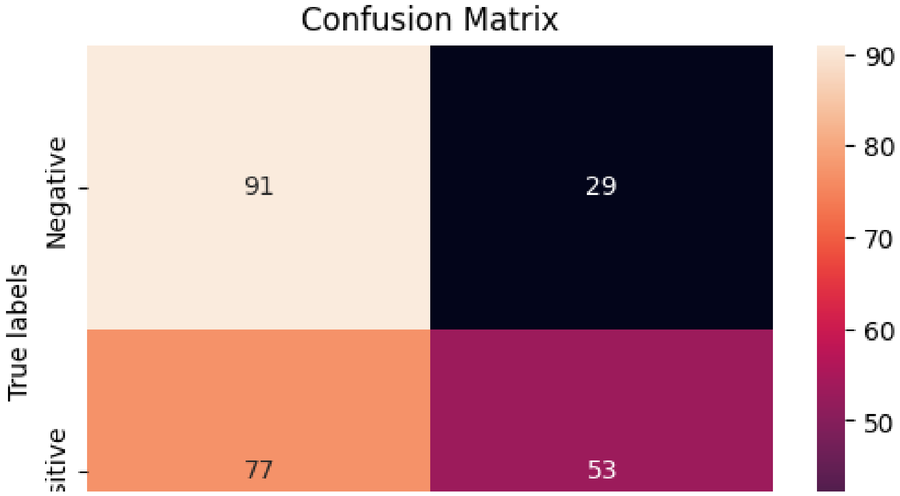




In [171]:

```
# Predict labels for testing data
predicted_labels = (model_bilstm_3.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 3s 32ms/step
Accuracy: 0.576
Precision: 0.6463414634146342
Recall: 0.4076923076923077
F1-score: 0.5
```



Bi-LSTM (3 Layers and 0.7 Dropout)

In [172]:

```
model_bilstm_4 = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_pad_len),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=64, return_sequences=True))
,  # First BiLSTM layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=128, return_sequences=True)
),  # Second BiLSTM layer
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(units=64)),  # Third BiLSTM layer
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(units=1, activation='sigmoid')
])
model_bilstm_4.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'
])
model_bilstm_4.summary()
```

```
Model: "sequential_29"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_29 (Embedding)    (None, 30, 16)            65376

 bidirectional_10 (Bidirecti  (None, 30, 128)          41472
 onal)

 bidirectional_11 (Bidirecti  (None, 30, 256)          263168
 onal)

 bidirectional_12 (Bidirecti  (None, 128)              164352
 onal)

 dropout_27 (Dropout)        (None, 128)               0

 dense_29 (Dense)            (None, 1)                 129

=================================================================
Total params: 534,497
Trainable params: 534,497
Non-trainable params: 0
_________________________________________________________________
```

In [173]:

```
num_epochs = 30
history_bilstm_4 = model_bilstm_4.fit(training_padded, training_labels_final, epochs=num_
epochs, validation_data=(testing_padded, testing_labels_final))
```

