Twitter sentiment natural language processing analysis

Stakeholder: A marketing team at a technology company specifically Google

Project Overview

The main aim of this project is to analyse the tweet-product-dataset, obtained from Twitter . Through this analysis, we are to achieve the following:

- Identify some of the sentiments associated with the tweets of the various products.
- Process the data and analyse it to determine its key characteristics.
- Come up with a model that accurately evaluates and predicts the sentiments towards given product lines

Business Overview

Let’s understand a Sentiment Analysis problem from a business standpoint. The good news is: with the power of the internet, businesses today get a huge number of customer feedback through their business website, social media page, business listings, etc. However, the bad news is: a majority of businesses do not even know how to use this information to improve themselves.

Business Problem Statement:

The marketing team wants to understand consumer sentiment towards Apple and Google products. They wish to explore emotions, and sentiments expressed by users of Apple and Google products.They need insights that can inform marketing strategies, brand perception, and customer satisfaction initiatives for their target audience.

Currently the marketing team collects sentiments manually by having a team to classify each tweet as either positive, negative or neutral. This is both expensive, and takes a lot of time. It also means the team doesn’t get real-time information to make quick and strategic course-corrections

The answer to the question of manual sentiment analysis is automation. Automating the process of sentiment analysis will also ensure that the team is able to get real-time sentiment insights -e.g. on a biweekly basis:

- Knowing the number and actual positive sentiment tweets can enable the team to reinforce these positive attributes in advertising via consumer testimonials. This will serve to boost brand credibility and boost sales among new customers
- Knowing the number and actual negative sentiment tweets will enable the team to know what product improvements to make, and to reach out to consumers with the issues to have them sorted out. This will reduce customer churn

The main business objective is:

How do we use available data to create a model that will be able to classify future tweets with the correct sentiments?

- This will save the team both time and money by moving away from manual process
- This will also help grow loyalty and sales, as articulated above in the problem statment

The specific business objectives to be answered include:

- What is the overall sentiment towards Apple and Google products on Twitter
- Are there any recurring themes or topics associated with positive or negative sentiments towards these brands?
- Are there any notable differences in sentiment between Apple and Google products?

Project success metric

Our project success metric is to have a model with an accuracy of `60%` or above.

Data Understanding

We work with the twitter sentiment dataset obtained from CrowdFlower that involves customers emotions and tweets towards different brands and products .

- The dataset contains 9093 rows and 3 columns
- With the columns being :
- tweet : Has the tweet text and message from customers
- is_there_an_emotion_directed_at_a_brand_or_product : has either positive , negative ,no emotion ,can't tell
- emotion_in_tweet_is_directed_at (brands/product) : that included iphone,ipad,google

Summary of work done

Data cleaning

Our data cleaning process entails

- Identifying and handling missing data
- We dropped some rows missing tweet comments on tweet text column
- The brand column missing some product names, thus crawled through the tweet text rows and extracted brand product mentions and used them to fill the missing values
- We drop 7% of rows which completed missed brand/product mentions, and we couldn’t completely identify any mentions from tweet text columns. We felt that this 7% would not have any significant effect on data quality
- We dropped duplicated data
- We identified some rows with duplicated tweets, matching exactly word for word

Data Preparation

In the data preparation process :-

- Before training the model, we performed various pre-processing steps on the dataset that mainly dealt with removing stopwords, removing special characters like emojis, hashtags, etc. The text document is then converted into lowercase for better generalization.
- Subsequently, the punctuations were cleaned and removed, thereby reducing the unnecessary noise from the dataset. After that, we also removed the repeating characters from the words along with removin the URLs as they do not have any significant importance.
- At last, we then performed Lemmatization(reducing the derived words to their root form, known as lemma) for better results.

List of libraries/packages used:-

- We used the nltk stopwords library to remove stopwords which contains a great collection of English stopwords
- WordLemmatizer- normalize words and reduce them to their canonical form
- word_tokenize - It is a function from the NLTK (Natural Language Toolkit) library used to split a text string into individual words or tokens
- re(regular expressions)- It is a built-in module in Python that provides support for working with regular expressions

Modelling

We evaluated 6 different models in this project.

- Logistic Regression
- Multinomial Naive Bayes
- Decision Trees
- Random Forest
- Gradient Boosting Classifier
- Extreme Gradient Boosting Classifier(XGBoost).

Logistic Regression and Multinomial Naive Bayes are popular choices for their simplicity and ability to handle text data. Decision Trees are intuitive and can capture complex relationships. Random Forest is an ensemble method that improves performance by combining multiple decision trees. Gradient Boosting and XGBoost are powerful algorithms that handle high-dimensional

Evaluation

- Our final model which was a tuned gradient boosting classifier model had an accuracy of 61% which was above our threshold of 60%.
- Validation approach for sentiment analysis model involved a careful division of data into train and test data, training the model, tuning hyperparameters, evaluating performance, and refining the model iteratively to achieve the best possible accuracy in sentiment prediction.

Note on our Next steps

- We found some time to also run another model(a neural network model) -RoBERT(Robustly Optimized BERT approach) is a state-of-the-art natural language processing (NLP) model- on our data this model achieved an accuracy of ~66%
- code for that model is included here- ROBERTA-MODEL (roberta.ipynb)

Importing the necessary packages

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
import string
import re
from nltk import FreqDist

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics import accuracy_score, f1_score, ConfusionMatrixDisplay, confu
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from xgboost import XGBClassifier
import warnings
warnings.filterwarnings(action='ignore', category=FutureWarning)
```

```
[nltk_data] Downloading package punkt to /home/pk/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/pk/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/pk/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/pk/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [2]:

```
df = pd.read_csv('tweet_product_company.csv', encoding='unicode_escape')
df.head()
```

Out[2]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_pro
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative em
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive em
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive em
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative em
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive em

In [3]:

```
df.shape
```

Out[3]:

```
(9093, 3)
```

In [4]:

```
df.is_there_an_emotion_directed_at_a_brand_or_product.value_counts()
```

Out[4]:

```
No emotion toward brand or product    5389
Positive emotion                      2978
Negative emotion                       570
I can't tell                           156
Name: is_there_an_emotion_directed_at_a_brand_or_product, dtype: int64
```

In [5]:

```
df.rename(columns ={'is_there_an_emotion_directed_at_a_brand_or_product' : 'sentime
```

In [6]:

```
df['sentiment'] = df['sentiment'].replace('No emotion toward brand or product', 'ne
```

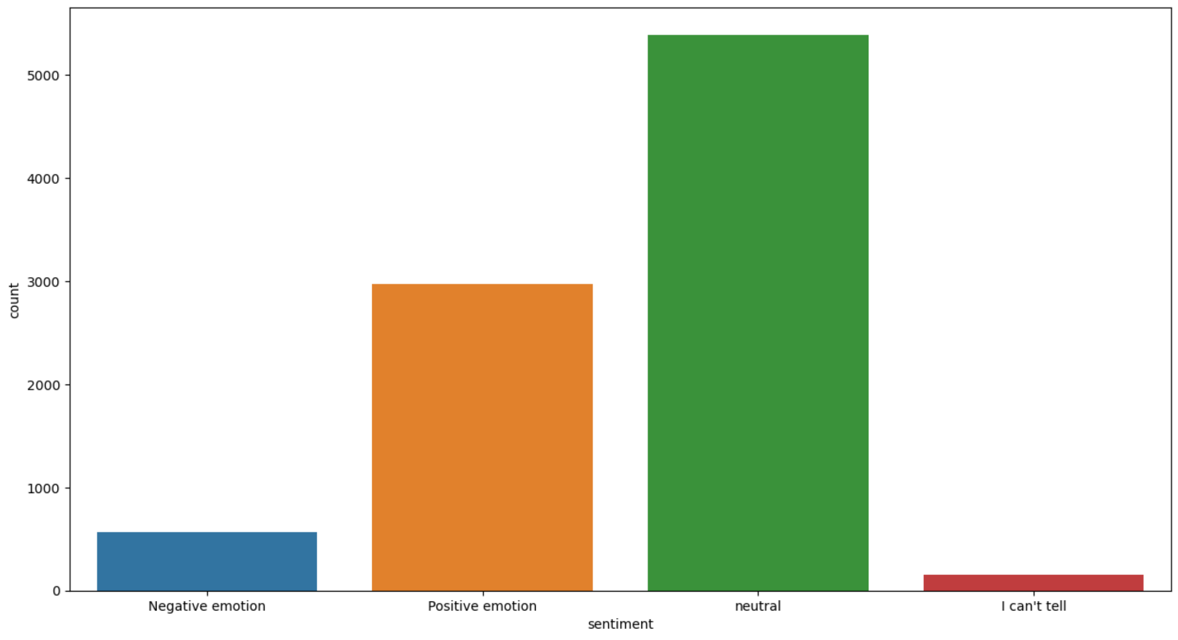
Let us visualize the emotions count

In [7]:

```
plt.figure(figsize = (15,8))
sns.countplot(x='sentiment', data= df)
```

Out[7]:

```
<AxesSubplot:xlabel='sentiment', ylabel='count'>
```



```
Most cutomers are seen to have No emotion toward brand or product followed by
those that have positive emotions/reviews towards the products
```

Data Cleaning

Identifying & Handling the missing values

In [8]:

```
df.isna().sum()
```

Out[8]:

```
tweet_text       1
brand         5802
sentiment        0
dtype: int64
```

In [9]:

```
df['tweet_text'] = df['tweet_text'].str.lower()
df['brand'] = df['brand'].str.lower()
```

In [10]:

```
missing_brand_df = df[df['brand'].isna()]
missing_brand_df.head(10)
```

Out[10]:

	tweet_text	brand	sentiment
5	@teachntech00 new ipad apps for #speechtherapy...	NaN	neutral
6	NaN	NaN	neutral
16	holler gram for ipad on the itunes app store -...	NaN	neutral
32	attn: all #sxsw frineds, @mention register fo...	NaN	neutral
33	anyone at #sxsw want to sell their old ipad?	NaN	neutral
34	anyone at #sxsw who bought the new ipad want ...	NaN	neutral
35	at #sxsw. oooh. rt @mention google to launch ...	NaN	neutral
37	spin play - a new concept in music discovery f...	NaN	neutral
39	vatornews - google and apple force print media...	NaN	neutral
41	hootsuite - hootsuite mobile for #sxsw ~ updat...	NaN	neutral

In [11]:

```
missing_brand_df.shape
```

Out[11]:

```
(5802, 3)
```

In [12]:

```
df.brand.value_counts()
```

Out[12]:

```
ipad                               946
apple                              661
ipad or iphone app                 470
google                             430
iphone                             297
other google product or service    293
android app                         81
android                             78
other apple product or service      35
Name: brand, dtype: int64
```

In [13]:

```
# Replace missing values with an empty string
# df['tweet_text'] = df['tweet_text'].fillna('')
df.dropna(subset=['tweet_text'], inplace=True)
missing_brand_df.dropna(subset=['tweet_text'], inplace=True)

# Define the keywords to search for
keywords = ['apple', 'ipad or iphone app', 'google', 'iphone', 'other google produc

# Search for keywords in 'tweet_text' column
filtered_df = missing_brand_df[missing_brand_df['tweet_text'].str.contains('|'.join
```

```
/tmp/ipykernel_5015/4002662394.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  missing_brand_df.dropna(subset=['tweet_text'], inplace=True)
```

In [14]:

```
filtered_df.head(10)
```

Out[14]:

	tweet_text	brand	sentiment
5	@teachntech00 new ipad apps for #speechtherapy...	NaN	neutral
16	holler gram for ipad on the itunes app store -...	NaN	neutral
32	attn: all #sxsw frineds, @mention register fo...	NaN	neutral
33	anyone at #sxsw want to sell their old ipad?	NaN	neutral
34	anyone at #sxsw who bought the new ipad want ...	NaN	neutral
35	at #sxsw. oooh. rt @mention google to launch ...	NaN	neutral
37	spin play - a new concept in music discovery f...	NaN	neutral
39	vatornews - google and apple force print media...	NaN	neutral
41	hootsuite - hootsuite mobile for #sxsw ~ updat...	NaN	neutral
42	hey #sxsw - how long do you think it takes us ...	NaN	neutral

In [15]:

```
filtered_df.shape
```

Out[15]:

```
(5095, 3)
```

In [16]:

```
# Assign the keyword value to the 'brand' column
filtered_df['brand'] = filtered_df['tweet_text'].apply(lambda x: next((word for wor
```

```
/tmp/ipykernel_5015/2583434774.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
l#returning-a-view-versus-a-copy)
  filtered_df['brand'] = filtered_df['tweet_text'].apply(lambda x: nex
t((word for word in keywords if word in x), None))
```

