In [1]:
```python
import re
import numpy as np
import pandas as pd
import seaborn as sns
import nltk
import matplotlib.pyplot as plt
from os.path import normpath
from operator import itemgetter
from functools import partial

# Set Seaborn theme and default palette
sns.set_theme(font_scale=1, style="darkgrid")
sns.set_palette("deep", desat=0.85, color_codes=True)

# Turn on inline plotting
%matplotlib inline

# Load Black auto-formatter
%load_ext nb_black

# Enable automatic reloading
%load_ext autoreload
%autoreload 2
```

In [2]:
```python
from sklearn.dummy import DummyClassifier
from sklearn.preprocessing import (
    StandardScaler,
    RobustScaler,
    MinMaxScaler,
    MaxAbsScaler,
    PowerTransformer,
    QuantileTransformer,
    FunctionTransformer,
)
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import (
    LogisticRegression,
    LogisticRegressionCV,
    SGDClassifier,
    RidgeClassifier,
    RidgeClassifierCV,
    PassiveAggressiveClassifier,
)
from sklearn.feature_extraction.text import (
    CountVectorizer,
    HashingVectorizer,
    TfidfTransformer,
    TfidfVectorizer,
)
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.pipeline import make_pipeline, Pipeline, FeatureUnion
from gensim.parsing.preprocessing import STOPWORDS
```

In [3]:
```python
# Import my modules
from tools import cleaning, plotting, language as lang, utils
from tools.modeling.vectorizers import Doc2Vectorizer
from tools.modeling.transformers import ArrayForcer, PandasWrapper
from tools.modeling.classification import diagnostics as diag

# Set my default MPL settings
```

```
plt.rcParams.update(plotting.MPL_DEFAULTS)

# RandomState for reproducibility
rando = np.random.RandomState(9547)
```

Overview of Dataset

In [4]:
```
df = pd.read_csv(normpath("data/crowdflower_tweets.csv"))
df.head()
```

Out[4]:

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

Looks like one text feature and two categorical features, one of which has a lot of null values. The feature names are very long and wordy, presumably to reflect the actual language used by CrowdFlower in crowdsourcing this dataset. I'm going to rename those before I do anything else.

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9093 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column                                              Non-Null Count  Dtype 
---  ------                                              --------------  ----- 
 0   tweet_text                                          9092 non-null   object
 1   emotion_in_tweet_is_directed_at                     3291 non-null   object
 2   is_there_an_emotion_directed_at_a_brand_or_product  9093 non-null   object
dtypes: object(3)
memory usage: 213.2+ KB
```

Cleaning

Renaming

In [6]:
```
# Assign new column names
df.columns = ["text", "object_of_emotion", "emotion"]
df.head()
```

Out[6]:

	text	object_of_emotion	emotion
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion

	text	object_of_emotion	emotion
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative emotion
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive emotion

Next, I take a look at the values of the categorical variables. The categories make sense, although the names are longer than necessary. I'm going to shorten some of them as well.

In [7]:
```
cleaning.show_uniques(df)
```

object_of_emotion	emotion
iPhone	Negative emotion
iPad or iPhone App	Positive emotion
iPad	No emotion toward brand or product
Google	I can't tell
Android	
Apple	
Android App	
Other Google product or service	
Other Apple product or service	

First, I convert the categorical columns to `CategoricalDtype`. This will make it easier to rename the categories, and is a convenient way to differentiate the categorical features from the text column.

In [8]:
```
# Convert categorical columns to categorical dtype
cat_cols = ["emotion", "object_of_emotion"]
df[cat_cols] = df.loc[:, cat_cols].astype("category")

# Delete temp variable
del cat_cols

# Display results
display(df["emotion"].head(3), df["object_of_emotion"].head(3))
```

```
0    Negative emotion
1    Positive emotion
2    Positive emotion
Name: emotion, dtype: category
Categories (4, object): ['I can't tell', 'Negative emotion', 'No emotion toward brand or product',
'Positive emotion']
0                iPhone
1    iPad or iPhone App
2                  iPad
Name: object_of_emotion, dtype: category
Categories (9, object): ['Android', 'Android App', 'Apple', 'Google', ..., 'Other Google product or
service', 'iPad', 'iPad or iPhone App', 'iPhone']
```

Next, I rename the categories for both categorical features.

I use a single `dict` mapping old category names to new ones. I only need one `dict` for both features because the method `Series.cat.rename_categories(...)` ignores irrelevant keys.

In [9]:

```
# Create mapping of old categories to new ones
new_cats = {
    # New 'emotion' categories
    "Negative emotion": "Negative",
    "Positive emotion": "Positive",
    "No emotion toward brand or product": "Neutral",
    "I can't tell": "Uncertain",
    # New 'object_of_emotion' categories
    "iPad or iPhone App": "iOS App",
    "Other Google product or service": "Other Google Product",
    "Other Apple product or service": "Other Apple Product",
}

# Rename categories in-place (ignores irrelevant keys)
df["emotion"].cat.rename_categories(new_cats, inplace=True)
df["object_of_emotion"].cat.rename_categories(new_cats, inplace=True)

# Delete renaming dict
del new_cats

# Show results
cleaning.show_uniques(df)
```

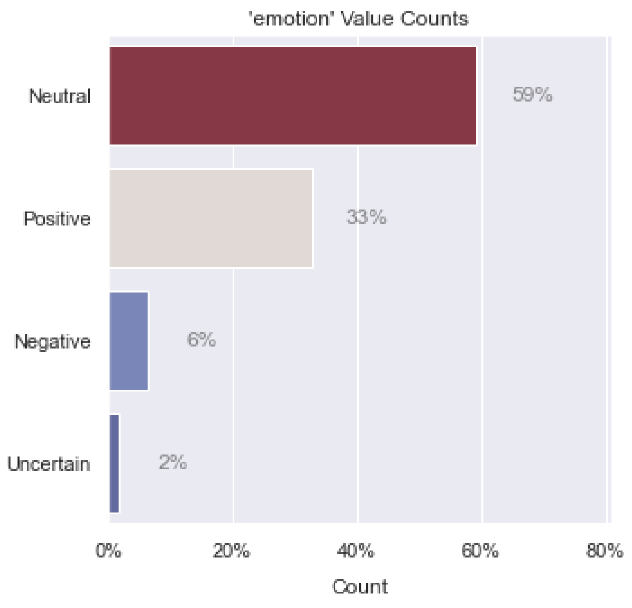
object_of_emotion	emotion
iPhone	Negative
iOS App	Positive
iPad	Neutral
Google	Uncertain
Android	
Apple	
Android App	
Other Google Product	
Other Apple Product	

In [10]:

```
plotting.countplot(df["emotion"], normalize=True)
```

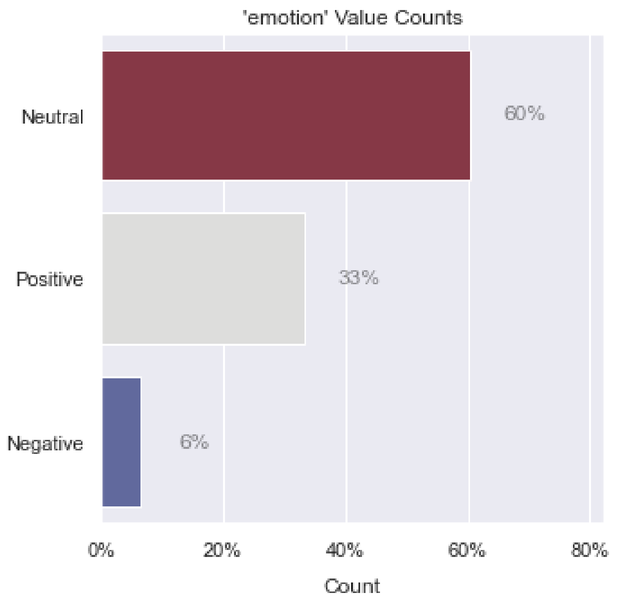
Out[10]:

```
<AxesSubplot:title={'center':"'emotion' Value Counts"}, xlabel='Count'>
```



In [11]:
```
# Remove 'Uncertain' category
df.emotion.cat.remove_categories("Uncertain", inplace=True)
plotting.countplot(df.emotion, normalize=True)
```

Out[11]:
```
<AxesSubplot:title={'center':"'emotion' Value Counts"}, xlabel='Count'>
```



Missing Values

According to the table below, there are a lot of missing values in the 'object_of_emotion' category. I bet, however, that these NaN values correspond to the 'Neutral' category. If a tweet doesn't express a brand-emotion, then there shouldn't be any brand in the 'object_of_emotion' column.

There's also one null 'text' row, and a bunch of null 'emotion' rows where the 'Uncertain' category used to be.

In [12]:
```
cleaning.info(df)
```

Out[12]:

	null	null_%	uniq	uniq_%	dup	dup_%

	null	null_%	uniq	uniq_%	dup	dup_%
object_of_emotion	5802	63.81	9	0.10	22	0.24
emotion	156	1.72	3	0.03	22	0.24
text	1	0.01	9065	99.69	22	0.24

I'll go ahead and drop the nulls in the 'text' and 'emotion' columns first.

In [13]:
```
df.dropna(subset=["text", "emotion"], inplace=True)
cleaning.info(df)
```

Out[13]:

	null	null_%	uniq	uniq_%	dup	dup_%
object_of_emotion	5654	63.27	9	0.10	22	0.25
text	0	0.00	8909	99.70	22	0.25
emotion	0	0.00	3	0.03	22	0.25

In [14]:
```
null_rows = cleaning.null_rows(df)
lang.readable_sample(null_rows["text"], random_state=rando)
```

	text
5140	RT @mention @mention New iPad Apps For Speech Therapy And Communication Are Showcased At #SXSW Conference {link} #sxswi #hcsm #sxswh
509	Please RT Follow the next big #college social network @mention chance to win an #iPad at 7,000 followers #socialmedia #SXSW
4916	millions of iPhone cases at #SXSW trade show but can any of them double as shuffleboard wax sprinklers? I think not. #fail (CC @mention
6384	RT @mention not launching any products at #SXSW but we're doing plenty else. {link}
790	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw"
8793	Google giving Social another go? {link} Google Circles, let's see what the guys at #SXSW make of it
8452	@mention The unofficial #SXSW torrents are a great way to hear what you can expect this year {link}
3645	U gotta fight for yr right to party & to privacy ACLU/google #sxsw #partylikeits1986
61	#futuremf @mention {link} spec for recipes on the web, now in google search: {link} #sxsw
4081	Hope people ask the tough questions. RT @mention Reminder: Android and Chrome TTS talk @mention 1 PM today! {link} #sxsw

Looks like some of the NaN values don't line up with the 'Neutral' category.

In [15]:
```
emotion_without_object = null_rows.loc[null_rows.emotion != "Neutral"]

# Delete variable
del null_rows

display(emotion_without_object.head(), emotion_without_object.shape)
```

	text	object_of_emotion	emotion
46	Hand-Held ���Hobo�: Drafthouse launches ���Ho...	NaN	Positive

	text	object_of_emotion	emotion
64	Again? RT @mention Line at the Apple store is ...	NaN	Negative
68	Boooo! RT @mention Flipboard is developing an ...	NaN	Negative
103	Know that &quot;dataviz&quot; translates to &q...	NaN	Negative
112	Spark for #android is up for a #teamandroid aw...	NaN	Positive

```
(357, 3)
```

In [16]:

```python
lang.readable_sample(
    emotion_without_object.groupby("emotion").get_group("Positive").text,
    random_state=rando,
)
```

	text	
3353	Whoohoo! Got it! ;) RT @mention New #UberSocial for #iPhone now in the App Store includes UberGuide to #SXSW (cont) {link}	
3928	dancing with myself at google 80s party.... ain't that the truth! need my girl @mention up in this joint #SXSW {link}	
783	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw rt @mention via @mention	
1365	@mention - re: "lack of #SXSW newsworthy announcements". Unless you count Google Circles. :) #googlecircles	
5307	RT @mention #SXSW News: Apple is getting into the music business? New device called an "iPod". Like a compact disc player without the disk.	
2447	Near Field Communication already here on android phones. #SXSW #bemyneighbor	
7530	#spiltbeer consequences of drunk techies #sxsw let's see what android has to offer compared to this.	
5965	RT @mention Having fun w/ @mention new Check-In's feature on iPhone	See @mention latest article "Roll your own 4square" {link} #SXSW
3442	Back in the big apple! Need to wean off my new foursquare addiction thanks to #sxsw and @mention Do people really care where I am? Nah	
3780	FYI @mention is working on an iPhone app, looking to release it this summer, hopefully. #SXSW #flipboard	

In [17]:

```python
# Create regex for finding each brand
re_apple = r"ipad\d?\s*app|ipad\d?|iphone\s*app|iphone|apple"
re_google = r"android\s*app|android|google"

# Find all brand/product name occurrences for each brand
findings = lang.locate_patterns(
    re_apple,
    re_google,
    docs=emotion_without_object["text"],
    exclusive=True,
    flags=re.I,
)

# Convert to lowercase
findings = findings.str.lower()

# View results
display(
    findings.value_counts(),
    findings.size,
)
```

```
google         122
ipad            98
apple           76
iphone          57
ipad2           26
android         19
iphone app       8
ipad app         4
ipad1            1
android app      1
Name: locate_patterns, dtype: int64
412
```

In [18]:

```
# Rename Apple apps to match categories defined previously
findings = findings.str.replace(
    r"ipad\s+app|iphone\s+app", "ios app", case=False, regex=True
)

# Fuzzy match with previously defined categories
findings = lang.fuzzy_match(findings, df["object_of_emotion"].cat.categories)

# View results
findings.sort_values("score")
```

Out[18]:

	original	match	score
5401	ipad2	iPad	89
3179	ipad2	iPad	89
8149	ipad2	iPad	89
6309	ipad2	iPad	89
3710	ipad2	iPad	89
...	...	...	...
3224	ipad	iPad	100
3179	ipad	iPad	100
3134	google	Google	100
3055	ipad	iPad	100
9054	ipad	iPad	100

412 rows × 3 columns

In [19]:

```
# Define sort order, i.e. fill priority
order = [
    "iOS App",
    "Android App",
    "iPhone",
    "iPad",
    "Android",
    "Apple",
    "Google",
]

# Sort values in reverse order
```

```
utils.explicit_sort(
    findings,
    order=order,
    by="match",
    ascending=False,
    inplace=True,
)