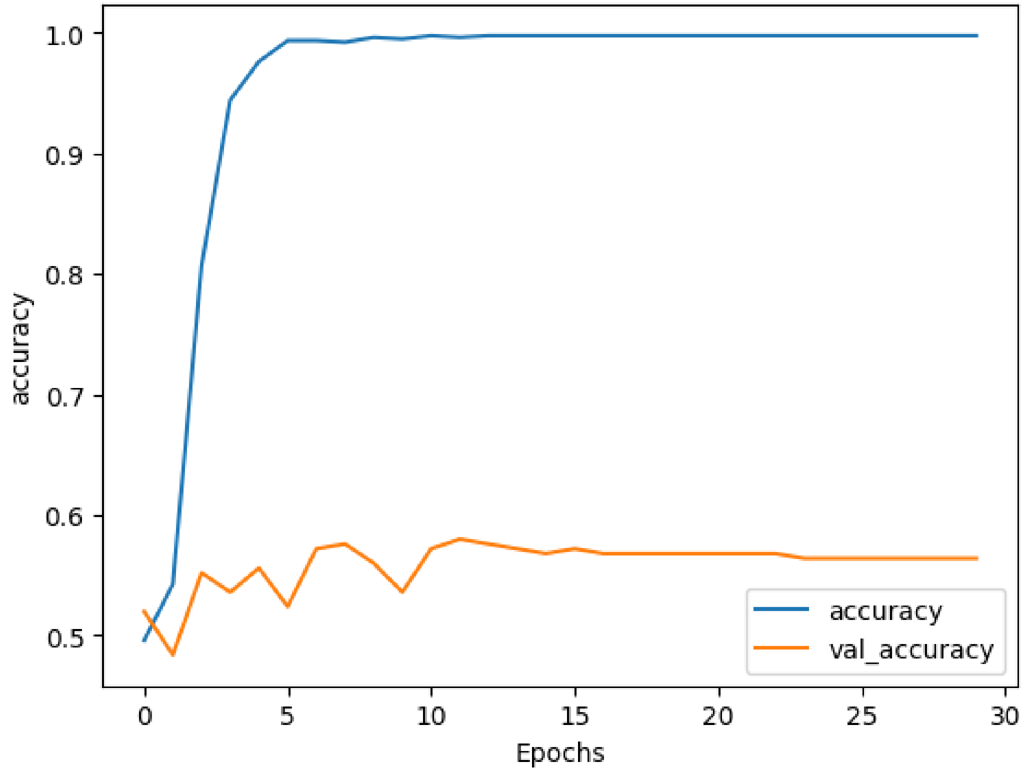
```
Epoch 1/30
24/24 [==============================] - 18s 222ms/step - loss: 0.6944 - accuracy: 0.4960
- val_loss: 0.6928 - val_accuracy: 0.5200
Epoch 2/30
24/24 [==============================] - 2s 101ms/step - loss: 0.6881 - accuracy: 0.5427
- val_loss: 0.6973 - val_accuracy: 0.4840
Epoch 3/30
24/24 [==============================] - 2s 103ms/step - loss: 0.4491 - accuracy: 0.8067
- val_loss: 0.9446 - val_accuracy: 0.5520
Epoch 4/30
24/24 [==============================] - 2s 104ms/step - loss: 0.1662 - accuracy: 0.9440
- val_loss: 1.5189 - val_accuracy: 0.5360
Epoch 5/30
```