In [17]:

```
filtered_df.head(10)
```

Out[17]:

	tweet_text	brand	sentiment
5	@teachntech00 new ipad apps for #speechtherapy...	ipad	neutral
16	holler gram for ipad on the itunes app store -...	ipad	neutral
32	attn: all #sxsw frineds, @mention register fo...	android	neutral
33	anyone at #sxsw want to sell their old ipad?	ipad	neutral
34	anyone at #sxsw who bought the new ipad want ...	ipad	neutral
35	at #sxsw. oooh. rt @mention google to launch ...	google	neutral
37	spin play - a new concept in music discovery f...	ipad	neutral
39	vatornews - google and apple force print media...	apple	neutral
41	hootsuite - hootsuite mobile for #sxsw ~ updat...	iphone	neutral
42	hey #sxsw - how long do you think it takes us ...	iphone	neutral

In [18]:

```
filtered_df.shape
```

Out[18]:

```
(5095, 3)
```

In [19]:

```
filtered_df.isna().sum()
```

Out[19]:

```
tweet_text    0
brand         0
sentiment     0
dtype: int64
```

In [20]:

```
# Convert the relevant columns to sets
missing_brand_set = set(missing_brand_df.index)
filtered_set = set(filtered_df.index)

# Find the rows in missing_brand_df not in filtered_df using set difference
result_set = missing_brand_set - filtered_set

# Create the resulting DataFrame
result_df = df.loc[result_set]

# Display the resulting DataFrame
pd.set_option('display.max_colwidth', None)
result_df.tail(20)
```

Out[20]:

	tweet_text	brand	sentiment
6095	rt @mention if you are attending #sxsw, please bring your old mobile phones for the hope phones project {link} @mention	NaN	neutral
8142	#sxsw #sxswi @mention is holding a secret show ft. mister heavenly+michael cera at 5. free beer at 4. location: {link}	NaN	neutral
8144	day 4: sxswi wrap-up with tracy shea @mention and rick and bobby liebling @mention - {link} #sxsw #hotsheet	NaN	neutral
2007	lonely planet has made the austin guide free for #sxsw grab it, you never know when you will need it. {link}	NaN	neutral
4055	no umbrella when you have &quot;the list&quot; via @mention ❱÷¼ are you all set? ❱÷_ {link} ❱÷_ #edchat #musedchat #sxsw #sxswi #newtwitter	NaN	neutral
8156	#wahoos #sxsw party free beers, tacos, and ritas thurs.\n{link}	NaN	neutral
6110	rt @mention if you're at #sxsw : meet us at fado. we will give you wristbands that magically lead to beer. come say hi! {link}	NaN	neutral
2016	@mention announces #windowsphone app today #sxsw {link} #wp7	NaN	neutral
4065	24 hours later, we retreat back to our hotel, bushwhacked. ringo deathstarr'd, never seen so many effects pedals #sxsw {link}	NaN	neutral
4066	grab the latest @mention before #sxsw! {link}	NaN	neutral
6114	rt @mention if you're racing around #sxsw, you best be fueling up with great local fare\n#eatshopaustinapp {link}	NaN	neutral
2020	austinl fans: rt @mention head over to {link} by 1pm cst today to win vip access to my acoustic solo set at #sxsw tonight	NaN	neutral
6115	rt @mention if you're trying to contact friends or family in #japan, @mention has created a person finder: {link} #sxsw	NaN	neutral
8160	all you #sxsw tweeps, check out our homies @mention this thursday. {link}	NaN	neutral
8164	@mention sxsw. first lecture: why my phone should turn off the stove. http://bit.ly/reword_app\n#sxsw #reword #appstore	NaN	neutral
6120	rt @mention important: make sure you're donating to the japanese red cross for #japan: {link} #sxswcares #sxsw #quake (pls rt)	NaN	neutral
6124	rt @mention in case you missed #sxsw news coverage, we were featured in today's fast company: {link}	NaN	neutral
6125	rt @mention in front of @mention popup store at #sxsw last night {link}	NaN	neutral

	tweet_text	brand	sentiment
6137	rt @mention interesting rt @mention @mention to launch major new social network called circles, possibly today {link} #sxsw	NaN	neutral

In [21]:

```
result_df.shape
```

Out[21]:

```
(706, 3)
```

In [22]:

```
707/9093
```

Out[22]:

```
0.07775211701308699
```

In [23]:

```
df_brand_present = df[df['brand'].notna()]
# Assuming 'df_brand_present' and 'filtered_df' are your DataFrames
concatenated_df = pd.concat([df_brand_present, filtered_df], ignore_index=True)

# Display the concatenated DataFrame
concatenated_df.head(10)
```

Out[23]:

	tweet_text	brand	sentiment
0	.@wesley83 i have a 3g iphone. after 3 hrs tweeting at #rise_austin, it was dead! i need to upgrade. plugin stations at #sxsw.	iphone	Negative emotion
1	@jessedee know about @fludapp ? awesome ipad/iphone app that you'll likely appreciate for its design. also, they're giving free ts at #sxsw	ipad or iphone app	Positive emotion
2	@swonderlin can not wait for #ipad 2 also. they should sale them down at #sxsw.	ipad	Positive emotion
3	@sxsw i hope this year's festival isn't as crashy as this year's iphone app. #sxsw	ipad or iphone app	Negative emotion
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech books/conferences) &amp; matt mullenweg (wordpress)	google	Positive emotion
5	#sxsw is just starting, #ctia is around the corner and #googleio is only a hop skip and a jump from there, good time to be an #android fan	android	Positive emotion
6	beautifully smart and simple idea rt @madebymany @thenextweb wrote about our #hollergram ipad app for #sxsw! http://bit.ly/ieavob	ipad or iphone app	Positive emotion
7	counting down the days to #sxsw plus strong canadian dollar means stock up on apple gear	apple	Positive emotion
8	excited to meet the @samsungmobileus at #sxsw so i can show them my sprint galaxy s still running android 2.1. #fail	android	Positive emotion
9	find &amp; start impromptu parties at #sxsw with @hurricaneparty http://bit.ly/gvlrin i can't wait til the android app comes out.	android app	Positive emotion

In [24]:

```
concatenated_df.shape
```

Out[24]:

```
(8386, 3)
```

In [25]:

```
concatenated_df.isna().sum()
```

Out[25]:

```
tweet_text    0
brand         0
sentiment     0
dtype: int64
```

Handling duplicates

In [26]:

```
duplicates_df = concatenated_df[concatenated_df.duplicated()]
```

In [27]:

```
duplicates_df.head(20)
```

Out[27]:

	tweet_text	brand	sentiment
197	before it even begins, apple wins #sxsw {link}	apple	Positive emotion
198	before it even begins, apple wins #sxsw {link}	apple	Positive emotion
272	if you're in a room full of people w/good wi-fi at #sxsw run #frostwire on your android {link} share pics, apps, vids w/others	android app	Positive emotion
970	counting down the days to #sxsw plus strong canadian dollar means stock up on apple gear	apple	Positive emotion
1445	win free ipad 2 from webdoc.com #sxsw rt	ipad	Positive emotion
1499	really enjoying the changes in gowalla 3.0 for android! looking forward to seeing what else they &amp; foursquare have up their sleeves at #sxsw	android app	Positive emotion
1507	#sxsw is just starting, #ctia is around the corner and #googleio is only a hop skip and a jump from there, good time to be an #android fan	android	Positive emotion
1901	oh. my. god. the #sxsw app for ipad is pure, unadulterated awesome. it's easier to browse events on ipad than on the website!!!	ipad or iphone app	Positive emotion
2322	rt @mention marissa mayer: google will connect the digital &amp; physical worlds through mobile - {link} #sxsw	google	Positive emotion
2323	rt @mention marissa mayer: google will connect the digital &amp; physical worlds through mobile - {link} #sxsw	google	Positive emotion
2324	rt @mention marissa mayer: google will connect the digital &amp; physical worlds through mobile - {link} #sxsw	google	Positive emotion
3074	i just noticed dst is coming this weekend. how many iphone users will be an hour late at sxsw come sunday morning? #sxsw #iphone	iphone	Negative emotion
3162	need to buy an ipad2 while i'm in austin at #sxsw. not sure if i'll need to q up at an austin apple store?	ipad	Positive emotion
3674	google to launch major new social network called circles, possibly today {link} #sxsw	google	neutral
3675	google to launch major new social network called circles, possibly today {link} #sxsw	google	neutral
3693	google to launch major new social network called circles, possibly today at #sxsw {link}	google	neutral
4485	marissa mayer: google will connect the digital &amp; physical worlds through mobile - {link} #sxsw	google	neutral
4486	marissa mayer: google will connect the digital &amp; physical worlds through mobile - {link} #sxsw	google	neutral
5356	win free ipad 2 from webdoc.com #sxsw rt	ipad	neutral
5357	win free ipad 2 from webdoc.com #sxsw rt	ipad	neutral

In [28]:

```
concatenated_df.drop_duplicates(inplace=True)
```

In [29]:

```
concatenated_df.head()
```

Out[29]:

	tweet_text	brand	sentiment
0	.@wesley83 i have a 3g iphone. after 3 hrs tweeting at #rise_austin, it was dead! i need to upgrade. plugin stations at #sxsw.	iphone	Negative emotion
1	@jessedee know about @fludapp ? awesome ipad/iphone app that you'll likely appreciate for its design. also, they're giving free ts at #sxsw	ipad or iphone app	Positive emotion
2	@swonderlin can not wait for #ipad 2 also. they should sale them down at #sxsw.	ipad	Positive emotion
3	@sxsw i hope this year's festival isn't as crashy as this year's iphone app. #sxsw	ipad or iphone app	Negative emotion
4	@sxtxstate great stuff on fri #sxsw: marissa mayer (google), tim o'reilly (tech books/conferences) &amp; matt mullenweg (wordpress)	google	Positive emotion

The most used hashtags

In [30]:

```

def hashtag_count(string):
    # Split the string into words
    words = string.split()
    # Create a list of hashtags
    hashtags = [word for word in words if word.startswith('#')]
    # Return number of hashtags
    return hashtags


```

In [31]:

```
hasht = []
# Loop through each row in the "no_stop" column
for index, row in concatenated_df.iterrows():
    tweet = row['tweet_text']
    hashtag_count(tweet)
    # Remove stop words from the tweet
    # Append the filtered tweet to the list
    hasht.append(hashtag_count(tweet))
```

In [32]:

```
pd.Series(hasht).value_counts().head(20)
```

Out[32]:

```
[#sxsw]                           3840
[#sxsw.]                           242
[#sxsw:]                           102
[#sxsw?]                            86
[#sxsw, #sxswi]                     85
[#sxsw,]                            85
[#sxsw, #ipad2]                     81
[#apple, #sxsw]                     81
[#sxsw!]                            70
[#sxsw❱û  ]                          56
[#google, #sxsw]                    52
[#ipad2, #sxsw]                     42
[#sxsw, #apple]                     37
[#ubersocial, #iphone, #sxsw]       35
[#sxswi, #sxsw]                     32
[#sxsw, #google]                    32
[]                                  31
[#ipad, #sxsw]                      31
[#tapworthy, #sxsw]                 29
[#newsapps, #sxsw]                  27
dtype: int64
```

```
Above is the value counts of the most used hashtag words respectively:
 SXSW- South By South West is an annual conglomeration of tech conferences,
parallel film, interactive media, and music festivals
 ipad
 google
```

Data Preprocessing

We will start by preprocessing the text data. We will follow the steps listed below

- Remove punctuations and urls
- Tokenize (split text into words)
- Remove stopwords
- Lemmatization of words

Step 1: Remove punctuations and urls
We will create 2 functions below to help us in removing the punctuations and urls

In [33]:

```python
# remove punctuations
def remove_punc(text):
    exclude = string.punctuation
    for char in exclude:
        text = text.replace(char,'')
    return text
```

In [34]:

```
# apply the remove punctuation funtion on the tweet_text column
concatenated_df['tweet_text'] = concatenated_df.tweet_text.apply(remove_punc)
concatenated_df.head()
```

Out[34]:

	tweet_text	brand	sentiment
0	wesley83 i have a 3g iphone after 3 hrs tweeting at riseaustin it was dead i need to upgrade plugin stations at sxsw	iphone	Negative emotion
1	jessedee know about fludapp awesome ipadiphone app that youll likely appreciate for its design also theyre giving free ts at sxsw	ipad or iphone app	Positive emotion
2	swonderlin can not wait for ipad 2 also they should sale them down at sxsw	ipad	Positive emotion
3	sxsw i hope this years festival isnt as crashy as this years iphone app sxsw	ipad or iphone app	Negative emotion
4	sxtxstate great stuff on fri sxsw marissa mayer google tim oreilly tech booksconferences amp matt mullenweg wordpress	google	Positive emotion

