In [14]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import re
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
```

In [15]:
```
df = pd.read_csv(r"C:\Users\admin\Desktop\sentiment_analysis_1.csv")
```

In [16]:
```
def clean_text(text):
    return re.sub(r'[^a-zA-Z\s]', '', text.lower()).strip()

df["cleaned_text"] = df["text"].apply(clean_text)
```

In [17]:
```
df.head(5)
```

Out[17]:

	text	label	cleaned_text
0	I love this product! It's fantastic.	positive	i love this product its fantastic
1	The experience was horrible and terrible.	negative	the experience was horrible and terrible
2	It's okay, nothing special.	neutral	its okay nothing special
3	Best purchase ever, highly recommend!	positive	best purchase ever highly recommend
4	Waste of money, very disappointed.	negative	waste of money very disappointed

In [18]:
```
tokenizer = Tokenizer(num_words=10000, oov_token="<OOV>")
tokenizer.fit_on_texts(df["cleaned_text"])
X = pad_sequences(tokenizer.texts_to_sequences(df["cleaned_text"]), maxlen=200)
df["encoded_label"] = LabelEncoder().fit_transform(df["label"])
y = to_categorical(df["encoded_label"], num_classes=3)
```

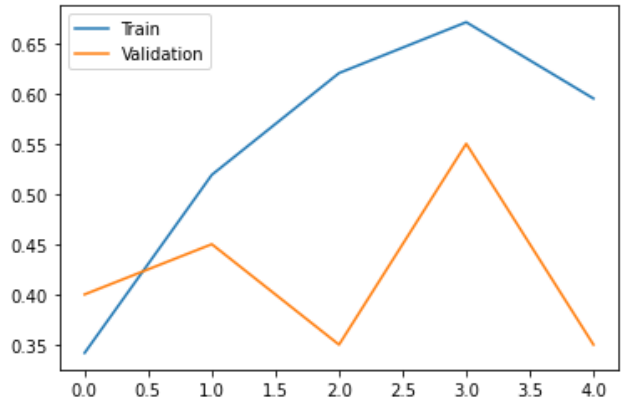
In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [20]:
```
model = Sequential([
    Embedding(10000, 128, input_length=200),
    SimpleRNN(64),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')
])
```

In [22]:
```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
history = model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))
```

```
Epoch 1/5
2/2 [==============================] - 3s 242ms/step - loss: 1.1233 - accuracy: 0.3418 - val_loss:
1.0804 - val_accuracy: 0.4000
Epoch 2/5
2/2 [==============================] - 0s 49ms/step - loss: 1.0599 - accuracy: 0.5190 - val_loss:
1.0845 - val_accuracy: 0.4500
Epoch 3/5
2/2 [==============================] - 0s 50ms/step - loss: 1.0146 - accuracy: 0.6203 - val_loss:
1.0710 - val_accuracy: 0.3500
Epoch 4/5
2/2 [==============================] - 0s 51ms/step - loss: 0.9732 - accuracy: 0.6709 - val_loss:
1.0530 - val_accuracy: 0.5500
Epoch 5/5
2/2 [==============================] - 0s 50ms/step - loss: 0.9539 - accuracy: 0.5949 - val_loss:
1.1110 - val_accuracy: 0.3500
```

In [23]:
```
plt.plot(history.history['accuracy'], label='Train')
plt.plot(history.history['val_accuracy'], label='Validation')
plt.legend()
plt.show()
```



In [24]:
```
label_encoder = LabelEncoder()
df["encoded_label"] = label_encoder.fit_transform(df["label"])
y = to_categorical(df["encoded_label"], num_classes=3)
```

In [25]:
```
def predict_sentiment(text):
    seq = pad_sequences(tokenizer.texts_to_sequences([clean_text(text)]), maxlen=200)
    prediction = model.predict(seq)[0]
    return label_encoder.inverse_transform([np.argmax(prediction)])[0]
```

In [26]:
```
print(predict_sentiment("The product was amazing! I loved it."))
print(predict_sentiment("It was okay, nothing special."))
print(predict_sentiment("Very poor craftsmanship, fell apart."))
```

```
1/1 [==============================] - 0s 181ms/step
positive
1/1 [==============================] - 0s 26ms/step
neutral
1/1 [==============================] - 0s 17ms/step
positive
```

In [ ]:
```

```