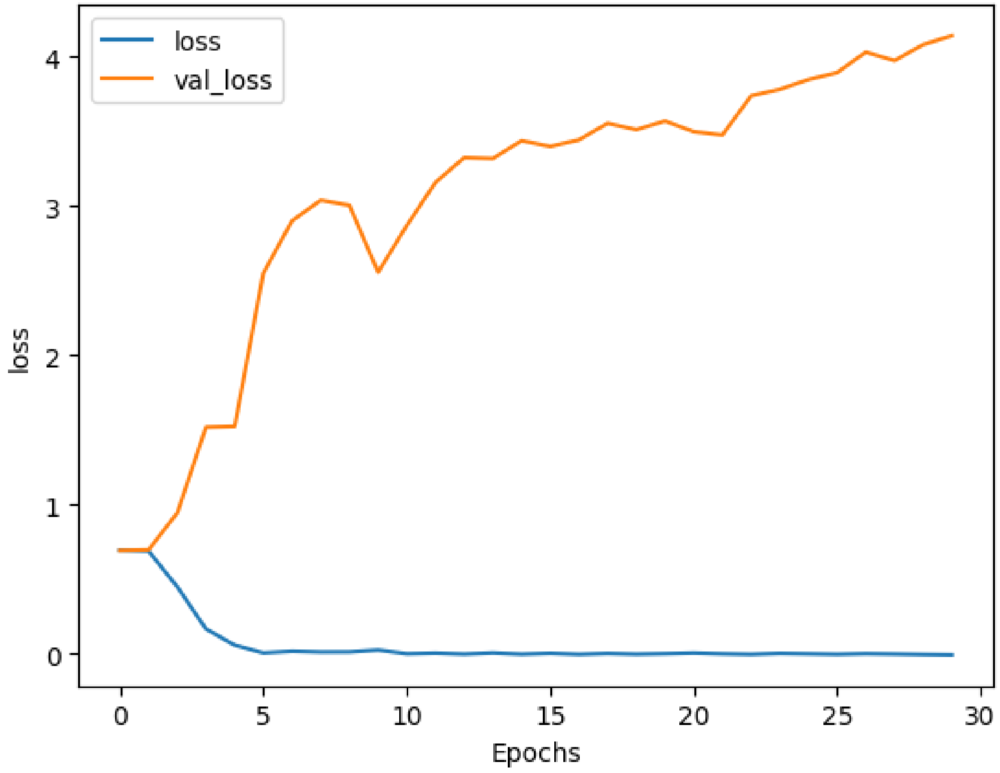
```
24/24 [==============================] - 2s 101ms/step - loss: 0.0584 - accuracy: 0.9760
- val_loss: 1.5243 - val_accuracy: 0.5560
Epoch 6/30
24/24 [==============================] - 2s 100ms/step - loss: 0.0058 - accuracy: 0.9933
- val_loss: 2.5514 - val_accuracy: 0.5240
Epoch 7/30
24/24 [==============================] - 2s 103ms/step - loss: 0.0173 - accuracy: 0.9933
- val_loss: 2.9025 - val_accuracy: 0.5720
Epoch 8/30
24/24 [==============================] - 2s 101ms/step - loss: 0.0120 - accuracy: 0.9920
- val_loss: 3.0400 - val_accuracy: 0.5760
Epoch 9/30
24/24 [==============================] - 3s 121ms/step - loss: 0.0125 - accuracy: 0.9960
- val_loss: 3.0061 - val_accuracy: 0.5600
Epoch 10/30
24/24 [==============================] - 2s 102ms/step - loss: 0.0253 - accuracy: 0.9947
- val_loss: 2.5574 - val_accuracy: 0.5360
Epoch 11/30
24/24 [==============================] - 2s 101ms/step - loss: 6.6547e-04 - accuracy: 0.9
973 - val_loss: 2.8706 - val_accuracy: 0.5720
Epoch 12/30
24/24 [==============================] - 2s 97ms/step - loss: 0.0050 - accuracy: 0.9960 -
val_loss: 3.1600 - val_accuracy: 0.5800
Epoch 13/30
24/24 [==============================] - 2s 95ms/step - loss: -0.0016 - accuracy: 0.9973
- val_loss: 3.3253 - val_accuracy: 0.5760
Epoch 14/30
24/24 [==============================] - 2s 99ms/step - loss: 0.0065 - accuracy: 0.9973 -
val_loss: 3.3197 - val_accuracy: 0.5720
Epoch 15/30
24/24 [==============================] - 2s 98ms/step - loss: -0.0022 - accuracy: 0.9973
- val_loss: 3.4390 - val_accuracy: 0.5680
Epoch 16/30
24/24 [==============================] - 2s 99ms/step - loss: 0.0040 - accuracy: 0.9973 -
val_loss: 3.4010 - val_accuracy: 0.5720
Epoch 17/30
24/24 [==============================] - 2s 97ms/step - loss: -0.0033 - accuracy: 0.9973
- val_loss: 3.4443 - val_accuracy: 0.5680
Epoch 18/30
24/24 [==============================] - 2s 99ms/step - loss: 0.0028 - accuracy: 0.9973 -
val_loss: 3.5556 - val_accuracy: 0.5680
Epoch 19/30
24/24 [==============================] - 2s 101ms/step - loss: -0.0021 - accuracy: 0.9973
- val_loss: 3.5127 - val_accuracy: 0.5680
Epoch 20/30
24/24 [==============================] - 2s 102ms/step - loss: 0.0013 - accuracy: 0.9973
- val_loss: 3.5711 - val_accuracy: 0.5680
Epoch 21/30
24/24 [==============================] - 2s 104ms/step - loss: 0.0058 - accuracy: 0.9973
- val_loss: 3.4992 - val_accuracy: 0.5680
Epoch 22/30
24/24 [==============================] - 3s 108ms/step - loss: 3.2881e-04 - accuracy: 0.9
973 - val_loss: 3.4769 - val_accuracy: 0.5680
Epoch 23/30
24/24 [==============================] - 3s 106ms/step - loss: -0.0033 - accuracy: 0.9973
- val_loss: 3.7398 - val_accuracy: 0.5680
Epoch 24/30
24/24 [==============================] - 2s 103ms/step - loss: 0.0033 - accuracy: 0.9973
- val_loss: 3.7829 - val_accuracy: 0.5640
Epoch 25/30
24/24 [==============================] - 2s 99ms/step - loss: 2.7001e-04 - accuracy: 0.99
73 - val_loss: 3.8491 - val_accuracy: 0.5640
Epoch 26/30
24/24 [==============================] - 2s 101ms/step - loss: -0.0029 - accuracy: 0.9973
- val_loss: 3.8951 - val_accuracy: 0.5640
Epoch 27/30
24/24 [==============================] - 2s 99ms/step - loss: 0.0013 - accuracy: 0.9973 -
val_loss: 4.0327 - val_accuracy: 0.5640
Epoch 28/30
24/24 [==============================] - 2s 98ms/step - loss: -0.0013 - accuracy: 0.9973
- val_loss: 3.9757 - val_accuracy: 0.5640
Epoch 29/30
24/24 [==============================] - 2s 101ms/step - loss: 0.0045 - accuracy: 0.9973
```

```
24/24 [==============================] - 2s 101ms/step - loss: -0.0045 - accuracy: 0.9973
- val_loss: 4.0836 - val_accuracy: 0.5640
Epoch 30/30
24/24 [==============================] - 3s 107ms/step - loss: -0.0070 - accuracy: 0.9973
- val_loss: 4.1424 - val_accuracy: 0.5640
```