# Fill in reverse, overwriting lower priority values
for i, brand in findings.match.items():
    df.at[i, "object_of_emotion"] = brand
df.loc[findings.index].sample(10, random_state=rando)
```

Out[19]:

	text	object_of_emotion	emotion
598	CNNMoney: Got a craving? #SXSW minds created a...	iPhone	Positive
5401	RT @mention Anyone at #sxsw want to make a qui...	iPad	Positive
5212	RT @mention #Apple saves #SXSW, set to open po...	Apple	Positive
639	Catch 22��_ I mean iPad 2 at #SXSW - {link} #a...	iPad	Positive
5586	RT @mention Buying iPad2? Turn in ur iPad1, Ap...	iPad	Positive
8898	@mention What's the wait time lookin like? The...	Apple	Positive
6371	RT @mention Nice! @mention just told me @menti...	iPad	Positive
7680	Google (tries again) to launch a new social ne...	Google	Negative
1813	3rd time a charm? All about privacy! RT @menti...	Google	Positive
1284	Trying to update software (4.0) on iPhone to d...	iPhone	Negative

In [20]:

```
# Get indices which were not filled
emotion_without_object.drop(findings.index, inplace=True)

# Drop unfilled observations
df.drop(emotion_without_object.index, inplace=True)

print(f"{emotion_without_object.shape[0]} observations dropped.")

del emotion_without_object
```

```
24 observations dropped.
```

In [21]:

```
object_without_emotion = df.loc[
    (df.emotion == "Neutral") & df.object_of_emotion.notnull()
]
display(object_without_emotion.head(), object_without_emotion.shape)
```

	text	object_of_emotion	emotion
63	#Smile RT @mention I think Apple's &quot;pop-u...	Apple	Neutral
265	The #SXSW Apple &quot;pop-up&quot; store was n...	Apple	Neutral
317	I arrived at #sxsw and my @mention issue hasn'...	iOS App	Neutral
558	haha. the google &quot;Party like it's 1986&qu...	Google	Neutral
588	Diller on Google TV: &quot;The first product w...	Other Google Product	Neutral

```
(91, 3)
```

Tweet 6517 seems clearly negative to me, and 7137 seems kind of sardonic. 2666 seems weakly positive. 8647, 5696, 7521, 668, and 265 don't seem to express an emotion toward a brand or product. Since most of them seem neutral to me, and that's consistent with their 'Neutral' label, I'm going to keep them that way.

```
In [22]: lang.readable_sample(object_without_emotion["text"], random_state=rando)
```

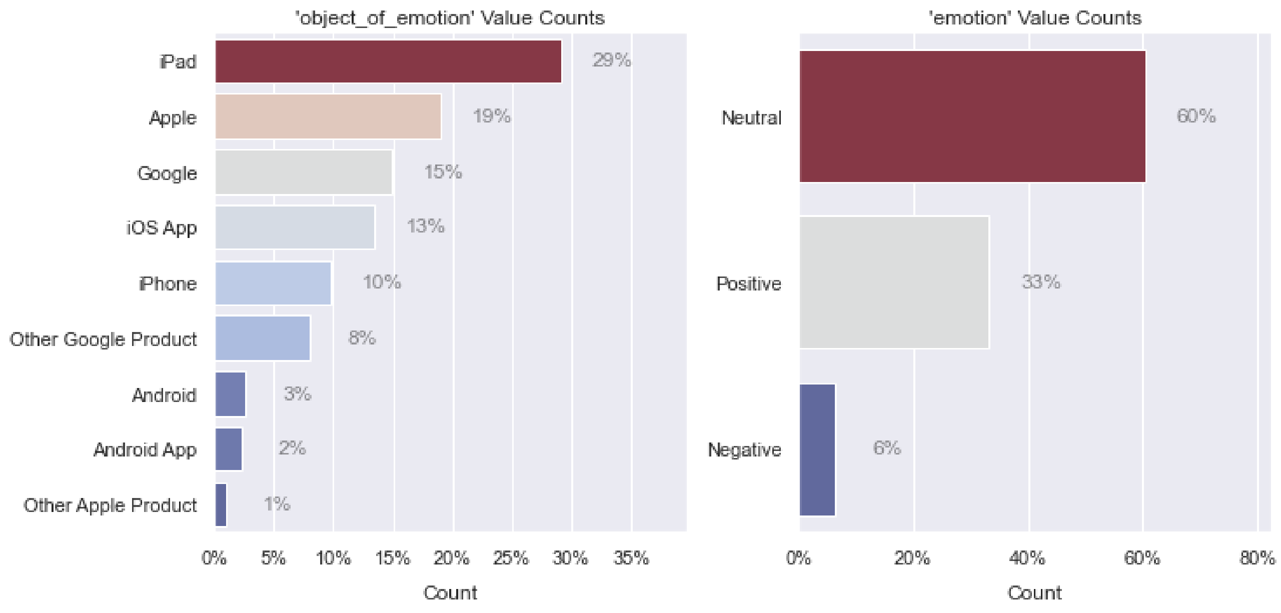
	text
6127	RT @mention In iPad Design Headaches: Take Two Tablets, Call Me in the AM panel - excited to hear @mention live! #sxsw
8271	Google Hot Pot - what the whattt pot....#SXSW
1628	@mention @mention Similarily, Tweetcaster for Android lets you zip tweets w annoying hash tags, like #sxsw
6517	RT @mention RT @mention Best thing I've heard this weekend at #SXSW "I gave my iPad 2 money to #Japan relief. I don't need an iPad 2." (@mention
8849	We can't wait to give an iPad to someone at #sxsw. Want in? Just head to www.pep.jobs/upc to enter. (must be present to win)
3011	#iPad interface/controls should be at the top, mobile controls at the bottom. #tapworthy #sxsw
787	Google to Launch Major New Social Network Called Circles, Possibly Today {link} #sxsw�💬 @mention
4826	Talking to a 15 year old iPhone developer genius who came to SXSW with his proud dad. #SXSW @mention BD Riley's Irish Pub {link}
1276	Apple has two Austin-area retail locations but in anticipation of all the gadget ... #ipad #sxsw #gadgets {link}
7173	ipad is not a game changer just a new game, its a about multiplicity and options not either or, but someone has to p[ay for content! #sxsw

```
In [23]: # Set object to null where emotion is neutral
         df.loc[object_without_emotion.index, "object_of_emotion"] = np.nan

         # Ensure that 'Neutral' rows line up with 'NaN' rows
         (df["emotion"] == "Neutral").equals(df["object_of_emotion"].isnull())
```