Step 2 : Removing the URLs

In [35]:

```
#remove urls
def cleaning_URLs(data):
    return re.sub('((www.[^s]+)|(https?://[^s]+))',' ',data)

#apply the function on the dataset
concatenated_df['tweet_text'] = concatenated_df['tweet_text'].apply(lambda x: clean
concatenated_df.tail()
```

Out[35]:

	tweet_text	brand	sentiment
8381	mention yup but i dont have a third app yet im on android any suggestions sxsw cc mention	android	neutral
8382	wave buzz rt mention we interrupt your regularly scheduled sxsw geek programming with big news link google circles	google	neutral
8383	googles zeiger a physician never reported potential ae yet fda relies on physicians quotwere operating wout dataquot sxsw health2dev	google	neutral
8384	some verizon iphone customers complained their time fell back an hour this weekend of course they were the new yorkers who attended sxsw	iphone	neutral
8385	ïï¡×ïàüêîò£áââ£÷ââûârt mention google tests ûïcheckin offersû at sxsw link	google	neutral

Removing Emoticons/ Emojis

In [36]:

```
# remove symbols, emoticons etc
def remove_emoticons(text):
    emoticon_pattern = re.compile(
        "(["
        "\U0001F600-\U0001F64F"  # Emoticons
        "\U0001F300-\U0001F5FF"  # Symbols & Pictographs
        "\U0001F680-\U0001F6FF"  # Transport & Map Symbols
        "\U0001F1E0-\U0001F1FF"  # Flags (iOS)
        "])"
    )
    return emoticon_pattern.sub(r"", text)

concatenated_df['tweet_text'] = concatenated_df['tweet_text'].apply(lambda x: remov
concatenated_df.tail()
```

Out[36]:

	tweet_text	brand	sentiment
8381	mention yup but i dont have a third app yet im on android any suggestions sxsw cc mention	android	neutral
8382	wave buzz rt mention we interrupt your regularly scheduled sxsw geek programming with big news link google circles	google	neutral
8383	googles zeiger a physician never reported potential ae yet fda relies on physicians quotwere operating wout dataquot sxsw health2dev	google	neutral
8384	some verizon iphone customers complained their time fell back an hour this weekend of course they were the new yorkers who attended sxsw	iphone	neutral
8385	{ï¡×ïà(ü))ê))î))ò))£))á)ââ)))£))÷)â)ûârt mention google tests ❱ûïcheckin offers❱û at sxsw link	google	neutral

The last row's text is an encoded string, thus we will drop this row

In [37]:

```
# drop the last row
concatenated_df= concatenated_df.drop(concatenated_df.index[-1])
concatenated_df.tail()
```

Out[37]:

	tweet_text	brand	sentiment
8380	google says want to give a lightning talk to a h4ckers audience at sxsw tonight email benmcgraw mention gmailcom for a spot on stage	google	neutral
8381	mention yup but i dont have a third app yet im on android any suggestions sxsw cc mention	android	neutral
8382	wave buzz rt mention we interrupt your regularly scheduled sxsw geek programming with big news link google circles	google	neutral
8383	googles zeiger a physician never reported potential ae yet fda relies on physicians quotwere operating wout dataquot sxsw health2dev	google	neutral
8384	some verizon iphone customers complained their time fell back an hour this weekend of course they were the new yorkers who attended sxsw	iphone	neutral

Step 3 :Tokenizing the data

Now we will tokenize all the cleaned tweets in our dataset. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.

In [38]:

```
def tokenize_text(text):
    return word_tokenize(text)

concatenated_df['tweet_text'] = concatenated_df['tweet_text'].apply(tokenize_text)
concatenated_df.head()
```

Out[38]:

	tweet_text	brand	sentiment
0	[wesley83, i, have, a, 3g, iphone, after, 3, hrs, tweeting, at, riseaustin, it, was, dead, i, need, to, upgrade, plugin, stations, at, sxsw]	iphone	Negative emotion
1	[jessedee, know, about, fludapp, awesome, ipadiphone, app, that, youll, likely, appreciate, for, its, design, also, theyre, giving, free, ts, at, sxsw]	ipad or iphone app	Positive emotion
2	[swonderlin, can, not, wait, for, ipad, 2, also, they, should, sale, them, down, at, sxsw]	ipad	Positive emotion
3	[sxsw, i, hope, this, years, festival, isnt, as, crashy, as, this, years, iphone, app, sxsw]	ipad or iphone app	Negative emotion
4	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	google	Positive emotion

In [39]:

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/pk/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[39]:

```
True
```

Step 4 : Check and Remove Stopwords

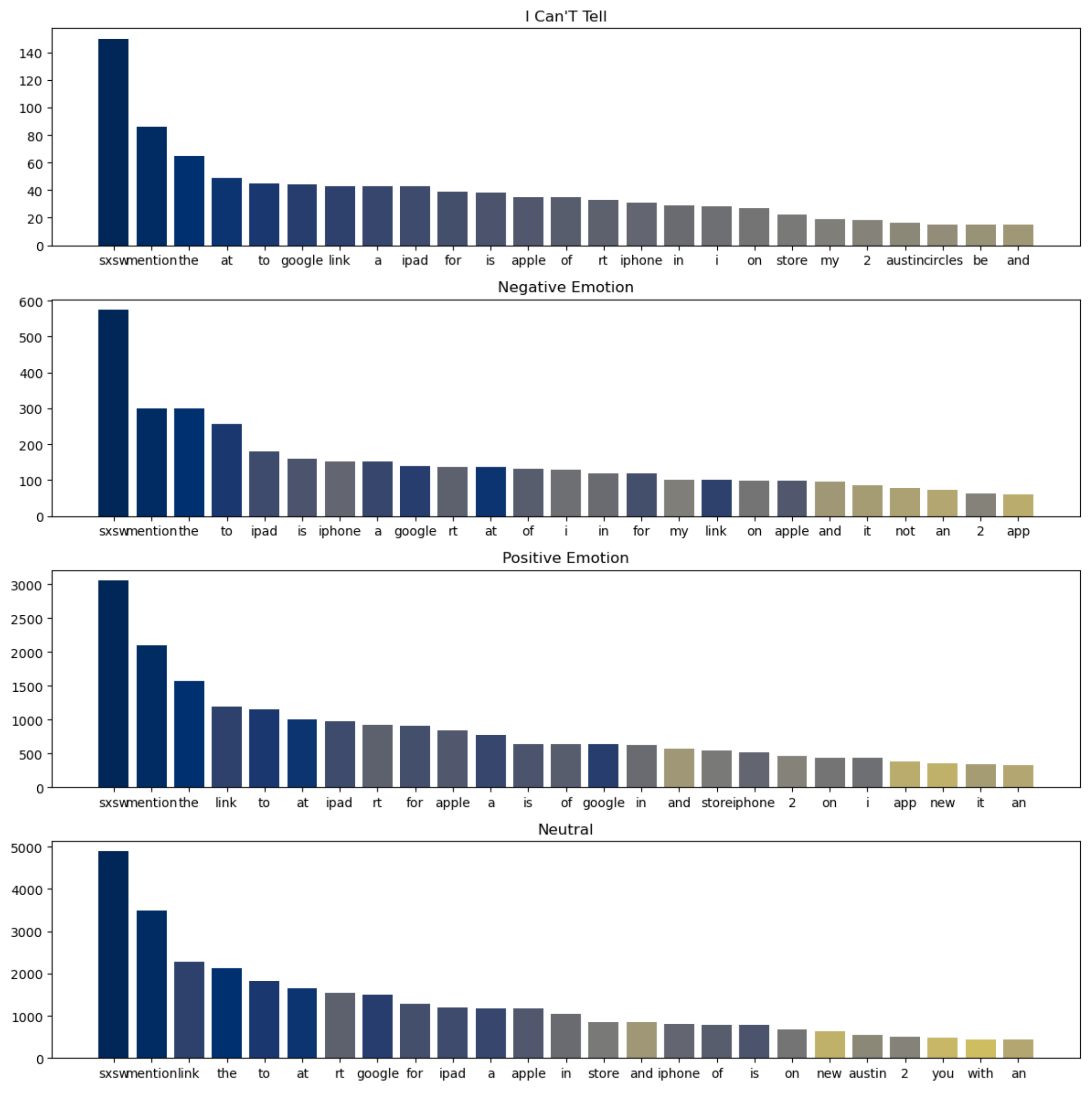
The stop words are used to eliminate unimportant words, allowing applications to focus on the important words instead.They provide no meaningful information, especially if we are building a text classification model. Therefore, we have to remove stopwords from our dataset. As the frequency of stop words are too high, removing them results in much smaller data in terms of size. Reduced size results in faster computations on text data and the text classification model need to deal

Text document is generally made up of characters, words, sentences, paragraphs, etc. If we want to get the intuition of text data then we do basic statistical analysis by exploring word frequency distribution in the data.

Below we plot the most frequent word distribution across different sentiments

In [40]:

```
# Set up figure and axes
fig, axes = plt.subplots(nrows=4, figsize=(12, 12))

# Empty dict to hold words that have already been plotted and their colors
plotted_words_and_colors = {}
# Establish color palette to pull from
# (If you get an error message about popping from an empty list, increase this #)
color_palette = sns.color_palette('cividis', n_colors=38)

# Creating a plot for each unique genre
data_by_genre = [y for _, y in concatenated_df.groupby('sentiment', as_index=False)
for idx, genre_df in enumerate(data_by_genre):
    # Find top 10 words in this genre
    all_words_in_genre = genre_df.tweet_text.explode()
    top_10 = all_words_in_genre.value_counts()[:25]

    # Select appropriate colors, reusing colors if words repeat
    colors = []
    for word in top_10.index:
        if word not in plotted_words_and_colors:
            new_color = color_palette.pop(0)
            plotted_words_and_colors[word] = new_color
        colors.append(plotted_words_and_colors[word])

    # Select axes, plot data, set title
    ax = axes[idx]
    ax.bar(top_10.index, top_10.values, color=colors)
    ax.set_title(genre_df.iloc[0].sentiment.title())

fig.tight_layout()
```



From the plots above we have identified the following stopwords, to be added to the default nltk stopwords list:

- SXSW - this is a tech conference being referred to thus will not give us any meaning
- Mention - this is a twitter functionality for addressing a particular account
- rt - twitter functionality for retweet
- link - place holder for a url
- the, to, at, a, in, of, is - these and others will be removed by the default stopword list

Below we add the custom stop words identified, then remove them from the corpus

In [41]:

```
# initialize stopword list
stopwords_list = stopwords.words('english')

# append custom stop words
stop_add = ['sxsw', 'mention', 'rt', 'link']
[stopwords_list.append(s) for s in stop_add]

# remove stopwords
#no_stop = [word for word in concatenated_df.tweet_text[] if word not in stopwords_

no_stop = []
# Loop through each row in the "tweet_text" column
for index, row in concatenated_df.iterrows():
    tweet = row['tweet_text']

    # Remove stop words from the tweet
    filtered_words = [word for word in tweet if word not in stopwords_list]

    # Append the filtered tweet to the list
    no_stop.append(filtered_words)

# add the cleaned stopwords as column to dataframe
concatenated_df['no_stop'] = no_stop
concatenated_df.head()
```

Out[41]:

	tweet_text	brand	sentiment	no_stop
0	[wesley83, i, have, a, 3g, iphone, after, 3, hrs, tweeting, at, riseaustin, it, was, dead, i, need, to, upgrade, plugin, stations, at, sxsw]	iphone	Negative emotion	[wesley83, 3g, iphone, 3, hrs, tweeting, riseaustin, dead, need, upgrade, plugin, stations]
1	[jessedee, know, about, fludapp, awesome, ipadiphone, app, that, youll, likely, appreciate, for, its, design, also, theyre, giving, free, ts, at, sxsw]	ipad or iphone app	Positive emotion	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, ts]
2	[swonderlin, can, not, wait, for, ipad, 2, also, they, should, sale, them, down, at, sxsw]	ipad	Positive emotion	[swonderlin, wait, ipad, 2, also, sale]
3	[sxsw, i, hope, this, years, festival, isnt, as, crashy, as, this, years, iphone, app, sxsw]	ipad or iphone app	Negative emotion	[hope, years, festival, isnt, crashy, years, iphone, app]
4	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	google	Positive emotion	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]

Step 5 : Lemmatization

Lemmatization is the process of grouping together different inflected forms of the same word.It gets to links similar meaning words as one word. The goal of lemmatization is to reduce a word to its root form, also called a lemma. For example, the verb "running" would be identified as "run."