In [174]:

```
plot_graphs(history_bilstm_4, "accuracy")
plot_graphs(history_bilstm_4, "loss")
```



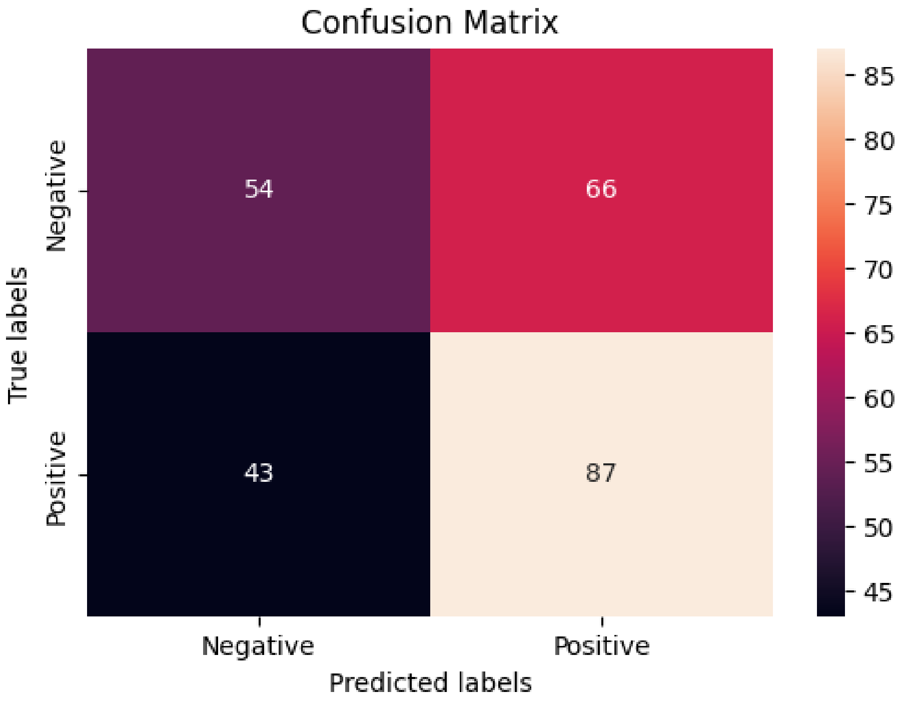


In [175]:

```
# Predict labels for testing data
predicted_labels = (model_bilstm_4.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
```

```
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 3s 35ms/step
Accuracy: 0.564
Precision: 0.5686274509803921
Recall: 0.6692307692307692
F1-score: 0.6148409893992932
```



QUESTION 2

Model with Word2Vec

In [260]:

```
import gensim
from gensim.models import Word2Vec

tokenized_sentences = [sentence.split() for sentence in processed_sentences]

# Train Word2Vec model
word2vec_model = Word2Vec(sentences=tokenized_sentences, vector_size=10, window=3, min_c
ount=1)

# Get word embeddings
word_embeddings = word2vec_model.wv
```