```
Out[23]: True
```

```
In [24]: fig = plotting.countplot(df.select_dtypes("category"), normalize=1)
```



There are 22 duplicate rows, but I'm not going to remove them yet. I'll dynamically remove duplicates as part of my preprocessing pipeline so I can catch as many as possible.

In [25]:
```
cleaning.dup_rows(df).sort_values("text")
```

Out[25]:

	text	object_of_emotion	emotion
3962	#SXSW is just starting, #CTIA is around the co...	Android	Positive
468	Before It Even Begins, Apple Wins #SXSW {link}	Apple	Positive
2559	Counting down the days to #sxsw plus strong Ca...	Apple	Positive
776	Google to Launch Major New Social Network Call...	NaN	Neutral
8483	I just noticed DST is coming this weekend. How...	iPhone	Negative
2232	Marissa Mayer: Google Will Connect the Digital...	NaN	Neutral
8747	Need to buy an iPad2 while I'm in Austin at #s...	iPad	Positive
4897	Oh. My. God. The #SXSW app for iPad is pure, u...	iOS App	Positive
5884	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5882	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5881	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5883	RT @mention Google to Launch Major New Social ...	NaN	Neutral
5885	RT @mention Google to Launch Major New Social ...	NaN	Neutral
6297	RT @mention Marissa Mayer: Google Will Connect...	NaN	Neutral
6299	RT @mention Marissa Mayer: Google Will Connect...	NaN	Neutral
6296	RT @mention Marissa Mayer: Google Will Connect...	Google	Positive
6298	RT @mention Marissa Mayer: Google Will Connect...	Google	Positive
6300	RT @mention Marissa Mayer: Google Will Connect...	NaN	Neutral
6546	RT @mention RT @mention Google to Launch Major...	NaN	Neutral
5338	RT @mention ��� GO BEYOND BORDERS! ��_ {link} ...	NaN	Neutral
5341	RT @mention ��� Happy Woman's Day! Make love, ...	NaN	Neutral

	text	object_of_emotion	emotion
3950	Really enjoying the changes in Gowalla 3.0 for...	Android App	Positive

In [26]:
```
funcs = [
    lang.lowercase,
    lang.strip_short,
    partial(lang.strip_punct, exclude="@#!?"),
    lang.strip_multiwhite,
    lang.strip_numeric,
    lang.strip_non_alphanum,
    lang.split_alphanum,
    lang.uni2ascii,
    lang.stem_text,
    lang.strip_handles,
    lang.limit_repeats,
    lang.wordnet_lemmatize,
    lang.stem_text,
]

func_names = utils.get_func_names(funcs)

funcs = [FunctionTransformer(func=x) for x in funcs]

funcs = pd.Series(dict(zip(func_names, funcs)))
funcs
```

Out[26]:
```
lowercase             FunctionTransformer(func=<function lowercase a...
strip_short           FunctionTransformer(func=<function strip_short...
strip_punct           FunctionTransformer(func=functools.partial(<fu...
strip_multiwhite      FunctionTransformer(func=<function strip_multi...
strip_numeric         FunctionTransformer(func=<function strip_numer...
strip_non_alphanum    FunctionTransformer(func=<function strip_non_a...
split_alphanum        FunctionTransformer(func=<function split_alpha...
uni2ascii             FunctionTransformer(func=<function uni2ascii a...
stem_text             FunctionTransformer(func=<function stem_text a...
strip_handles         FunctionTransformer(func=<function strip_handl...
limit_repeats         FunctionTransformer(func=<function limit_repea...
wordnet_lemmatize     FunctionTransformer(func=<function wordnet_lem...
dtype: object
```

In [27]:
```
filt_pipe = [
    "lowercase",
    "uni2ascii",
    "limit_repeats",
    "strip_punct",
    "split_alphanum",
    "strip_short",
    "strip_multiwhite",
]

filt_pipe = list(zip(filt_pipe, funcs.loc[filt_pipe].to_list()))
filt_pipe = PandasWrapper(Pipeline(filt_pipe))
filt_pipe.fit_transform(df.text)
```

Out[27]:
```
0       @wesley have iphone after hrs tweeting #rise a...
1       @jessedee know about @fludapp awesome ipad iph...
2       @swonderlin can not wait for #ipad also they s...
3       @sxsw hope this year festival isn crashy this ...
4       @sxtxstate great stuff fri #sxsw marissa mayer...
                              ...
9088                          ipad everywhere #sxsw link
9089    wave buzz @mention interrupt your regularly sc...
9090    google zeiger physician never reported potenti...
```

```
9091    some verizon iphone customers complained their...
9092       @mention google tests check offers@ #sxsw link
Name: text, Length: 8912, dtype: object
```

In [28]:
```python
df["clean_text"] = filt_pipe.fit_transform(df.text)
df.head()
```

Out[28]:

	text	object_of_emotion	emotion	clean_text
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative	@wesley have iphone after hrs tweeting #rise a...
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iOS App	Positive	@jessedee know about @fludapp awesome ipad iph...
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive	@swonderlin can not wait for #ipad also they s...
3	@sxsw I hope this year's festival isn't as cra...	iOS App	Negative	@sxsw hope this year festival isn crashy this ...
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive	@sxtxstate great stuff fri #sxsw marissa mayer...

In [29]:
```python
re_brand = fr"{re_apple}|{re_google}"
regex_brands = lang.locate_patterns(re_brand, docs=df.clean_text)
regex_brands = utils.implode(regex_brands).reindex_like(df)
df["brand_terms"] = regex_brands
del regex_brands
df["brand_terms"].head()
```

Out[29]:
```
0               [iphone]
1     [ipad, iphone app]
2                 [ipad]
3           [iphone app]
4               [google]
Name: brand_terms, dtype: object
```

In [30]:
```python
df["tokens"] = df.clean_text.map(nltk.casual_tokenize)
df["tokens"].head()
```

Out[30]:
```
0    [@wesley, have, iphone, after, hrs, tweeting, ...
1    [@jessedee, know, about, @fludapp, awesome, ip...
2    [@swonderlin, can, not, wait, for, #ipad, also...
3    [@sxsw, hope, this, year, festival, isn, crash...
4    [@sxtxstate, great, stuff, fri, #sxsw, marissa...
Name: tokens, dtype: object
```

In [31]:
```python
df["tagged"] = df.tokens.map(nltk.pos_tag)
df["tagged"].head()
```

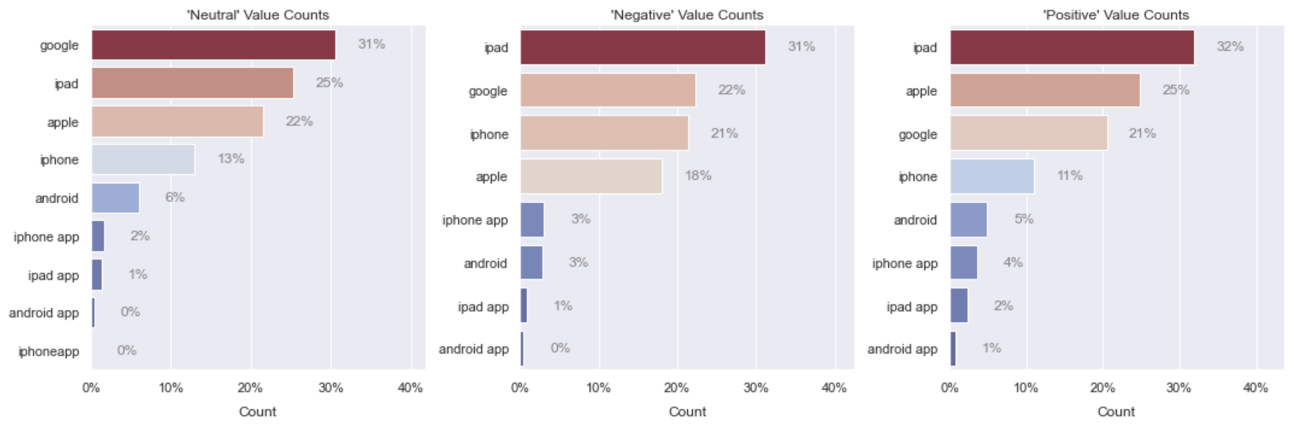
Out[31]:
```
0    [(@wesley, NNS), (have, VBP), (iphone, VBN), (...
1    [(@jessedee, NN), (know, VBP), (about, IN), (@...
2    [(@swonderlin, NNS), (can, MD), (not, RB), (wa...
3    [(@sxsw, RB), (hope, NN), (this, DT), (year, N...
4    [(@sxtxstate, JJ), (great, JJ), (stuff, NN), (...
Name: tagged, dtype: object
```

In [32]:
```python
df["pos_tags"] = utils.implode(df["tagged"].explode().map(itemgetter(1)))
df["pos_tags"].head()
```

