In [42]:
```
# Importing required libraries
import nltk
import pandas as pd
from nltk.corpus import stopwords
from textblob import Word
from sklearn.preprocessing import LabelEncoder
from collections import Counter
import wordcloud
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from keras.models import Sequential
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
```

In [43]:
```
#Loading the dataset
data = pd.read_csv( 'amazon_alexaa.tsv',sep ='\t')
#sep='\t' because of seperation by tab

data =data.dropna()
data.isnull().sum()
```

Out[43]:
```
rating              0
date                0
variation           0
verified_reviews    0
feedback            0
dtype: int64
```

In [44]:
```
# Creating a new column sentiment based on overall ratings
def Sentiment(df):
    if df['rating']>3.0:
        return "POSITIVE"
    elif df['rating']==3.0:
        return "Neutral"
    else:
        return "Negative"

data['Sentiment']=data.apply(Sentiment,axis=1)
```

In [81]:
```
data.head(10)
```

Out[81]:

	rating	date	variation	verified_reviews	feedback	Sentiment
0	5	31-Jul-18	Charcoal Fabric	Love my Echo!	1	POSITIVE
1	5	31-Jul-18	Charcoal Fabric	Loved it!	1	POSITIVE
2	4	31-Jul-18	Walnut Finish	Sometimes while playing a game, you can answer...	1	POSITIVE
3	5	31-Jul-18	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1	POSITIVE
4	5	31-Jul-18	Charcoal Fabric	Music	1	POSITIVE
5	5	31-Jul-18	Heather Gray Fabric	I received the echo as a gift. I needed anothe...	1	POSITIVE
6	3	31-Jul-18	Sandstone Fabric	Without having a cellphone, I cannot use many ...	1	Negative
7	5	31-Jul-18	Charcoal Fabric	I think this is the 5th one I've purchased. I'...	1	POSITIVE
8	5	30-Jul-18	Heather Gray Fabric	looks great	1	POSITIVE
9	5	30-Jul-18	Heather Gray Fabric	Love it! I've listened to songs I haven't hear...	1	POSITIVE

In [102]:
```
data_v1=data[["verified_reviews","Sentiment"]]
# Dropping specific columns
data_v1_clean = data_v1.dropna()
```

In [122]:
```
data_v1_clean.isnull().sum()
```

Out[122]:
```
verified_reviews    0
Sentiment           0
dtype: int64
```

Data Cleaning Process

In [115]:
```
def cleaning(df, stop_word):
    # Convert to lowercase
    df['verified_reviews'] = df['verified_reviews'].astype(str).str.lower()

    # Remove digits
    df['verified_reviews'] = df['verified_reviews'].str.replace(r'\d+', '', regex=True)

    # Remove stop words
    df['verified_reviews'] = df['verified_reviews'].apply(lambda x: ' '.join(word for word in x.split() if word not in stop_word))

    # Lemmatization
    df['verified_reviews'] = df['verified_reviews'].apply(lambda x: ' '.join([Word(word).lemmatize() for word in x.split()]))

    return df
```

x.split(): This splits the string x into a list of words. By default, the split method uses whitespace as the delimiter.

if x not in stop_word: This is a condition that checks if each word is not in the stop_word list. The stop_word variable should be a predefined list of stop words that you want to remove.

' '.join(...): This joins the list of words back into a single string with a space as the delimiter, effectively reconstructing the sentence without the stop words.

In [118]:
```
stop_word=stopwords.words('english')
data_v1_clean=cleaning(data_v1_clean,stop_word)
```

In [119]:
```
data_v1_clean
```

Out[119]:

	verified_reviews	Sentiment
0	love echo!	POSITIVE
1	loved it!	POSITIVE
2	sometimes playing game, answer question correc...	POSITIVE
3	lot fun thing. yr old learns dinosaurs, contro...	POSITIVE
4	music	POSITIVE
...	...	...
3145	perfect kids, adult everyone between!	POSITIVE
3146	listening music, searching locations, checking...	POSITIVE
3147	love things, running entire home, tv, lights, ...	POSITIVE
3148	complaint sound quality great, mostly use comm...	POSITIVE
3149	good	POSITIVE

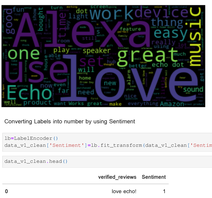
3149 rows × 2 columns

In [120]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [124]:
```
common_words = ""

for i in data_v1['verified_reviews']:
    tokens = i.split()
    common_words += ' '.join(tokens) + ' '

wordcloud = WordCloud().generate(common_words)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



Converting Labels into number by using Sentiment

In [127]:
```
lb=LabelEncoder()
data_v1_clean['Sentiment']=lb.fit_transform(data_v1_clean['Sentiment'])
```

In [35]:
```
data_v1_clean.head()
```

Out[35]:

	verified_reviews	Sentiment
0	love echo!	1
1	loved it!	1
2	sometimes playing game, answer question correc...	1
3	lot fun thing. yr old learns dinosaurs, contro...	1
4	music	1

In [37]:
```
# Tokenizer configuration
tokenizer = Tokenizer(num_words=500, split=' ')