In [261]:

```
# Function to convert sentences to averaged word vectors
def sentence_to_avg_vector(sentence, model):
    tokens = sentence.split()
    word_vectors = []
    for token in tokens:
        if token in model.wv:
            word_vectors.append(model.wv[token])
    if len(word_vectors) > 0:
        return np.mean(word_vectors, axis=0)
    else:
        return np.zeros(model.vector_size)

train_vectors = [sentence_to_avg_vector(sentence, word2vec_model) for sentence in trainin
g_sentences]
test_vectors = [sentence_to_avg_vector(sentence, word2vec_model) for sentence in testing_
sentences]
```

In [262]:

```
training_padded = pad_sequences(train_vectors, maxlen=max_pad_len, padding=padding_type,
truncating=trunc_type)
testing_padded = pad_sequences(test_vectors, maxlen=max_pad_len, padding=padding_type, t
runcating=trunc_type)
```

In [264]:

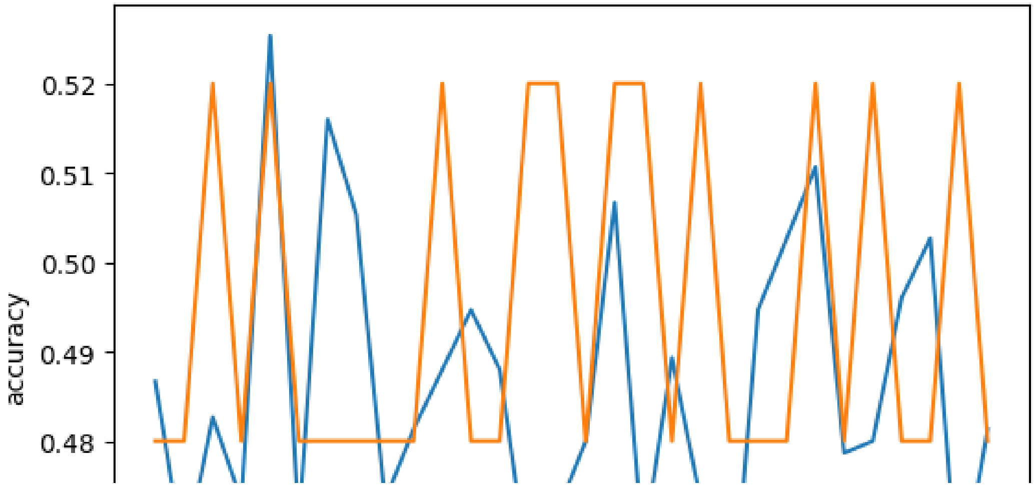
```
num_epochs = 30
history_word2vec = model_gru_1.fit(training_padded, training_labels_final, epochs=num_ep
ochs, validation_data=(testing_padded, testing_labels_final))
```