```
Out[32]: 0    [NNS, VBP, VBN, IN, NN, VBG, NN, NN, VBD, JJ, ...
         1    [NN, VBP, IN, NNP, JJ, NN, NN, NN, IN, PRP, JJ...
         2    [NNS, MD, RB, VB, IN, NN, RB, PRP, MD, NN, PRP...
         3     [RB, NN, DT, NN, NN, NN, NN, DT, NN, NN, NN, NN]
         4    [JJ, JJ, NN, NN, NNP, NN, NN, NN, NN, RB, JJ, ...
         Name: pos_tags, dtype: object
```

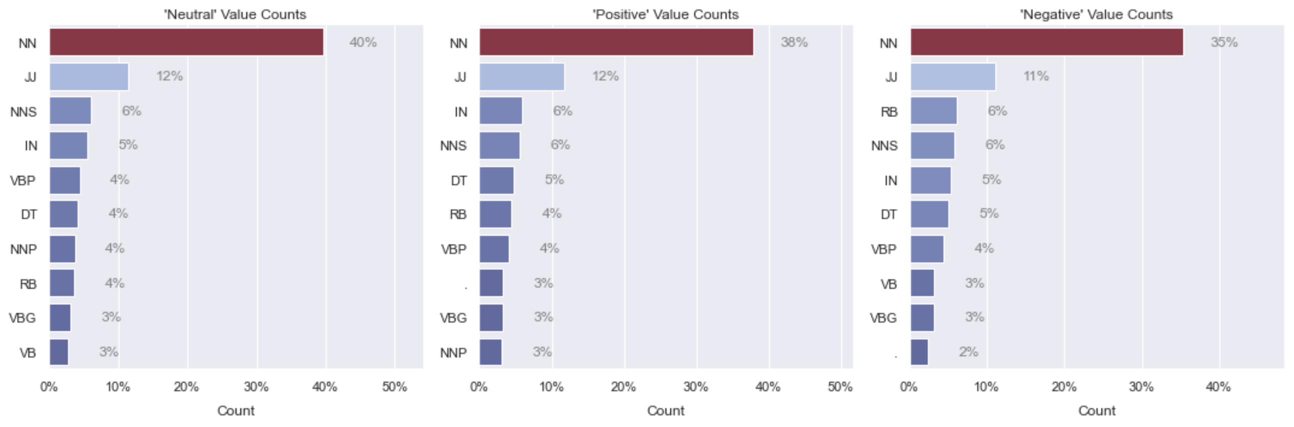
Exploration

```
In [33]: fig = plotting.countplot(
             df.explode("brand_terms").groupby("emotion")["brand_terms"],
             normalize=True,
         )
```



```
In [34]: grouped = df.explode("pos_tags").groupby("emotion")["pos_tags"]

         fig = plotting.countplot(grouped, normalize=True, topn=10)
```



```
In [35]: # Try POS filtering
         # Try tfidf vectorizing with just POS tags
```

```
In [36]: stop_words = list(STOPWORDS) + ["sxsw", "#sxsw", "quot", "link"]
         stop_words.sort()
         print(stop_words)

         ['#sxsw', 'a', 'about', 'above', 'across', 'after', 'afterwards', 'again', 'against', 'all', 'almos
         t', 'alone', 'along', 'already', 'also', 'although', 'always', 'am', 'among', 'amongst', 'amoungs
        t', 'amount', 'an', 'and', 'another', 'any', 'anyhow', 'anyone', 'anything', 'anyway', 'anywhere',
        'are', 'around', 'as', 'at', 'back', 'be', 'became', 'because', 'become', 'becomes', 'becoming', 'b
        een', 'before', 'beforehand', 'behind', 'being', 'below', 'beside', 'besides', 'between', 'beyond',
        'bill', 'both', 'bottom', 'but', 'by', 'call', 'can', 'cannot', 'cant', 'co', 'computer', 'con', 'c
        ould', 'couldnt', 'cry', 'de', 'describe', 'detail', 'did', 'didn', 'do', 'does', 'doesn', 'doing',
        'don', 'done', 'down', 'due', 'during', 'each', 'eg', 'eight', 'either', 'eleven', 'else', 'elsewhe
        re', 'empty', 'enough', 'etc', 'even', 'ever', 'every', 'everyone', 'everything', 'everywhere', 'ex
        cept', 'few', 'fifteen', 'fifty', 'fill', 'find', 'fire', 'first', 'five', 'for', 'former', 'former
        ly', 'forty', 'found', 'four', 'from', 'front', 'full', 'further', 'get', 'give', 'go', 'had', 'ha
        s', 'hasnt', 'have', 'he', 'hence', 'her', 'here', 'hereafter', 'hereby', 'herein', 'hereupon', 'he
        rs', 'herself', 'him', 'himself', 'his', 'how', 'however', 'hundred', 'i', 'ie', 'if', 'in', 'inc',
        'indeed', 'interest', 'into', 'is', 'it', 'its', 'itself', 'just', 'keep', 'kg', 'km', 'last', 'lat
        ter', 'latterly', 'least', 'less', 'link', 'ltd', 'made', 'make', 'many', 'may', 'me', 'meanwhile',
        'might', 'mill', 'mine', 'more', 'moreover', 'most', 'mostly', 'move', 'much', 'must', 'my', 'mysel
        f', 'name', 'namely', 'neither', 'never', 'nevertheless', 'next', 'nine', 'no', 'nobody', 'none',
        'noone', 'nor', 'not', 'nothing', 'now', 'nowhere', 'of', 'off', 'often', 'on', 'once', 'one', 'onl
        y', 'onto', 'or', 'other', 'others', 'otherwise', 'our', 'ours', 'ourselves', 'out', 'over', 'own',
        'part', 'per', 'perhaps', 'please', 'put', 'quite', 'quot', 'rather', 're', 'really', 'regarding',
        'same', 'say', 'see', 'seem', 'seemed', 'seeming', 'seems', 'serious', 'several', 'she', 'should',
        'show', 'side', 'since', 'sincere', 'six', 'sixty', 'so', 'some', 'somehow', 'someone', 'somethin
        g', 'sometime', 'sometimes', 'somewhere', 'still', 'such', 'sxsw', 'system', 'take', 'ten', 'than',
        'that', 'the', 'their', 'them', 'themselves', 'then', 'thence', 'there', 'thereafter', 'thereby',
        'therefore', 'therein', 'thereupon', 'these', 'they', 'thick', 'thin', 'third', 'this', 'those', 't
        hough', 'three', 'through', 'throughout', 'thru', 'thus', 'to', 'together', 'too', 'top', 'toward',
        'towards', 'twelve', 'twenty', 'two', 'un', 'under', 'unless', 'until', 'up', 'upon', 'us', 'used',
        'using', 'various', 'very', 'via', 'was', 'we', 'well', 'were', 'what', 'whatever', 'when', 'whenc
        e', 'whenever', 'where', 'whereafter', 'whereas', 'whereby', 'wherein', 'whereupon', 'wherever', 'w
        hether', 'which', 'while', 'whither', 'who', 'whoever', 'whole', 'whom', 'whose', 'why', 'will', 'w
        ith', 'within', 'without', 'would', 'yet', 'you', 'your', 'yours', 'yourself', 'yourselves']
```

In [37]:
```python
brand_docs = (
    pd.Series(df.groupby(["emotion", "object_of_emotion"]).groups)
    .map(lambda x: df.loc[x, "clean_text"])
    .map(lambda x: " ".join(x))
)