We are using Lemmatization over stemming - Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling. Lemmatization considers the context and converts the word to its meaningful base form, which is called Lemma

In [42]:

```
# initialize the lemmatizer
lem = WordNetLemmatizer()

lema = []
# Loop through each row in the "no_stop" column
for index, row in concatenated_df.iterrows():
    tweet = row['no_stop']

    # Remove stop words from the tweet
    lematized_words = [lem.lemmatize(word) for word in tweet]

    # Append the filtered tweet to the list
    lema.append(lematized_words)

# add the cleaned stopwords as column to dataframe
concatenated_df['lemmatized'] = lema
concatenated_df.head()
```

Out[42]:

	tweet_text	brand	sentiment	no_stop	lemmatized
0	[wesley83, i, have, a, 3g, iphone, after, 3, hrs, tweeting, at, riseaustin, it, was, dead, i, need, to, upgrade, plugin, stations, at, sxsw]	iphone	Negative emotion	[wesley83, 3g, iphone, 3, hrs, tweeting, riseaustin, dead, need, upgrade, plugin, stations]	[wesley83, 3g, iphone, 3, hr, tweeting, riseaustin, dead, need, upgrade, plugin, station]
1	[jessedee, know, about, fludapp, awesome, ipadiphone, app, that, youll, likely, appreciate, for, its, design, also, theyre, giving, free, ts, at, sxsw]	ipad or iphone app	Positive emotion	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, ts]	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, t]
2	[swonderlin, can, not, wait, for, ipad, 2, also, they, should, sale, them, down, at, sxsw]	ipad	Positive emotion	[swonderlin, wait, ipad, 2, also, sale]	[swonderlin, wait, ipad, 2, also, sale]
3	[sxsw, i, hope, this, years, festival, isnt, as, crashy, as, this, years, iphone, app, sxsw]	ipad or iphone app	Negative emotion	[hope, years, festival, isnt, crashy, years, iphone, app]	[hope, year, festival, isnt, crashy, year, iphone, app]
4	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	google	Positive emotion	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]

VISUALIZATIONS

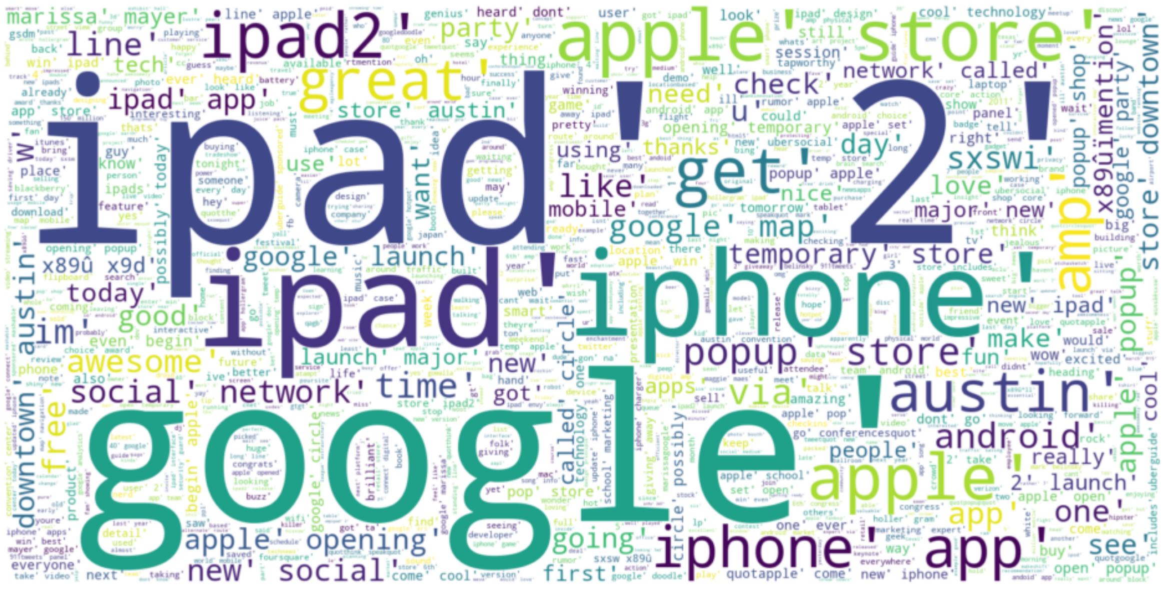
Word cloud visualization

A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes. By highlighting the most frequent terms in a corpus, word clouds can help identify commonly used or repeated words. Conversely, rare or unique terms may also stand out in the visualization, providing insights into less common aspects or specific attributes.

Plot a word Cloud for the positive tweets

In [43]:

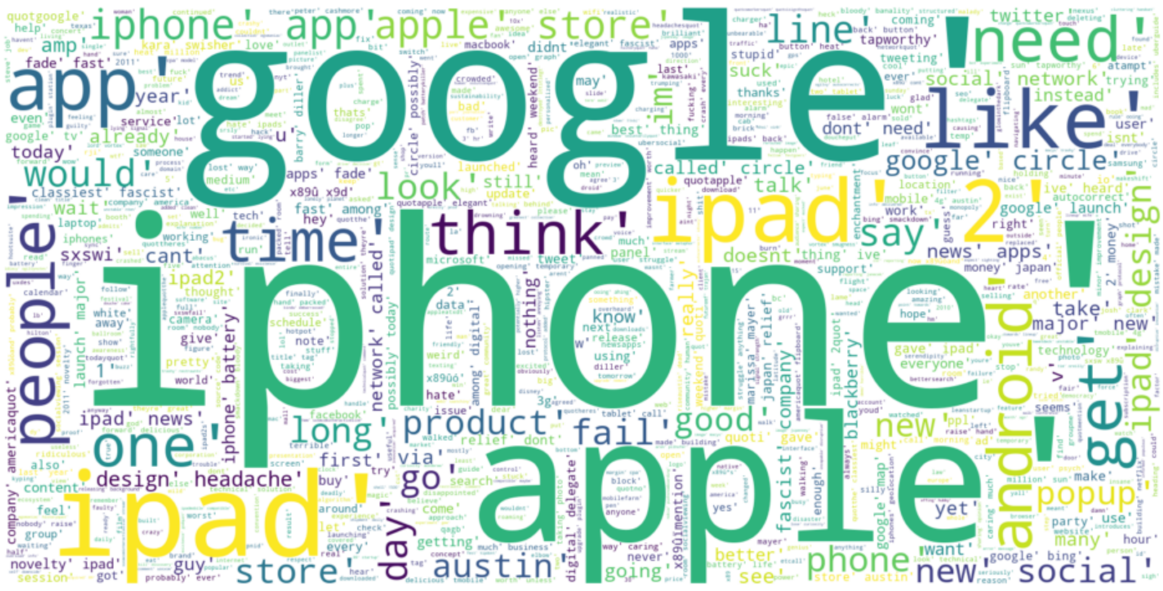
```
def generate_word_cloud(data, label):
    tweet = " ".join(str(tweet) for tweet in data.loc[data.sentiment == label].lemm
    wordcloud = WordCloud(background_color='white', width=1600, height=800, max_wor
    plt.figure(figsize=(12,8))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
#Wordcloud for the Positive emotion
generate_word_cloud(concatenated_df,'Positive emotion')
```



Plot a word Cloud for the Negative tweets

In [44]:

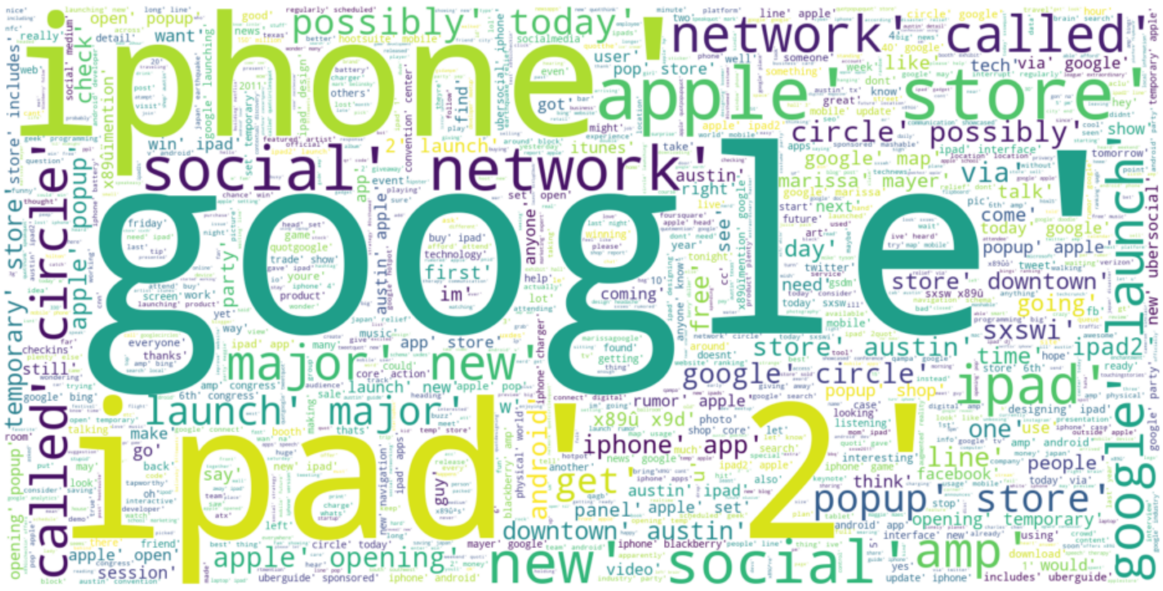
```
# Word cloud for the negative emotion
generate_word_cloud(concatenated_df, 'Negative emotion')
```



Plot a word Cloud for the neutral tweets

In [45]:

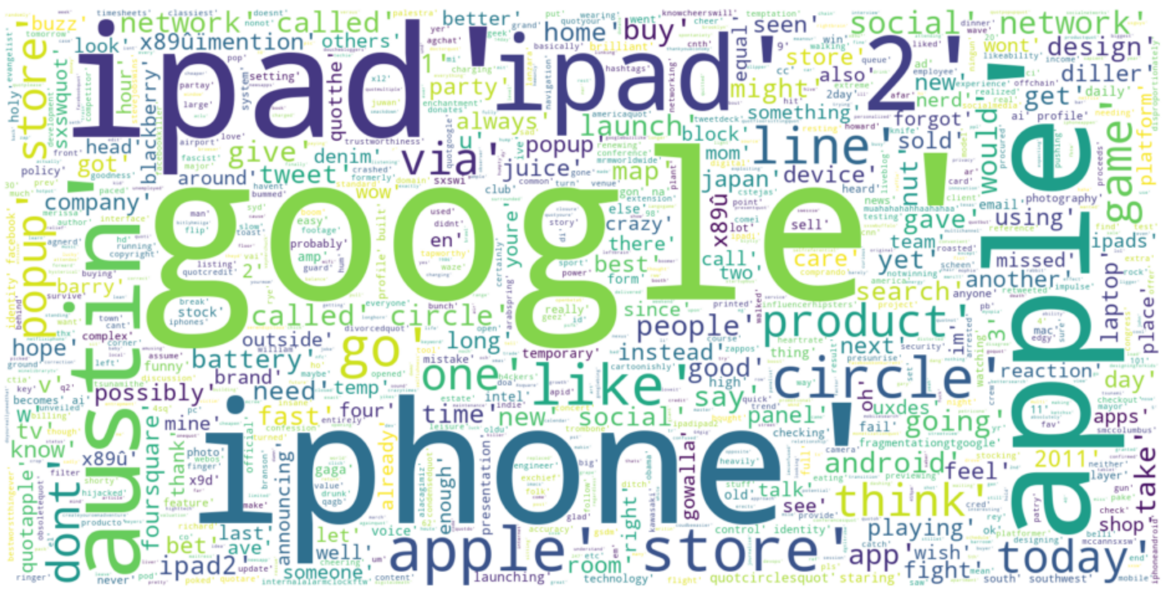
```
# Word cloud for the Neutral emotion
generate_word_cloud(concatenated_df, 'neutral')
```



Plot a word Cloud for the tweet which can`t tell the emotion

In [46]:

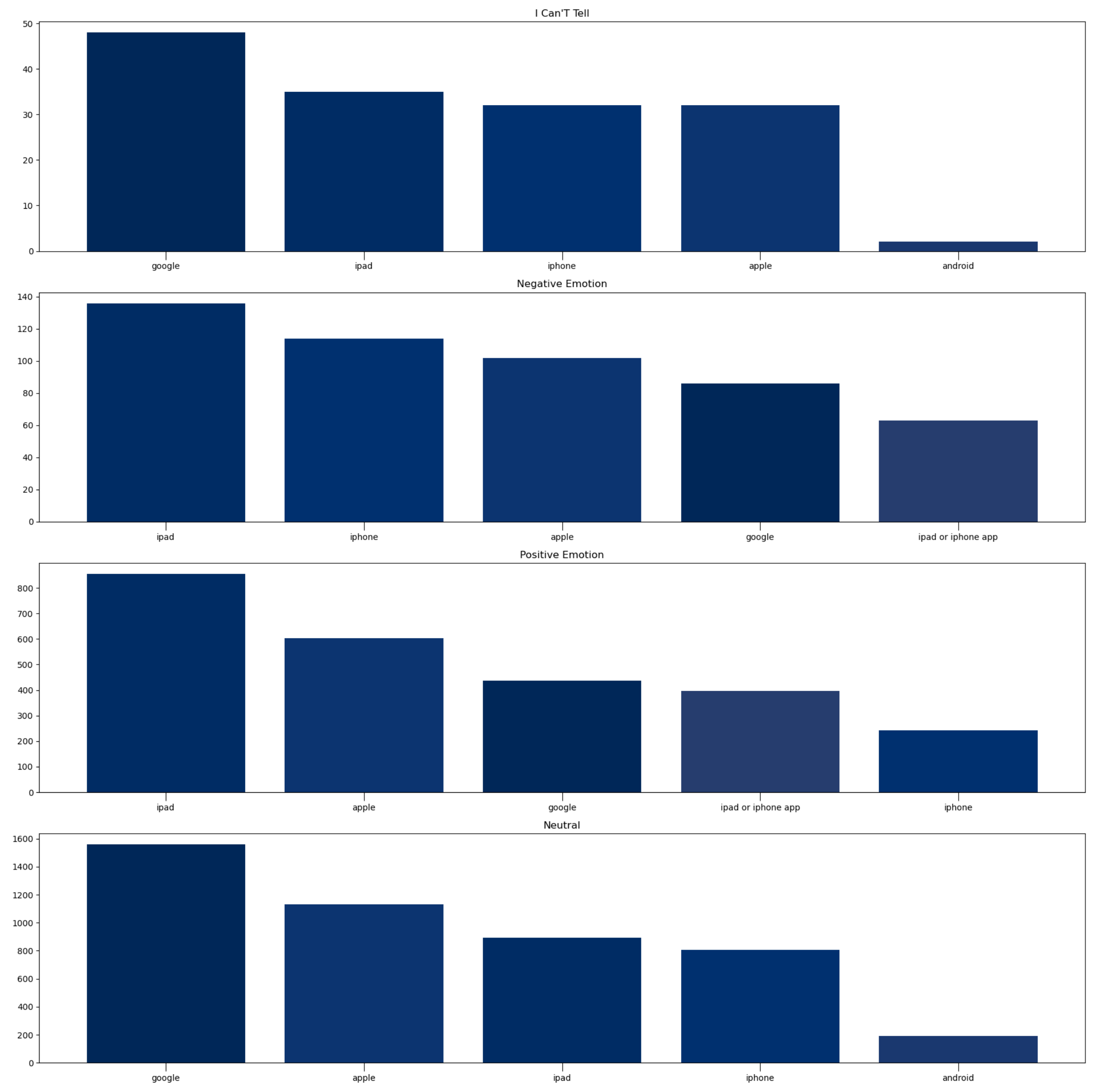
```
# Word cloud for 'I can't tell'
generate_word_cloud(concatenated_df, "I can't tell")
```



Brand Analysis Based on the emotions

In [47]:

```
# Brand analysis
# Set up figure and axes
fig, axes = plt.subplots(nrows=4, figsize=(18, 18))
# Empty dict to hold words that have already been plotted and their colors
plotted_words_and_colors = {}
# Establish color palette to pull from
# (If you get an error message about popping from an empty list, increase this #)
color_palette = sns.color_palette('cividis', n_colors=38)
# Creating a plot for each unique genre
data_by_genre = [y for _, y in concatenated_df.groupby('sentiment', as_index=False)
for idx, genre_df in enumerate(data_by_genre):
    # Find top 10 words in this genre
    all_words_in_genre = genre_df.brand.explode()
    top_10 = all_words_in_genre.value_counts()[:5]
    # Select appropriate colors, reusing colors if words repeat
    colors = []
    for word in top_10.index:
        if word not in plotted_words_and_colors:
            new_color = color_palette.pop(0)
            plotted_words_and_colors[word] = new_color
        colors.append(plotted_words_and_colors[word])
    # Select axes, plot data, set title
    ax = axes[idx]
    ax.bar(top_10.index, top_10.values, color=colors)
    ax.set_title(genre_df.iloc[0].sentiment.title())
    ax.tick_params(axis='x', size=10)
fig.tight_layout()
```



```
The google and ipad are the two brands seen to have most tweets and reviews.
The ipad customers/users are seen to have top positive and negative emotions
whereas
google customers have either neutral or can not say emotions towards the products
at the highest count
```

Label Encoding

```
We label encode the sentiments in order to have 1 represent negative emotion and
2 for the positive emotion, 3 for neutral emotion and 0 for can`t tell the emotion
thus ease in use
```

In [48]:

```
le = LabelEncoder()
concatenated_df['sentiment_target'] = le.fit_transform(concatenated_df.sentiment)
concatenated_df.head()
```

Out[48]:

	tweet_text	brand	sentiment	no_stop	lemmatized	sentiment_target
0	[wesley83, i, have, a, 3g, iphone, after, 3, hrs, tweeting, at, riseaustin, it, was, dead, i, need, to, upgrade, plugin, stations, at, sxsw]	iphone	Negative emotion	[wesley83, 3g, iphone, 3, hrs, tweeting, riseaustin, dead, need, upgrade, plugin, stations]	[wesley83, 3g, iphone, 3, hr, tweeting, riseaustin, dead, need, upgrade, plugin, station]	1
1	[jessedee, know, about, fludapp, awesome, ipadiphone, app, that, youll, likely, appreciate, for, its, design, also, theyre, giving, free, ts, at, sxsw]	ipad or iphone app	Positive emotion	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, ts]	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, t]	2
2	[swonderlin, can, not, wait, for, ipad, 2, also, they, should, sale, them, down, at, sxsw]	ipad	Positive emotion	[swonderlin, wait, ipad, 2, also, sale]	[swonderlin, wait, ipad, 2, also, sale]	2
3	[sxsw, i, hope, this, years, festival, isnt, as, crashy, as, this, years, iphone, app, sxsw]	ipad or iphone app	Negative emotion	[hope, years, festival, isnt, crashy, years, iphone, app]	[hope, year, festival, isnt, crashy, year, iphone, app]	1
4	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	google	Positive emotion	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	2

Now we have done our text preprocessing part and we will move onto the Vectorization and Model Selection Vectorization and Model Selection.

In [49]:

```
concatenated_df['products'] = concatenated_df['brand'].map({'other apple product or
                                                  'android': 'google products',
                                                  'android app': 'google products',
                                                  'other google product or service ': 'goo
                                                  'iphone': 'apple products',
                                                  'google':'google products',
                                                  'ipad or iphone app': 'apple products',
                                                  'apple':'apple products',
                                                  'ipad':'apple products',
                                                  'itunes': 'apple products'})
```

In [50]:

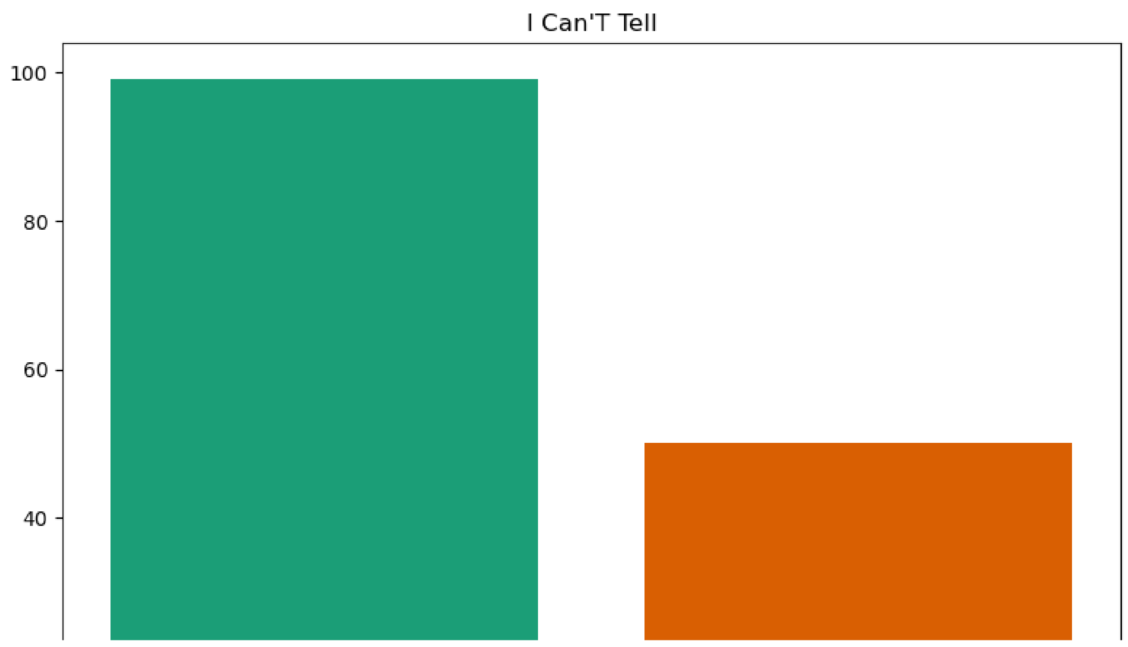
```
concatenated_df.head()
```

Out[50]:

	tweet_text	brand	sentiment	no_stop	lemmatized	sentiment_target
0	[wesley83, i, have, a, 3g, iphone, after, 3, hrs, tweeting, at, riseaustin, it, was, dead, i, need, to, upgrade, plugin, stations, at, sxsw]	iphone	Negative emotion	[wesley83, 3g, iphone, 3, hrs, tweeting, riseaustin, dead, need, upgrade, plugin, stations]	[wesley83, 3g, iphone, 3, hr, tweeting, riseaustin, dead, need, upgrade, plugin, station]	1
1	[jessedee, know, about, fludapp, awesome, ipadiphone, app, that, youll, likely, appreciate, for, its, design, also, theyre, giving, free, ts, at, sxsw]	ipad or iphone app	Positive emotion	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, ts]	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, t]	2
2	[swonderlin, can, not, wait, for, ipad, 2, also, they, should, sale, them, down, at, sxsw]	ipad	Positive emotion	[swonderlin, wait, ipad, 2, also, sale]	[swonderlin, wait, ipad, 2, also, sale]	2
3	[sxsw, i, hope, this, years, festival, isnt, as, crashy, as, this, years, iphone, app, sxsw]	ipad or iphone app	Negative emotion	[hope, years, festival, isnt, crashy, years, iphone, app]	[hope, year, festival, isnt, crashy, year, iphone, app]	1
4	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	google	Positive emotion	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	2

In [51]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set up figure and axes
fig, axes = plt.subplots(ncols=1, nrows=4, figsize=(8, 24))

# Empty dict to hold words that have already been plotted and their colors
plotted_words_and_colors = {}

# Establish color palette to pull from
# (If you get an error message about popping from an empty list, increase this #)
color_palette = sns.color_palette('Dark2', n_colors=38)

# Creating a plot for each unique genre
data_by_genre = [y for _, y in concatenated_df.groupby('sentiment', as_index=False)

for idx, genre_df in enumerate(data_by_genre):
    # Find top 10 words in this genre
    all_words_in_genre = genre_df.products.explode()
    top_10 = all_words_in_genre.value_counts()[:25]

    # Select appropriate colors, reusing colors if words repeat
    colors = []
    for word in top_10.index:
        if word not in plotted_words_and_colors:
            new_color = color_palette.pop(0)
            plotted_words_and_colors[word] = new_color
        colors.append(plotted_words_and_colors[word])

    # Select axes, plot data, set title
    ax = axes[idx]
    ax.bar(top_10.index, top_10.values, color=colors)
    ax.set_title(genre_df.iloc[0].sentiment.title())
    ax.tick_params(axis='x', size=10)

plt.tight_layout()
plt.show()
```



```
From the plot above we can derive that the apple products have more customers and
thus reviews unlike the google products
```

One Hot Encoding

In [52]:

```
final_df= pd.get_dummies(concatenated_df, columns=['products'])
```

In [53]:

```
# Convert token lists to strings
final_df["lemmatized"] = final_df["lemmatized"].str.join(" ")
```

In [54]:

```
final_df.head()
```

Out[54]:

	tweet_text	brand	sentiment	no_stop	lemmatized	sentiment_target	p
0	[wesley83, i, have, a, 3g, iphone, after, 3, hrs, tweeting, at, riseaustin, it, was, dead, i, need, to, upgrade, plugin, stations, at, sxsw]	iphone	Negative emotion	[wesley83, 3g, iphone, 3, hrs, tweeting, riseaustin, dead, need, upgrade, plugin, stations]	wesley83 3g iphone 3 hr tweeting riseaustin dead need upgrade plugin station	1	
1	[jessedee, know, about, fludapp, awesome, ipadiphone, app, that, youll, likely, appreciate, for, its, design, also, theyre, giving, free, ts, at, sxsw]	ipad or iphone app	Positive emotion	[jessedee, know, fludapp, awesome, ipadiphone, app, youll, likely, appreciate, design, also, theyre, giving, free, ts]	jessedee know fludapp awesome ipadiphone app youll likely appreciate design also theyre giving free t	2	
2	[swonderlin, can, not, wait, for, ipad, 2, also, they, should, sale, them, down, at, sxsw]	ipad	Positive emotion	[swonderlin, wait, ipad, 2, also, sale]	swonderlin wait ipad 2 also sale	2	
3	[sxsw, i, hope, this, years, festival, isnt, as, crashy, as, this, years, iphone, app, sxsw]	ipad or iphone app	Negative emotion	[hope, years, festival, isnt, crashy, years, iphone, app]	hope year festival isnt crashy year iphone app	1	
4	[sxtxstate, great, stuff, on, fri, sxsw, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	google	Positive emotion	[sxtxstate, great, stuff, fri, marissa, mayer, google, tim, oreilly, tech, booksconferences, amp, matt, mullenweg, wordpress]	sxtxstate great stuff fri marissa mayer google tim oreilly tech booksconferences amp matt mullenweg wordpress	2	

Train Test Split

In [55]:

```
X= final_df['lemmatized']
y= final_df['sentiment_target']
```

In [56]:

```
X_train, X_test, y_train, y_test= train_test_split(X, y, test_size=0.3, random_stat
```

Modelling

We have used several different models which are:

- Logistic Regression- Baseline model
- Multinomial Naive Bayes Classifier
- Gradient boosting classifier
- XGBoost
- Decision Trees
- Random Forest The idea behind choosing these models is that we want to try all the classifiers on the dataset ranging from simple ones to complex models, and then try to find out the one which gives the best performance among them.

Before we let our data to train we have to numerically represent the preprocessed data. The well-known techniques for vectorization of words in Natural Language Processing are: CountVectorization and Tf-IDF transformation. Let's dive a bit into the theoretical background of those vectorization techniques. CountVectorization generates a sparse matrix representing all the words in the document.

In [57]:

```
# Instantiate CountVectorizer class and fit transform on X_train
vectorizer = CountVectorizer()
train_vec= vectorizer.fit_transform(X_train)
```

In [58]:

```
# Transform the test set
test_vec= vectorizer.transform(X_test)
```

In [59]:

```
# Instantiate Tf-IDF class and fit transform on X_train
tfidf= TfidfVectorizer()
train_tfidf= tfidf.fit_transform(X_train)
```

In [60]:

```
# Transform the test set
test_tfidf= tfidf.transform(X_test)
```

Logistic Regression

In [61]:

```
# Instantiate Logistic Regression
lr= LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=150, ran
# Fit on training data
lr.fit(train_vec, y_train)
```

Out[61]:

```
LogisticRegression(max_iter=150, multi_class='multinomial', random_sta
te=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [62]:

```
y_train_pred= lr.predict(train_vec)
```

In [63]:

```
y_test_pred= lr.predict(test_vec)
```

In [64]:

```
print(accuracy_score(y_train,y_train_pred))
print(accuracy_score(y_test,y_test_pred))
```

```
0.9089352961314618
0.6333865814696485
```

In [65]:

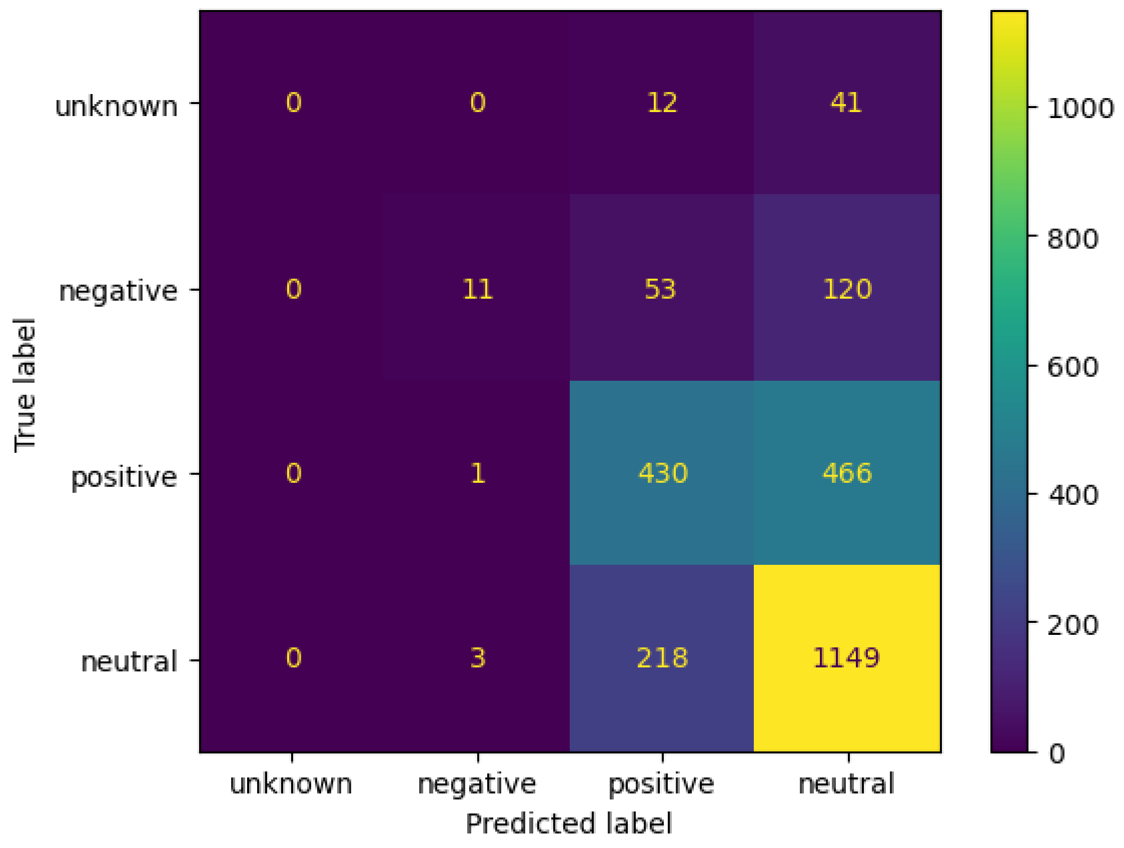
```
print(f1_score(y_train,y_train_pred, average='weighted'))
print(f1_score(y_test,y_test_pred, average='weighted'))
```

```
0.9075866446084968
0.6134361490809348
```

In [66]:

```
# Function for the logistic regression model
def logistic_reg(X_train, X_test, y_train, y_test):
    # Instantiate the model with max_iter=150
    lr = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=150
    # Fit the model to the training data
    lr.fit(X_train, y_train)
    # Training set accuracy
    print("Training Model Accuracy:")
    print(accuracy_score(y_train, lr.predict(X_train)))
    # Check the accuracy of the model
    test_preds = lr.predict(X_test)
    print("Accuracy Score:", accuracy_score(y_test, test_preds))
    # Access the parameters used in the model
    print("Model Parameters:")
    print(lr.get_params)
    # Plot a confusion matrix to visualize the actual and predicted values
    cm = confusion_matrix(y_test, test_preds)
    display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['unknown'
    display.plot()

# Initialize and evaluate the model
logistic_reg(train_tfidf, test_tfidf, y_train, y_test)
```

```
Training Model Accuracy:
0.8036631290653886
Accuracy Score: 0.6349840255591054
Model Parameters:
<bound method BaseEstimator.get_params of LogisticRegression(max_iter=
150, multi_class='multinomial', random_state=42)>
```



The results from the logistic regression model show that:

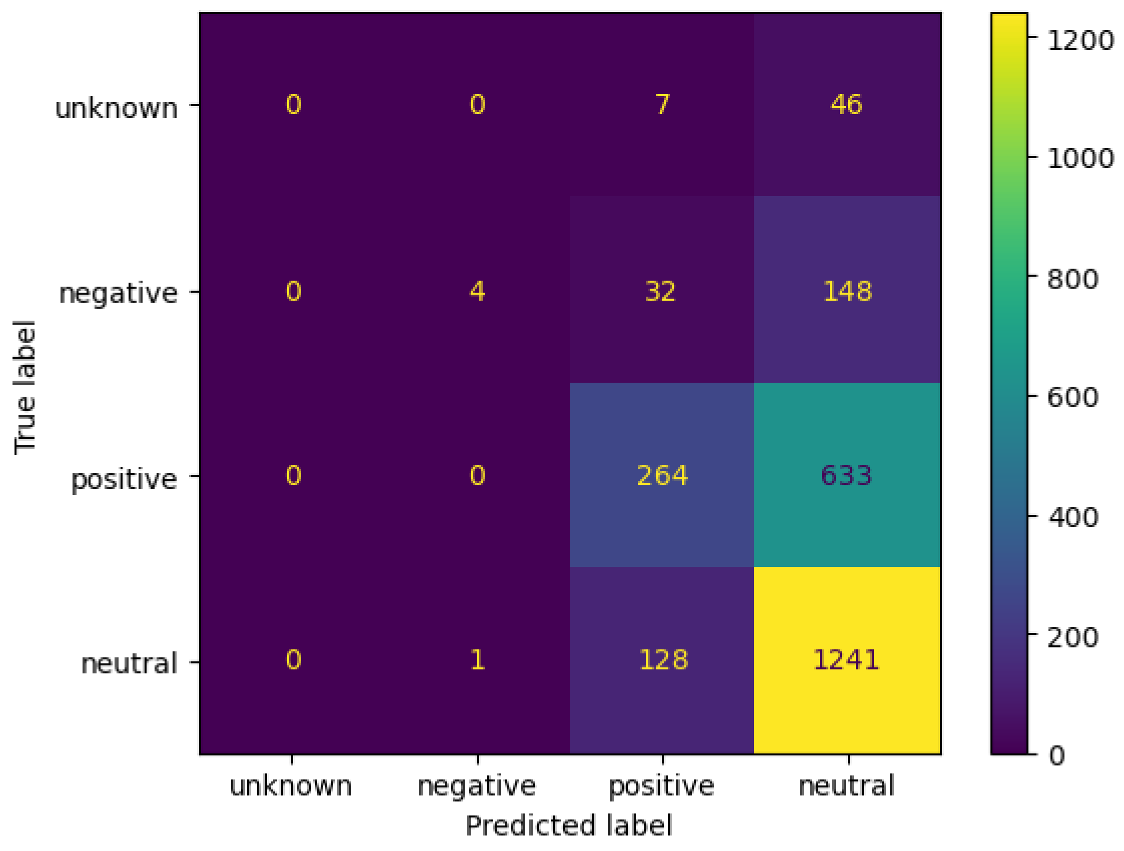
- The model is overfitting the train data, since the training accuracy is way higher compared to the test accuracy.
- We will try another model and check on its accuracy aiming at improving the results.

MultinomialNB

In [67]:

```
# Function for the MultinomialNB model
def multinomialnb_model(X_train, X_test, y_train, y_test):
    # Instantiate the model
    mnbmodel = MultinomialNB()
    # Fit the model to the training data
    mnbmodel.fit(X_train, y_train)
    # Training set accuracy
    print("Training Model Accuracy:")
    print(accuracy_score(y_train, mnbmodel.predict(X_train)))
    # Check the accuracy of the model
    test_preds = mnbmodel.predict(X_test)
    print("Accuracy Score:", accuracy_score(y_test, test_preds))
    # Access the parameters used in the model
    print("Model Parameters:")
    print(mnbmodel.get_params)
    # Plot a confusion matrix to visualize the actual and predicted values
    cm = confusion_matrix(y_test, test_preds)
    display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['unknown'
    display.plot()

# Evaluate the model
multinomialnb_model(train_tfidf, test_tfidf, y_train, y_test)
```

```
Training Model Accuracy:
0.7564190345771996
Accuracy Score: 0.6026357827476039
Model Parameters:
<bound method BaseEstimator.get_params of MultinomialNB()>
```



The model results from the MultinomialNB model shows that:

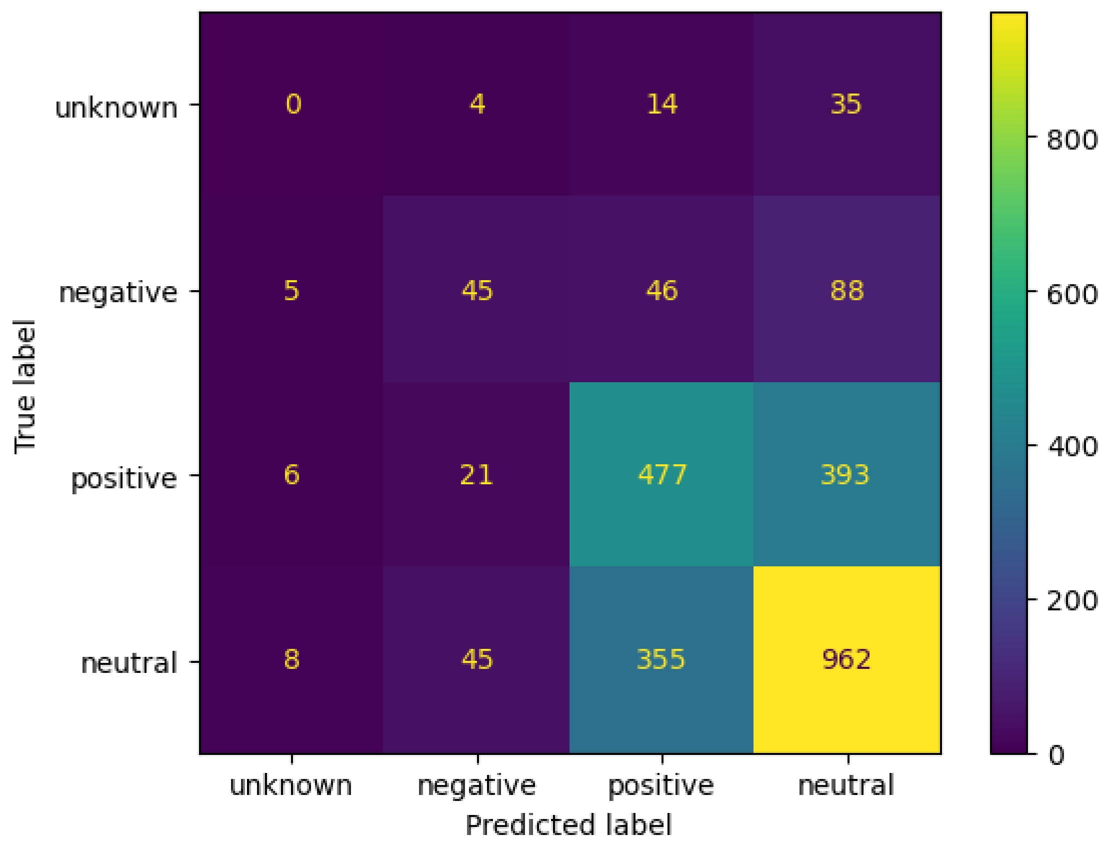
- Our model is overfitting, however, less slightly than the logistic regression model.
- Try another model-Decision Trees- and evaluate its performance.

Decision Trees

In [68]:

```
# Function for the Decision trees model
def tree_model(X_train, X_test, y_train, y_test):
    # Instantiate the model
    tree = DecisionTreeClassifier(random_state=42)

    # Fit the model to the training data
    tree.fit(train_tfidf, y_train)
    # Training set accuracy
    print("Training Model Accuracy:")
    print(accuracy_score(y_train, tree.predict(X_train)))
    # Check the accuracy of the model
    test_preds = tree.predict(X_test)
    print("Accuracy Score:", accuracy_score(y_test, test_preds))
    # Access the parameters used in the model
    print("Model Parameters:")
    print(tree.get_params)
    # Plot a confusion matrix to visualize the actual and predicted values
    cm = confusion_matrix(y_test, test_preds)
    display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['unknown'
    display.plot()

# Evaluate the model
tree_model(train_tfidf, test_tfidf, y_train, y_test)
```

```
Training Model Accuracy:
0.9589181787059227
Accuracy Score: 0.5926517571884984
Model Parameters:
<bound method BaseEstimator.get_params of DecisionTreeClassifier(rando
m_state=42)>
```



The Decision Tree model has given the following results:

- The model is highly overfitting, way worse than the base model-logistic regression
- The accuracy level on the unseen data is very poor, which proves the model is not the best solution.

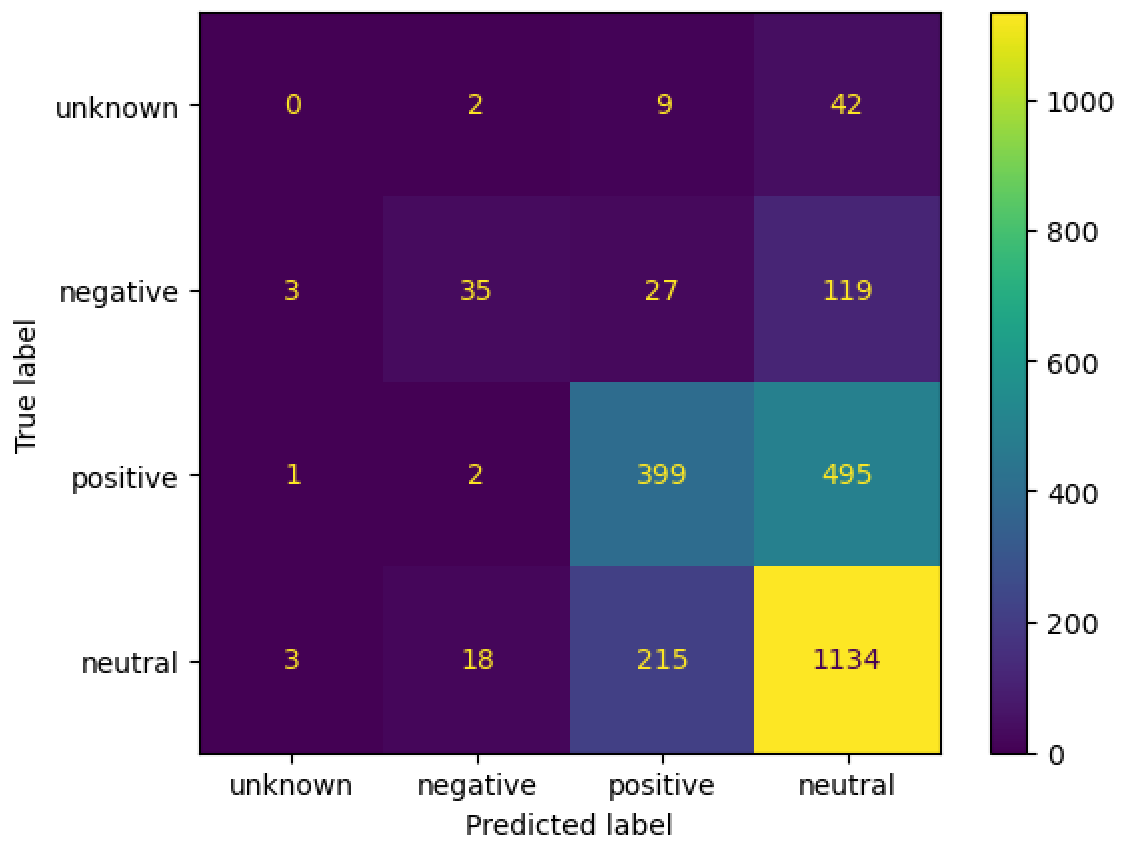
We will try another model, and evaluate its performance: Random Forest

Random Forest

In [69]:

```
# Function for the Random Forest model
def forest_model(X_train, X_test, y_train, y_test):
    # Instantiate the model
    forest = RandomForestClassifier(random_state=42)

    # Fit the model to the training data
    forest.fit(train_tfidf, y_train)
    # Training set accuracy
    print("Training Model Accuracy:")
    print(accuracy_score(y_train, forest.predict(X_train)))
    # Check the accuracy of the model
    test_preds = forest.predict(X_test)
    print("Accuracy Score:", accuracy_score(y_test, test_preds))
    # Access the parameters used in the model
    print("Model Parameters:")
    print(forest.get_params)
    # Plot a confusion matrix to visualize the actual and predicted values
    cm = confusion_matrix(y_test, test_preds)
    display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['unknown'
    display.plot()

# Evaluate the model
forest_model(train_tfidf, test_tfidf, y_train, y_test)
```

```
Training Model Accuracy:
0.9587470044505306
Accuracy Score: 0.6261980830670927
Model Parameters:
<bound method BaseEstimator.get_params of RandomForestClassifier(rando
m_state=42)>
```



This model is much better compared to the Decision Tree model, in that it is slightly less overfitting the test data. However, the margin between the train accuracy and the test accuracy is very large, therefore we'll not use this model as our final model.

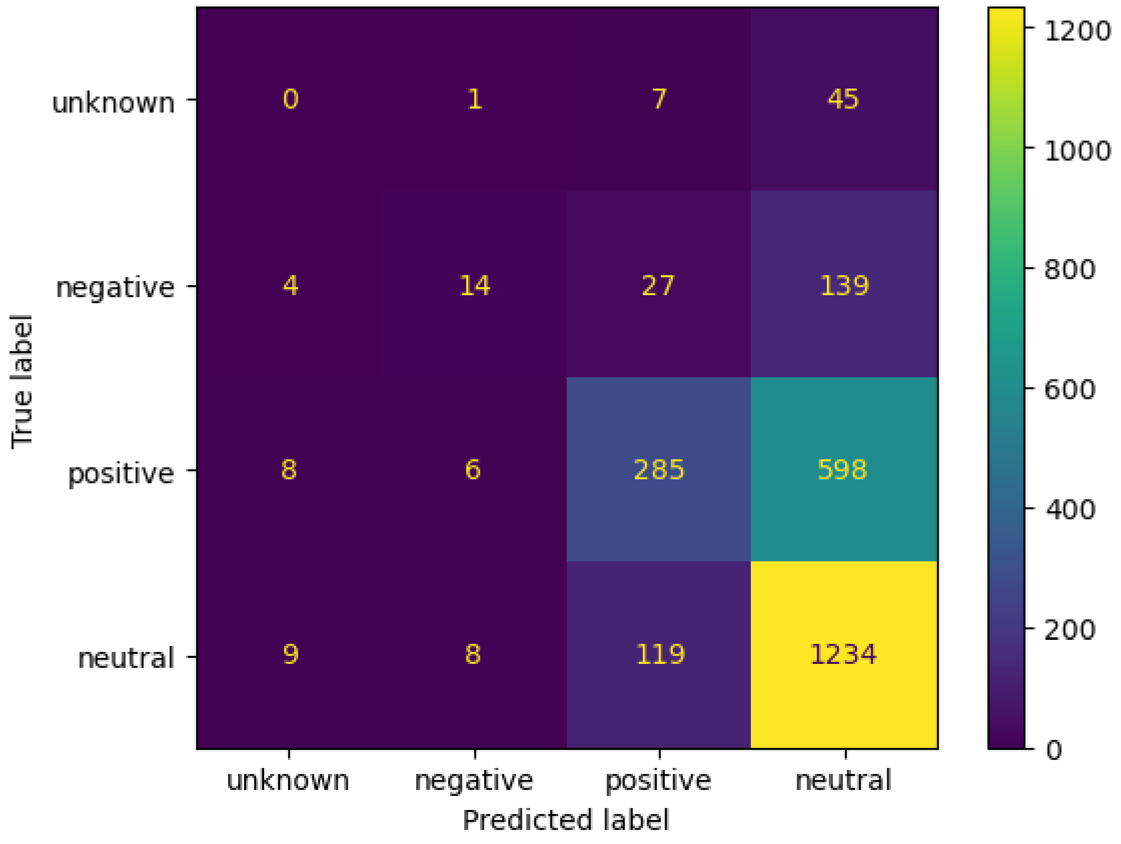
- We will try another model- Gradient Boosted model

Gradient Boost

In [70]:

```python
# Function for the Gradient Boosted model
def gb_model(X_train, X_test, y_train, y_test):
    # Instantiate the model
    gradient_bst = GradientBoostingClassifier(random_state=42)

    # Fit the model to the training data
    gradient_bst.fit(train_tfidf, y_train)
    # Training set accuracy
    print("Training Model Accuracy:")
    print(accuracy_score(y_train, gradient_bst.predict(X_train)))
    # Check the accuracy of the model
    test_preds = gradient_bst.predict(X_test)
    print("Accuracy Score:", accuracy_score(y_test, test_preds))
    # Access the parameters used in the model
    print("Model Parameters:")
    print(gradient_bst.get_params)
    # Plot a confusion matrix to visualize the actual and predicted values
    cm = confusion_matrix(y_test, test_preds)
    display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['unknown'
    display.plot()

# Evaluate the model
gb_model(train_tfidf, test_tfidf, y_train, y_test)
```

```
Training Model Accuracy:
0.7254364943512496
Accuracy Score: 0.612220447284345
Model Parameters:
<bound method BaseEstimator.get_params of GradientBoostingClassifier(r
andom_state=42)>
```



The performance of this model is fairly good, since overfitting of the test data is minimal

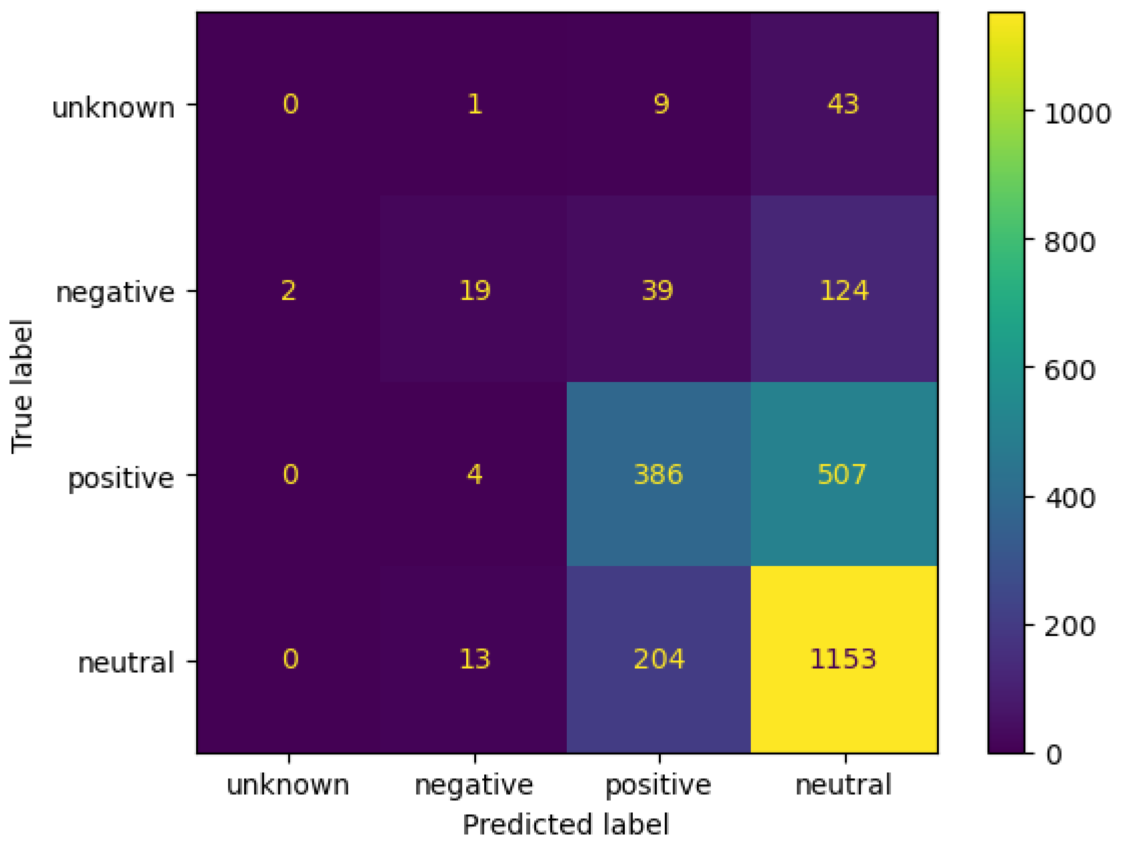
- This is generally a good model, and with proper tuning will be used as a final model

XGBoost

In [71]:

```
# Function for the XGBoost model
def xgboost_model(X_train, X_test, y_train, y_test):
    # Instantiate the model
    xgboost = XGBClassifier(random_state=42)
    # Fit the model to the training data
    xgboost.fit(train_tfidf, y_train)
    # Training set accuracy
    print("Training Model Accuracy:")
    print(accuracy_score(y_train, xgboost.predict(X_train)))
    # Check the accuracy of the model
    test_preds = xgboost.predict(X_test)
    print("Accuracy Score:", accuracy_score(y_test, test_preds))
    # Access the parameters used in the model
    print("Model Parameters:")
    print(xgboost.get_params)
    # Plot a confusion matrix to visualize the actual and predicted values
    cm = confusion_matrix(y_test, test_preds)
    display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['unknown'
    display.plot()

# Evaluate the model
xgboost_model(train_tfidf, test_tfidf, y_train, y_test)
```

```
Training Model Accuracy:
0.8242040397124273
Accuracy Score: 0.6222044728434505
Model Parameters:
<bound method XGBModel.get_params of XGBClassifier(base_score=None, bo
oster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_type
s=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_ty
pe=None,
              interaction_constraints=None, learning_rate=None, max_bi
n=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints
=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              objective='multi:softprob', predictor=None, ...)>
```



The performance of this model is badly off compared to Gradient Boosted model.

- Therefore, we'll tune the parameters of the Gradient Boosted model, to optimize its performance.

Grid Search

In [72]:

```
# create a grid of parameters to test
param_grid = [{
        'criterion': ['friedman_mse'],
     'n_estimators': [1000],
     'learning_rate': [0.1],
     'n_estimators': [500,1000],
     'max_depth': [50,100],
     'min_samples_split': [8],
     'min_samples_leaf': [2]
}]
# create grid with estimators
gridsearch_gb = GridSearchCV(estimator=gradient_bst,
                            param_grid=param_grid,
                            scoring="accuracy",
                            cv=5,
                            n_jobs=1)
# fit the grid with the data
gridsearch_gb.fit(train_tfidf, y_train)
# Get the best parameters
print("Best Parameters:", gridsearch_gb.best_params_)
# Make predictions on the test data using the trained pipeline
y_pred_gridsearchgb = gridsearch_gb.predict(test_tfidf)
# Generate and print the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred_gridsearchgb))
```

Final model

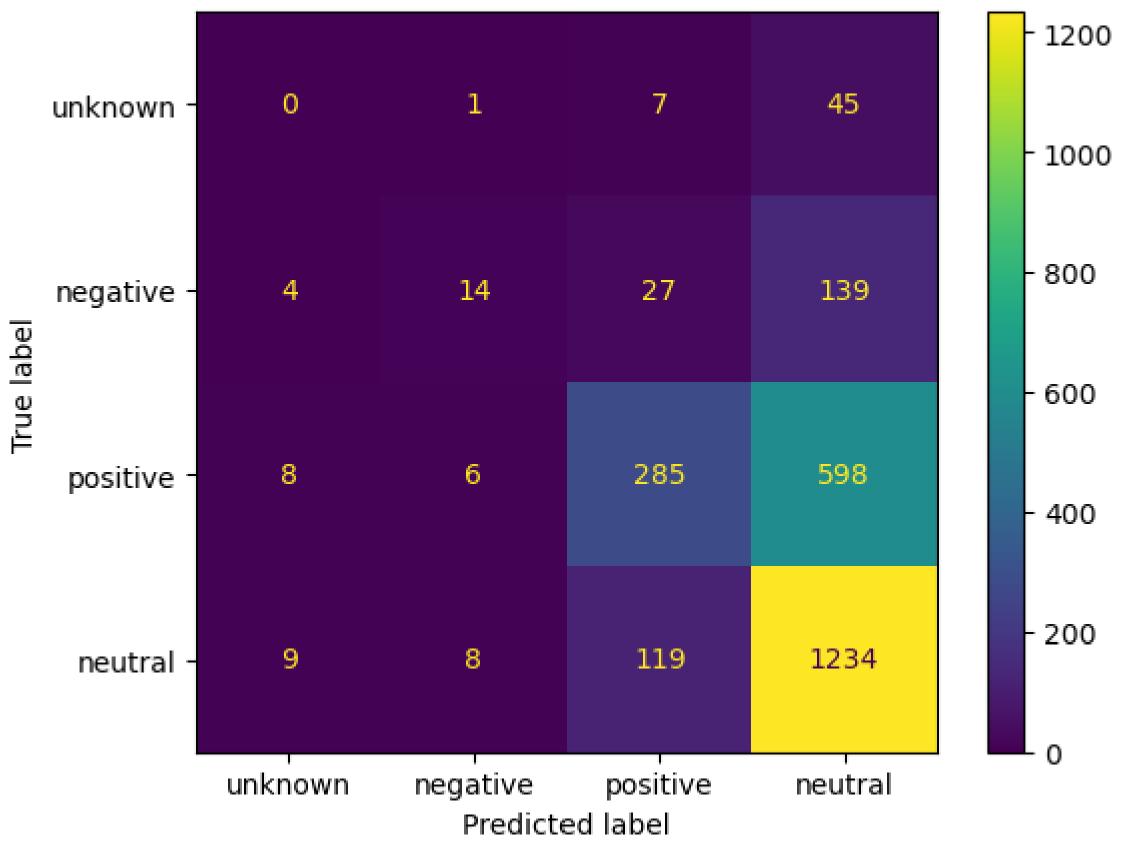
This model is using the best parameters obtained from the above GridSearchCV, where we were determining the best explained parameters for the `Gradient Boosted Model`. These parameters are:

- 'criterion': 'friedman_mse'
- 'learning_rate': 0.1
- 'max_depth': 200
- 'min_samples_leaf': 2
- 'min_samples_split': 8
- 'n_estimators': 1000

In [73]:

```
# Function for the final model
def final_model(X_train, X_test, y_train, y_test):
    # Instantiate the model using the obtained best parameters
    gradient_bst = GradientBoostingClassifier(criterion='friedman_mse',
                                              n_estimators=1000,
                                              learning_rate=0.1,
                                              max_depth=200,
                                              min_samples_split=8,
                                              min_samples_leaf=2,
                                              random_state=42
                                              )
    # Fit the model to the training data
    gradient_bst.fit(train_tfidf, y_train)
    # Training set accuracy
    print("Training Model Accuracy:")
    print(accuracy_score(y_train, gradient_bst.predict(X_train)))
    # Check the accuracy of the model
    test_preds = gradient_bst.predict(X_test)
    print("Test Accuracy Score:")
    print(accuracy_score(y_test, test_preds))
    # Access the parameters used in the model
    print("Model Parameters:")
    print(gradient_bst.get_params)
    # Plot a confusion matrix to visualize the actual and predicted values
    cm = confusion_matrix(y_test, test_preds)
    display = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['unknown'
    display.plot()
# Evaluate the model
gb_model(train_tfidf, test_tfidf, y_train, y_test)
```

```
Training Model Accuracy:
0.7254364943512496
Accuracy Score: 0.612220447284345
Model Parameters:
<bound method BaseEstimator.get_params of GradientBoostingClassifier(r
andom_state=42)>
```



Conclusions

The main objective of this project was to create a model that can classify twitter product reviews/ tweets into the appropriate sentiments-positive, negative, neutral. Through this project, we have met this objective, and also obtained the following results:

- Most of the customers have neutral emotion towards the products, followed by those with positive emotion. There are customers with negative emotions as well.
- The most reviewed/tweeted about products are Apple followed by Google products which is expected in the US market demographic.
- The accuracy of predicting the product user sentiment is at around `61%`, following the best modelling outcome, after optimizing through tuning. This meets our success metric of `60%` accuracy.

Recommendations

Based on the above conclusions, we recommend the following:

- The stakeholders should target majorly the users whose sentiments towards their products is negative so as to retain and further add onto their customer base.
- Our model will help the team to access faster and real time insights on product reviews, since they won't do this manually anymore.
- Knowing the number and actual positive sentiment tweets will enable the team to reinforce these positive attributes in advertising via consumer testimonials. This will serve to boost brand credibility and boost sales among new customers.
- Knowing the number and actual negative sentiment tweets will enable the team to know what product improvements to make, and to reach out to consumers with the issues to have them sorted out. This will reduce customer churn.

- We also expect that the team will have better customer engagement since they will be responding with strategies based on the voice of the customer. This will make customers feel heard and improve the loyalty and engagement with the brand.

Next Steps

- Try to use neural networks and transfer learning to improve accuracy of the model.
- Deploy the model to enable the marketing team to have an interface to work with.
- Develop dashboards to enable real time display of insights gleaned from customer`s product reviews/tweets.

Note on our Next steps

- We found some time to also run another model(a neural network model) -RoBERT(Robustly Optimized BERT approach) is a state-of-the-art natural language processing (NLP) model- on our data this model achieved an accuracy of `~66%`
- code for that model is included here- ROBERTA-MODEL (roberta.ipynb)