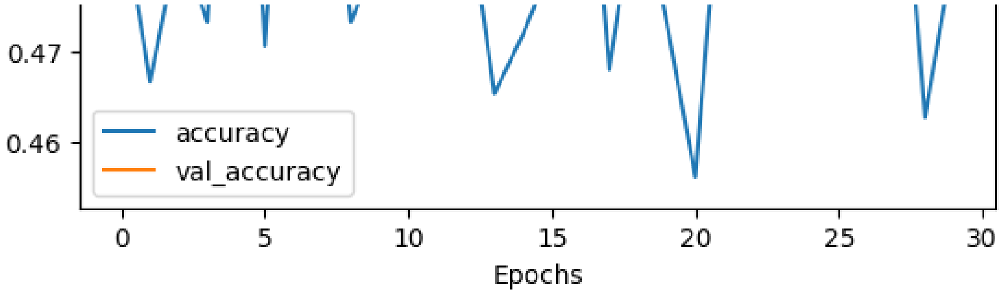
```
Epoch 1/30
24/24 [==============================] - 2s 64ms/step - loss: 0.6945 - accuracy: 0.4867 -
val_loss: 0.6936 - val_accuracy: 0.4800
Epoch 2/30
24/24 [==============================] - 1s 49ms/step - loss: 0.6952 - accuracy: 0.4667 -
val_loss: 0.6940 - val_accuracy: 0.4800
Epoch 3/30
24/24 [==============================] - 1s 49ms/step - loss: 0.6944 - accuracy: 0.4827 -
val_loss: 0.6929 - val_accuracy: 0.5200
Epoch 4/30
24/24 [==============================] - 1s 49ms/step - loss: 0.6942 - accuracy: 0.4733 -
val_loss: 0.6937 - val_accuracy: 0.4800
Epoch 5/30
24/24 [==============================] - 1s 46ms/step - loss: 0.6924 - accuracy: 0.5253 -
val_loss: 0.6929 - val_accuracy: 0.5200
Epoch 6/30
24/24 [==============================] - 1s 44ms/step - loss: 0.6942 - accuracy: 0.4707 -
val_loss: 0.6941 - val_accuracy: 0.4800
Epoch 7/30
24/24 [==============================] - 1s 37ms/step - loss: 0.6938 - accuracy: 0.5160 -
val_loss: 0.6938 - val_accuracy: 0.4800
Epoch 8/30
24/24 [==============================] - 1s 40ms/step - loss: 0.6923 - accuracy: 0.5053 -
val_loss: 0.6933 - val_accuracy: 0.4800
Epoch 9/30
24/24 [==============================] - 1s 43ms/step - loss: 0.6967 - accuracy: 0.4733 -
val_loss: 0.6937 - val_accuracy: 0.4800
Epoch 10/30
24/24 [==============================] - 1s 40ms/step - loss: 0.6957 - accuracy: 0.4813 -
val_loss: 0.6942 - val_accuracy: 0.4800
Epoch 11/30
24/24 [==============================] - 1s 42ms/step - loss: 0.6939 - accuracy: 0.4880 -
val_loss: 0.6927 - val_accuracy: 0.5200
Epoch 12/30
24/24 [==============================] - 1s 47ms/step - loss: 0.6938 - accuracy: 0.4947 -
val_loss: 0.6932 - val_accuracy: 0.4800
Epoch 13/30
24/24 [==============================] - 1s 47ms/step - loss: 0.6938 - accuracy: 0.4880 -
val_loss: 0.6934 - val_accuracy: 0.4800
Epoch 14/30
```