brand_docs = brand_docs.drop(index=np.nan, level=0)
brand_docs
```

Out[37]:
```
Negative  Android                 they took away the lego pit but replaced with ...
          Android App             beware the android #sxsw app for schedules com...
          Apple                   again? @mention line the apple store insane #s...
          Google                  @mention false alarm google circles not coming...
          Other Apple Product     @mention meant itunes doesn work for don run a...
          Other Google Product    @mention google launch major new social networ...
          iOS App                 @sxsw hope this year festival isn crashy this ...
          iPad                    attending @mention ipad design headaches #sxsw...
          iPhone                  @wesley have iphone after hrs tweeting #rise a...
Neutral   NaN                     @teachntech new ipad apps for #speechtherapy a...
Positive  Android                 #sxsw just starting #ctia around the corner an...
          Android App             find amp start impromptu parties #sxsw with @h...
          Apple                   counting down the days #sxsw plus strong canad...
          Google                  @sxtxstate great stuff fri #sxsw marissa mayer...
          Other Apple Product     pedicab iphone charger would epic win #sxsw pu...
          Other Google Product    gotta love this #sxsw google calendar featurin...
          iOS App                 @jessedee know about @fludapp awesome ipad iph...
          iPad                    @swonderlin can not wait for #ipad also they s...
          iPhone                  love @mention iphone case from #sxsw but can g...
dtype: object
```

In [38]:
```python
tfidf = TfidfVectorizer(
    tokenizer=partial(nltk.casual_tokenize, strip_handles=True),
    stop_words=stop_words,
    max_features=None,
    ngram_range=(1, 3),
    norm="l2",
)
brand_vecs = tfidf.fit_transform(brand_docs.values)
brand_vecs = lang.frame_doc_vecs(
    brand_vecs,
    tfidf.vocabulary_,
    brand_docs.index,
).T
brand_vecs
```

Out[38]:

									Negative	Neutral	
	Android	**Android App**	**Apple**	**Google**	**Other Apple Product**	**Other Google Product**	**iOS App**	**iPad**	**iPhone**	**NaN**	**Andro**
!	0.04388	0.025603	0.054796	0.062841	0.0	0.041581	0.073591	0.040762	0.135289	0.171880	0.2073
! !	0.00000	0.000000	0.005163	0.010996	0.0	0.016977	0.022535	0.000000	0.011047	0.013888	0.0274
! ! !	0.00000	0.000000	0.000000	0.005793	0.0	0.008943	0.007914	0.000000	0.005820	0.004413	0.0072
! ! #angrybirds	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000565	0.0000
! ! #apple	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
...	...	...	...	...	...	...	...	...	...	...	
zynga facebook	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.002262	0.0000
zynga facebook microsoft	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.002262	0.0000
zzzs	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.014167	0.000000	0.0000
zzzs iphone	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.014167	0.000000	0.0000
zzzs iphone battery	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.014167	0.000000	0.0000

104142 rows × 19 columns

In [39]:

```
brand_vecs = brand_vecs.sort_index(1)
brand_vecs
```

Out[39]:

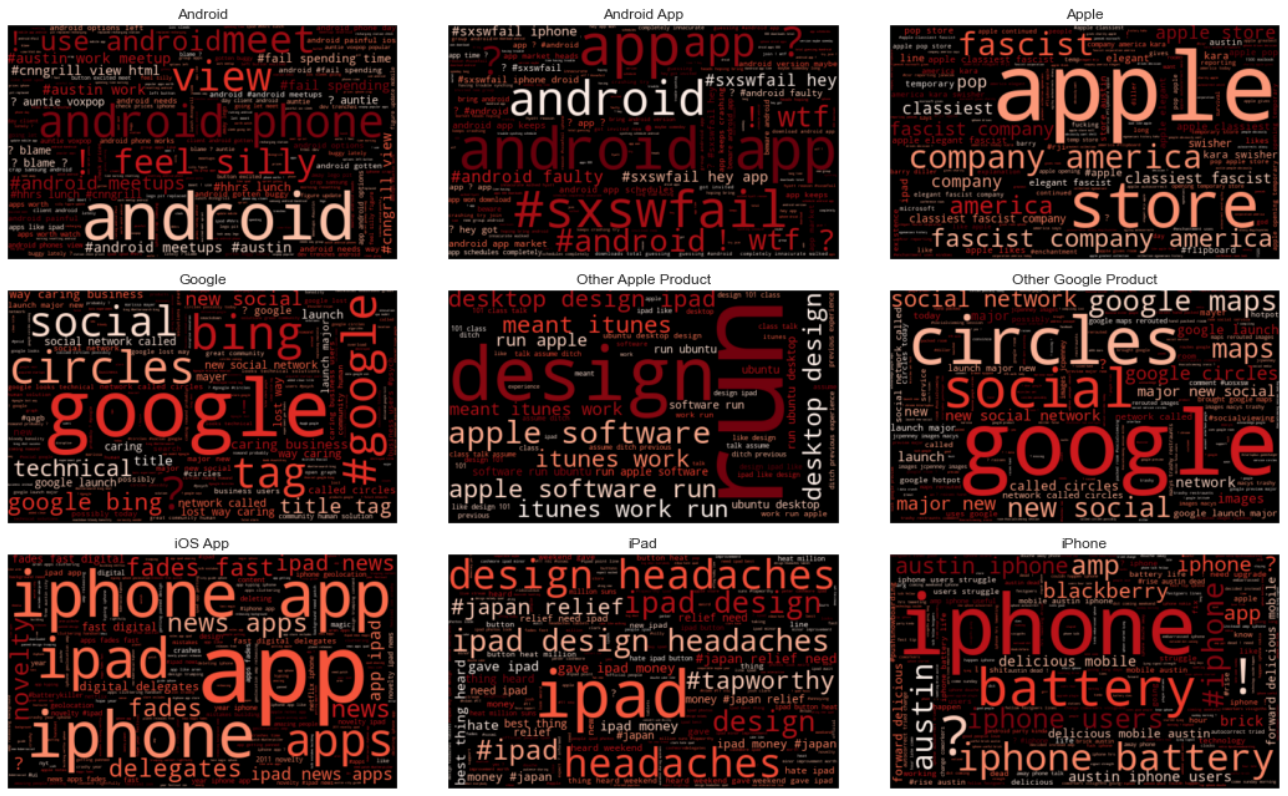
									Negative	Neutral	
	Android	**Android App**	**Apple**	**Google**	**Other Apple Product**	**Other Google Product**	**iOS App**	**iPad**	**iPhone**	**NaN**	**Andro**
!	0.04388	0.025603	0.054796	0.062841	0.0	0.041581	0.073591	0.040762	0.135289	0.171880	0.2073
! !	0.00000	0.000000	0.005163	0.010996	0.0	0.016977	0.022535	0.000000	0.011047	0.013888	0.0274
! ! !	0.00000	0.000000	0.000000	0.005793	0.0	0.008943	0.007914	0.000000	0.005820	0.004413	0.0072
! ! #angrybirds	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000565	0.0000
! ! #apple	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
...	...	...	...	...	...	...	...	...	...	...	
zynga facebook	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.002262	0.0000

										Negative	Neutral	
	Android	Android App	Apple	Google	Other Apple Product	Other Google Product	iOS App	iPad	iPhone	NaN	Andro
zynga facebook microsoft	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.002262	0.0000
zzzs	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.014167	0.000000	0.0000
zzzs iphone	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.014167	0.000000	0.0000
zzzs iphone battery	0.00000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.014167	0.000000	0.0000

104142 rows × 19 columns

In [40]:

```
fig = plotting.wordcloud(brand_vecs.loc[:, "Negative"], cmap="Reds", random_state=rando)
fig.savefig("negative.png")
```



In [41]:

```
fig = plotting.wordcloud(
    brand_vecs.loc[:, "Positive"], cmap="Greens", random_state=rando
)
fig.savefig("positive.png")
```

Android
Android App
Apple
Google
Other Apple Product
Other Google Product
iOS App
iPad
iPhone

Modeling

Train-Test-Split

In [42]:
```
X = df["text"].to_numpy()
y = df.emotion.to_numpy()
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    random_state=rando,
    stratify=y,
    shuffle=True,
)
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[42]:
```
((6684,), (6684,), (2228,), (2228,))
```

Baseline Dummy Model

In [43]:
```
dummy = DummyClassifier(strategy="stratified", random_state=rando)
dummy
```

Out[43]:
```
DummyClassifier(random_state=RandomState(MT19937) at 0x23DF344AC40,
                strategy='stratified')
```

In [44]:
```
tfidf.set_params(
    lowercase=True,
    max_features=300,
    tokenizer=nltk.casual_tokenize,
    token_pattern=None,
    ngram_range=(1, 2),
    stop_words=None,
)
```

Out[44]:
```
TfidfVectorizer(max_features=300, ngram_range=(1, 2), token_pattern=None,
                tokenizer=<function casual_tokenize at 0x0000023DEFEC44C0>)
```

In [45]:
```
main_pipe = Pipeline(
    [
        ("filt", None),
        ("stem", None),
        ("vec", tfidf),
        ("sca", None),
        ("cls", dummy),
    ]
)
main_pipe
```

Out[45]:
```
Pipeline(steps=[('filt', None), ('stem', None),
                ('vec',
                 TfidfVectorizer(max_features=300, ngram_range=(1, 2),
                                 token_pattern=None,
                                 tokenizer=<function casual_tokenize at 0x0000023DEFEC44C0>)),
                ('sca', None),
                ('cls',
                 DummyClassifier(random_state=RandomState(MT19937) at 0x23DF344AC40,
                                 strategy='stratified'))])
```

In [46]:
```
vecs = main_pipe[:-2].fit_transform(X_train)
display(vecs)
vecs = lang.frame_doc_vecs(vecs, tfidf.vocabulary_)
vecs
```

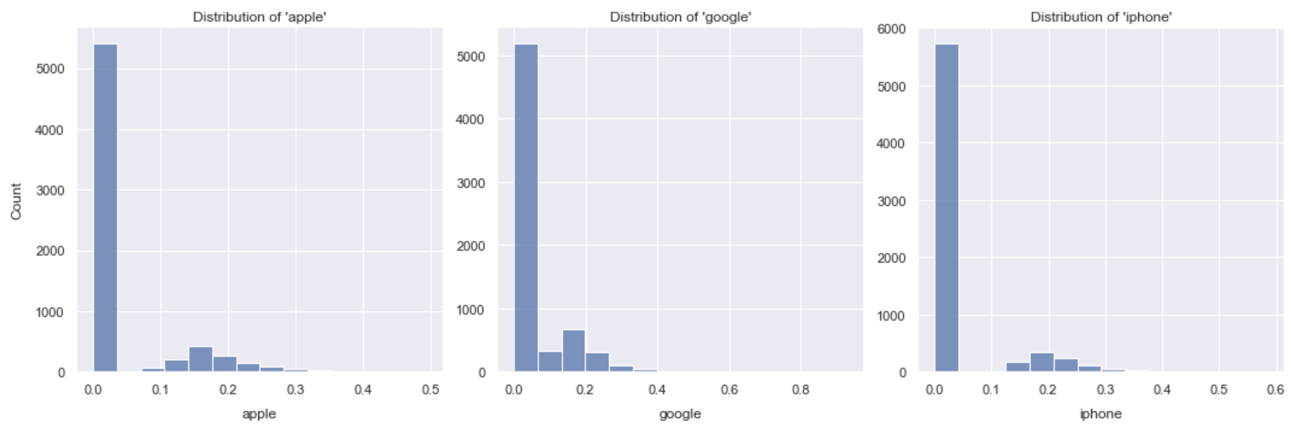
```
<6684x300 sparse matrix of type '<class 'numpy.float64'>'
	with 117966 stored elements in Compressed Sparse Row format>
```

Out[46]:

	!	! !	! #sxsw	! rt	! {	"	#android	#apple	#austin	#google	...	{ link	}	}
0	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	...	0.098219	0.098183	0.
1	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	...	0.156812	0.156754	0.
2	0.412489	0.541492	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	...	0.093418	0.093383	0.
3	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	...	0.079545	0.079516	0.
4	0.413061	0.542243	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6679	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.
6680	0.000000	0.000000	0.0	0.0	0.0	0.273935	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.
6681	0.167749	0.000000	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	...	0.000000	0.000000	0.
6682	0.000000	0.000000	0.0	0.0	0.0	0.394321	0.0	0.000000	0.0	0.0	...	0.101288	0.101251	0.
6683	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.288618	0.0	0.0	...	0.124422	0.124377	0.

6684 rows × 300 columns

In [47]:
```
fig = plotting.multi_dist(data=vecs[["apple", "google", "iphone"]])
```



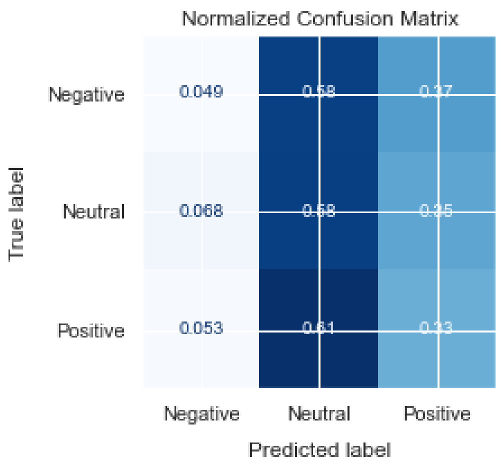
In [48]:
```
high_corr = diag.high_correlations(vecs, thresh=0.8)
print(f"Found {high_corr.size} high correlations.")
high_corr.head()
```

```
Found 56 high correlations.
```

Out[48]:
```
(            )               0.882495
, possibly  called circles    0.807647
            circles ,         0.855776
            launch major      0.853018
            major             0.816668
dtype: float64
```

In [49]:
```
test_fit = partial(
    diag.test_fit,
    X_train=X_train,
    X_test=X_test,
    y_train=y_train,
    y_test=y_test,
)
test_fit(main_pipe)
```