# It will convert words into their corresponding unique integer index values,
#limited to the top 500 words specified earlier.
tokenizer.fit_on_texts(data_v1_clean['verified_reviews'].values)

# Convert texts to sequences
X = tokenizer.texts_to_sequences(data['verified_reviews'].values)
X=pad_sequences(X)
X
```

Out[37]:
```
array([[  0,   0,   0, ...,   0,   1,   2],
       [  0,   0,   0, ...,   0, 187,   5],
       [  0,   0,   0, ..., 253, 263,  24],
       ...,
       [  0,   0,   0, ..., 120, 146,  30],
       [  0,   0,   0, ..., 120,  54, 278],
       [  0,   0,   0, ...,   0,   0, 171]])
```

In [40]:
```
model= Sequential()
model.add(Embedding(500,120,input_length=X.shape[1]))
model.add(SpatialDropout1D(0.4))
model.add(LSTM(176, dropout=0.2,recurrent_dropout=0.2))
model.add(Dense(2,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
C:\Users\patil\anaconda3\lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
spatial_dropout1d (SpatialDropout1D)	?	0
lstm (LSTM)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)

```
Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)
None
```

In [ ]:
```

```

1.model.add(Embedding(500,120,input_length=X.shape[1])) 120: This is the dimension of the dense embedding. Each word will be represented as a 120-dimensional vector.

2.model.add(SpatialDropout1D(0.4)) The SpatialDropout1D layer is a regularization technique used in neural networks to help prevent overfitting.

Here's a bit more detail on why and how it's used: @Overfitting: One of the common problems in training neural networks is overfitting, where the model performs well on training data but poorly on unseen test data. Dropout helps mitigate this by making the model less sensitive to specific weights of neurons.

@Noise reduction: By zeroing out some of the input features during training, dropout effectively introduces noise into the learning process. This noise forces the network to learn more generally useful patterns rather than memorizing the training data.

3.dropout=0.2 The dropout rate for the input connections in the layer. Dropout is a regularization technique to prevent overfitting in neural networks. In this context, it randomly drops 20% of the input neurons during each training iteration, helping the model generalize better.

4.what is recurrent_dropout The recurrent_dropout parameter is another form of dropout specific to recurrent layers like LSTM. While dropout applies to the input connections (what's fed into the LSTM), recurrent_dropout applies to the hidden state connections (within the LSTM cells).

In [69]:
```
#Splitting the data into training and testing
y=pd.get_dummies(data_v1_clean['Sentiment'])
X_train, X_test , y_train, y_test =train_test_split(X,y,test_size=0.3,random_state=42)
```

In [62]:
```
X_train= np.array(X_train)
X_test=np.array(X_test)
y_train=np.array(y_train)
y_test=np . array (y_test)
```

In [ ]:
```

```

why convert into arrays ? Bcoz we need to create a particular form so that the model should understand

In [76]:
```
batch_size=32
history=model.fit(X_train, y_train, epochs =10, batch_size=batch_size, verbose = 1)
```

```
Epoch 1/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 18s 255ms/step - accuracy: 0.9757 - loss: 0.0749
Epoch 2/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 19s 275ms/step - accuracy: 0.9689 - loss: 0.0795
Epoch 3/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 20s 290ms/step - accuracy: 0.9515 - loss: 0.1169
Epoch 4/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 21s 296ms/step - accuracy: 0.9735 - loss: 0.0729
Epoch 5/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 20s 293ms/step - accuracy: 0.9654 - loss: 0.0767
Epoch 6/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 20s 289ms/step - accuracy: 0.9758 - loss: 0.0676
Epoch 7/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 20s 293ms/step - accuracy: 0.9681 - loss: 0.0690
Epoch 8/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 20s 294ms/step - accuracy: 0.9782 - loss: 0.0673
Epoch 9/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 20s 289ms/step - accuracy: 0.9771 - loss: 0.0573
Epoch 10/10
69/69 ━━━━━━━━━━━━━━━━━━━━ 20s 290ms/step - accuracy: 0.9821 - loss: 0.0503
```

In [78]:
```
model. evaluate (X_test, y_test )
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 68ms/step - accuracy: 0.9019 - loss: 0.4454
```

Out[78]:
```
[0.4220684806346461, 0.9037036895751953]
```

In [80]:
```
model.save("Sentiment_Analysis_MLP.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

In [92]:
```
# Convert the training history to a DataFrame
history_dict = history.history
history_df = pd.DataFrame(history_dict)

# Display the DataFrame
print(history_df)
```

```
   accuracy      loss
0  0.964156  0.094625
1  0.961434  0.092069
2  0.965517  0.090105
3  0.968681  0.091467
4  0.968683  0.074642
5  0.975953  0.069230
6  0.970508  0.070414
7  0.977768  0.063154
8  0.976407  0.057400
9  0.977768  0.056600
```

In [104]:
```
# Plotting accuracy and loss
plt.plot(history.history['accuracy'], label='Accuracy')
plt.plot(history.history['loss'], label='Loss')
plt.xlabel('Epochs')
plt.ylabel('Value')
plt.title('Model Training History')
plt.legend()
plt.show()
```

In [ ]:
```

```