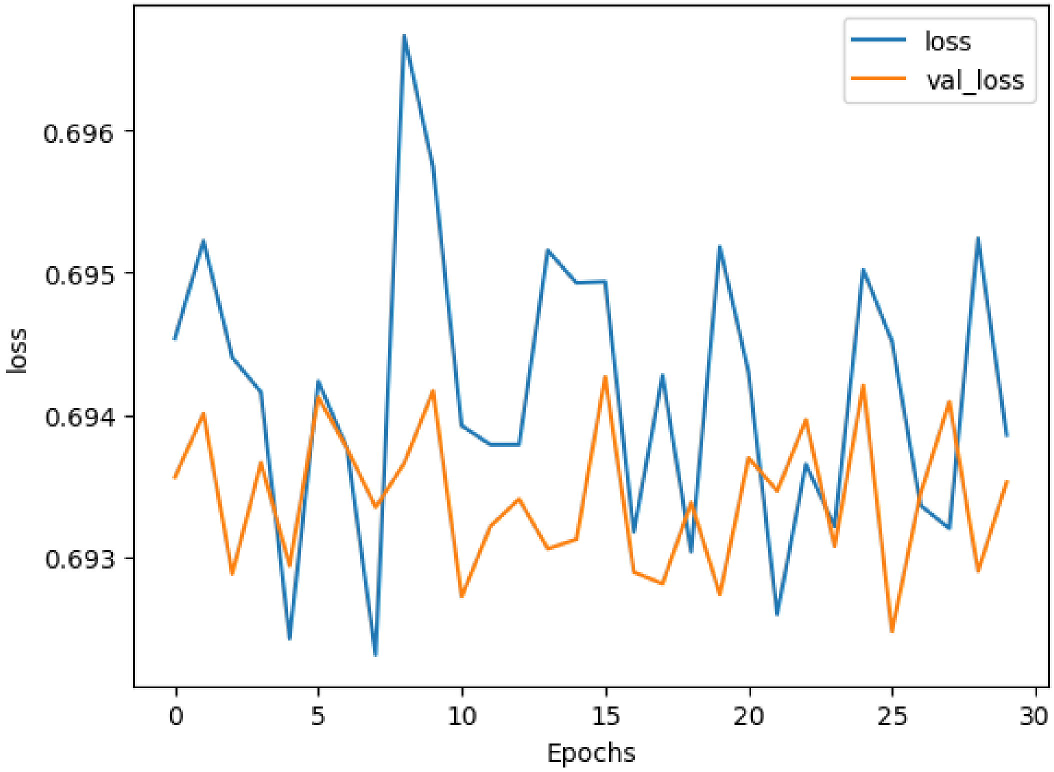
```
24/24 [==============================] - 1s 44ms/step - loss: 0.6952 - accuracy: 0.4653 -
val_loss: 0.6931 - val_accuracy: 0.5200
Epoch 15/30
24/24 [==============================] - 1s 46ms/step - loss: 0.6949 - accuracy: 0.4720 -
val_loss: 0.6931 - val_accuracy: 0.5200
Epoch 16/30
24/24 [==============================] - 1s 42ms/step - loss: 0.6949 - accuracy: 0.4800 -
val_loss: 0.6943 - val_accuracy: 0.4800
Epoch 17/30
24/24 [==============================] - 1s 44ms/step - loss: 0.6932 - accuracy: 0.5067 -
val_loss: 0.6929 - val_accuracy: 0.5200
Epoch 18/30
24/24 [==============================] - 1s 45ms/step - loss: 0.6943 - accuracy: 0.4680 -
val_loss: 0.6928 - val_accuracy: 0.5200
Epoch 19/30
24/24 [==============================] - 1s 46ms/step - loss: 0.6930 - accuracy: 0.4893 -
val_loss: 0.6934 - val_accuracy: 0.4800
Epoch 20/30
24/24 [==============================] - 1s 49ms/step - loss: 0.6952 - accuracy: 0.4733 -
val_loss: 0.6927 - val_accuracy: 0.5200
Epoch 21/30
24/24 [==============================] - 1s 47ms/step - loss: 0.6943 - accuracy: 0.4560 -
val_loss: 0.6937 - val_accuracy: 0.4800
Epoch 22/30
24/24 [==============================] - 1s 40ms/step - loss: 0.6926 - accuracy: 0.4947 -
val_loss: 0.6935 - val_accuracy: 0.4800
Epoch 23/30
24/24 [==============================] - 1s 43ms/step - loss: 0.6937 - accuracy: 0.5027 -
val_loss: 0.6940 - val_accuracy: 0.4800
Epoch 24/30
24/24 [==============================] - 1s 43ms/step - loss: 0.6932 - accuracy: 0.5107 -
val_loss: 0.6931 - val_accuracy: 0.5200
Epoch 25/30
24/24 [==============================] - 1s 41ms/step - loss: 0.6950 - accuracy: 0.4787 -
val_loss: 0.6942 - val_accuracy: 0.4800
Epoch 26/30
24/24 [==============================] - 1s 41ms/step - loss: 0.6945 - accuracy: 0.4800 -
val_loss: 0.6925 - val_accuracy: 0.5200
Epoch 27/30
24/24 [==============================] - 1s 41ms/step - loss: 0.6934 - accuracy: 0.4960 -
val_loss: 0.6934 - val_accuracy: 0.4800
Epoch 28/30
24/24 [==============================] - 1s 38ms/step - loss: 0.6932 - accuracy: 0.5027 -
val_loss: 0.6941 - val_accuracy: 0.4800
Epoch 29/30
24/24 [==============================] - 1s 42ms/step - loss: 0.6952 - accuracy: 0.4627 -
val_loss: 0.6929 - val_accuracy: 0.5200
Epoch 30/30
24/24 [==============================] - 1s 42ms/step - loss: 0.6939 - accuracy: 0.4813 -
val_loss: 0.6935 - val_accuracy: 0.4800
```

In [265]:

```
plot_graphs(history_word2vec, "accuracy")
plot_graphs(history_word2vec, "loss")
```







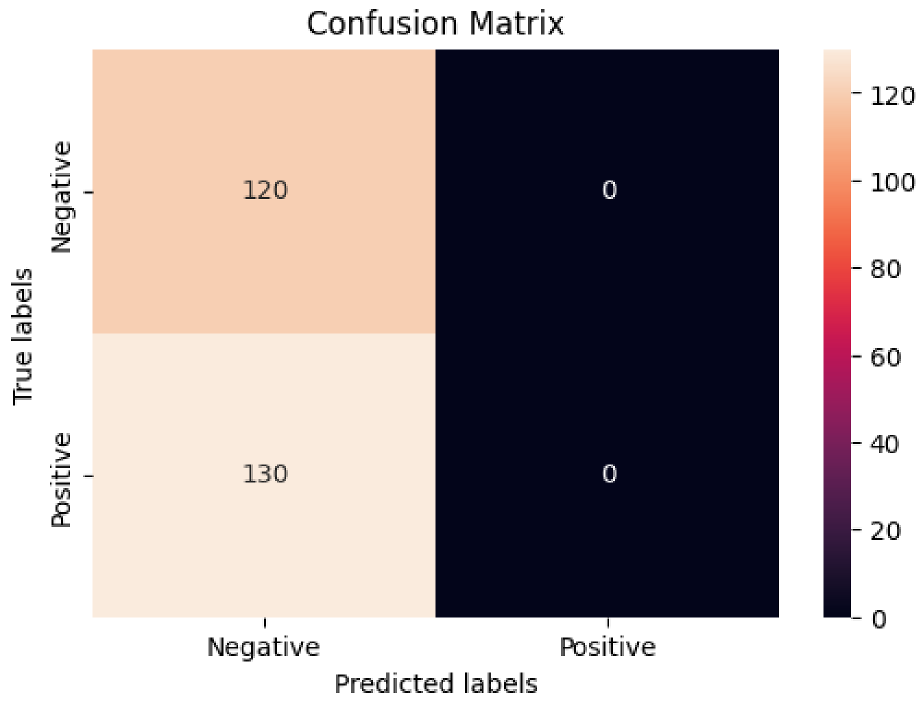
In [267]:

```
# Predict labels for testing data
predicted_labels = (model_gru_1.predict(testing_padded) > 0.5).astype("int32")

# Calculate evaluation metrics
accuracy = accuracy_score(testing_labels_final, predicted_labels)
precision = precision_score(testing_labels_final, predicted_labels)
recall = recall_score(testing_labels_final, predicted_labels)
f1 = f1_score(testing_labels_final, predicted_labels)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

# Plot confusion matrix
cm = confusion_matrix(testing_labels_final, predicted_labels)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=["Negative", "Positive"], yticklabels=[
"Negative", "Positive"])
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```

```
8/8 [==============================] - 0s 20ms/step
Accuracy: 0.48
Precision: 0.0
Recall: 0.0
F1-score: 0.0
```

```
c:\Users\Moazzam Umer\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\m
etrics\_classification.py:1334: UndefinedMetricWarning: Precision is ill-defined and bein
```

```
g set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



Model with Glove

In [ ]:

```
import gensim.downloader as api
from sklearn.model_selection import train_test_split

# Load pre-trained GloVe embeddings
glove_model = api.load("glove-twitter-100")

# Get the vocabulary size
vocab_size = len(glove_model.key_to_index)

# Convert tweets to GloVe embeddings
X_glove = np.array([np.mean([glove_model[word] for word in text.split() if word in glove_model.key_to_index] or [np.zeros(100)], axis=0) for text in processed_sentences])

# Split data into train and test sets
X_train_glove, X_test_glove, y_train_glove, y_test_glove = train_test_split(X_glove, labels, test_size=0.25, random_state=42)
```

In [ ]:

```
num_epochs = 30
history_word2vec = model_gru_1.fit(X_train_glove, y_train_glove, epochs=num_epochs, validation_split=0.2)
```

Model with Fasttext

In [ ]:

```
import gensim.downloader as api

# Load pre-trained FastText embeddings
fasttext_model = api.load("fasttext-wiki-news-subwords-300")

# Get the vocabulary size
vocab_size = len(fasttext_model.key_to_index)
```

```
# Convert tweets to FastText embeddings
X_fasttext = np.array([np.mean([fasttext_model[word] for word in text.split() if word in
fasttext_model.key_to_index] or [np.zeros(300)], axis=0) for text in processed_sentences
])

# Split data into train and test sets
X_train_fasttext, X_test_fasttext, y_train_fasttext, y_test_fasttext = train_test_split(
X_fasttext, labels, test_size=0.25, random_state=42)
```

In [ ]:

```
num_epochs = 30
history_glove = model_gru_1.fit(X_train_fasttext, y_train_fasttext, epochs=num_epochs, v
alidation_split=0.2)
```