	Negative	Neutral	Positive	macro avg	weighted avg	accuracy	bal accuracy
precision	0.067	0.601	0.319	0.329	0.473	0.462	0.329
recall	0.077	0.571	0.338	0.329	0.462		
f1-score	0.072	0.585	0.328	0.329	0.467		
support	0.064	0.605	0.332				



Baseline Model

In [50]:
```
logit = LogisticRegression(
    class_weight="balanced",
    multi_class="multinomial",
    solver="lbfgs",
    max_iter=1e4,
    verbose=0,
    random_state=rando,
)
logit
```

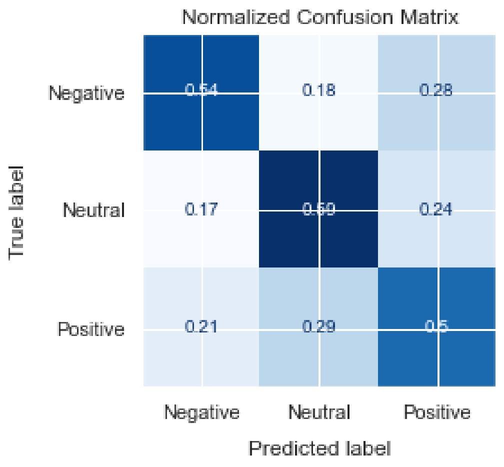
Out[50]:
```
LogisticRegression(class_weight='balanced', max_iter=10000.0,
                   multi_class='multinomial',
                   random_state=RandomState(MT19937) at 0x23DF344AC40)
```

In [51]:
```
main_pipe.set_params(cls=logit)
```

Out[51]:
```
Pipeline(steps=[('filt', None), ('stem', None),
                ('vec',
                 TfidfVectorizer(max_features=300, ngram_range=(1, 2),
                                 token_pattern=None,
                                 tokenizer=<function casual_tokenize at 0x0000023DEFEC44C0>)),
                ('sca', None),
                ('cls',
                 LogisticRegression(class_weight='balanced', max_iter=10000.0,
                                    multi_class='multinomial',
                                    random_state=RandomState(MT19937) at 0x23DF344AC40))])
```

In [52]:
```
test_fit(main_pipe)
```

	Negative	Neutral	Positive	macro avg	weighted avg	accuracy	bal accuracy
precision	0.166	0.770	0.507	0.481	0.645	0.559	0.545
recall	0.542	0.593	0.499	0.545	0.559		
f1-score	0.255	0.670	0.503	0.476	0.588		
support	0.064	0.605	0.332				



Model Mark II

In [53]:
```
funcs
```

Out[53]:
```
lowercase             FunctionTransformer(func=<function lowercase a...
strip_short           FunctionTransformer(func=<function strip_short...
strip_punct           FunctionTransformer(func=functools.partial(<fu...
strip_multiwhite      FunctionTransformer(func=<function strip_multi...
strip_numeric         FunctionTransformer(func=<function strip_numer...
strip_non_alphanum    FunctionTransformer(func=<function strip_non_a...
split_alphanum        FunctionTransformer(func=<function split_alpha...
uni2ascii             FunctionTransformer(func=<function uni2ascii a...
stem_text             FunctionTransformer(func=<function stem_text a...
strip_handles         FunctionTransformer(func=<function strip_handl...
limit_repeats         FunctionTransformer(func=<function limit_repea...
wordnet_lemmatize     FunctionTransformer(func=<function wordnet_lem...
dtype: object
```

In [56]:
```
filt_pipe = [
    "lowercase",
    "uni2ascii",
    "limit_repeats",
    "strip_punct",
    "split_alphanum",
    "strip_short",
    "strip_handles",
    "strip_multiwhite",
]

filt_pipe = Pipeline(list(zip(filt_pipe, funcs.loc[filt_pipe].to_list())))
filt_pipe
```

Out[56]:
```
Pipeline(steps=[('lowercase',
                 FunctionTransformer(func=<function lowercase at 0x0000023DF33B04C0>)),
                ('uni2ascii',
                 FunctionTransformer(func=<function uni2ascii at 0x0000023DF33B0A60>)),
                ('limit_repeats',
                 FunctionTransformer(func=<function limit_repeats at 0x0000023DF33B0820>)),
                ('strip_punct',
                 FunctionTransformer(func=functools.partial(<function strip_punct at 0x0000...
                ('split_alphanum',
                 FunctionTransformer(func=<function split_alphanum at 0x0000023DF33B0790>)),
                ('strip_short',
                 FunctionTransformer(func=<function strip_short at 0x0000023DF33B0550>)),
                ('strip_handles',
                 FunctionTransformer(func=<function strip_handles at 0x0000023DF33B09D0>)),
                ('strip_multiwhite',
                 FunctionTransformer(func=<function strip_multiwhite at 0x0000023DF33B05E0>))])
```

In [ ]:
```
main_pipe = Pipeline(
    [
        ("filt", None),
        ("stem", None),
        ("vec", tfidf),
        ("sca", None),
        ("cls", logit),
    ]
)
main_pipe
```

In [ ]:
```
from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold

cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=10, random_state=rando)

grid = dict(filt=[funcs.limit_repeats, funcs.lowercase, funcs.strip_punct])
search = GridSearchCV(
    main_pipe,
    param_grid=grid,
    scoring="recall_weighted",
    cv=cv,
    n_jobs=1,
)
search
```

In [ ]:

```
search.fit(X_train, y_train)
```

In [ ]:

```
pd.DataFrame(search.cv_results_)
```

In [ ]:

```
test_fit(main_pipe)
```

In [ ]:

```
filt_pipe = [
    "lowercase",
    "unidecode",
    "limit_repeats",
    "strip_punct",
    "split_alphanum",
    "strip_short",
    "remove_handles",
    "strip_multiwhite",
]

filt_pipe = Pipeline(list(zip(filt_pipe, funcs.loc[filt_pipe].to_list())))
filt_pipe
```

In [ ]:

```
d2v = Doc2Vectorizer(
    tokenizer=nltk.casual_tokenize,
    token_pattern=None,
    stop_words=stop_words,
    ngram_range=(1, 2),
    n_features=300,
    epochs=40,
    workers=3,
    seed=48,
)
classify_pipe = Pipeline(
    [
        ("filt", filt_pipe),
        ("stem", FunctionTransformer(stem_text)),
        ("vec", d2v),
        ("dense", ArrayForcer(force_dense=True)),
        ("scale", StandardScaler()),
        ("clas", logit),
    ]
)
classify_pipe
```

In [ ]:

```

```

In [ ]:

```
classify_pipe["vec"].set_params(dbow_words=0)
classify_pipe.fit(X_train, y_train)
diag.standard_report(classify_pipe, X_test, y_test)
```

In [ ]:

```
classify_pipe.set_params(vec=tfidf, vec__max_features=300)
classify_pipe.fit(X_train, y_train)
diag.standard_report(classify_pipe, X_test, y_test)
```

In [ ]:
```
diag.classification_report(y_test, classify_pipe.predict(X_test), heatmap=True)
```

In [ ]:
```
classify_pipe.set_params(clas=DummyClassifier(strategy="stratified"))
classify_pipe.fit(X_train, y_train)
plot_confusion_matrix(classify_pipe, X_test, y_test, cmap="Blues", normalize="true")
```

In [ ]:
```
from sklearn.pipeline import FeatureUnion


double_vec = FeatureUnion(
    [
        ("d2v", d2v),
        ("tfidf", tfidf),
    ]
)
double_vec
```

In [ ]: