1 Project 4 – NLP Model – Work Notebook

- Student name: Greg Osborne
- Student pace: self paced / part time
- Scheduled project review date/time: 3/23/23
- Instructor name: Morgan Jones
- Blog post URL: https://medium.com/@gregosborne (https://medium.com/@gregosborne)

1.1 Work Document

This document only exists to experiment with different visualizations after Word2Vec Vectorization takes place. This started as a copy of student.ipynb. All text explaining the process is deleted for brevity's sake. This text can still be found in the student.ipynb notebook.

2 Table of Contents

Project 4 – NLP Model

3 Python Libraries

In [1]:
```
# DataFrames and computation
import pandas as pd
import numpy as np

# Graphing
import matplotlib.pyplot as plt
import seaborn as sns
from plotly import graph_objs as go
from collections import Counter
%matplotlib inline
import plotly.express as px
from palettable.colorbrewer.qualitative import Pastel1_7

# Import the Tokenize library
from nltk.tokenize import RegexpTokenizer

# For Frequency Distributions
from nltk import FreqDist
from matplotlib.ticker import MaxNLocator

# For Wordclouds
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator

# For stopwords
import nltk
from nltk.corpus import stopwords

# For stemming words
from nltk.stem.snowball import SnowballStemmer

# The Train/Test Split
from sklearn.model_selection import train_test_split

# Import the TfidfVectorizer class
from sklearn.feature_extraction.text import TfidfVectorizer

# Bag of Words
from sklearn.feature_extraction.text import CountVectorizer

# Word2Vec
import gensim

# For Min/Max Scaling
from sklearn.preprocessing import MinMaxScaler

# Scores
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_score, recall_score, accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report

# Multinomial Naive Bayes
from sklearn.naive_bayes import MultinomialNB

# Logistic Regression Model Classifier
from sklearn.linear_model import LogisticRegression
```

```
# Support Vector Machine
from sklearn import svm

# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

# For XGBoost Classifier
from xgboost import XGBClassifier

# The Confusion Matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

# For Hyperparameter Tuning
from sklearn.model_selection import GridSearchCV

# Setting DataFrame Display settings
pd.set_option("display.max_rows", 600)

# Formatting decimals to show three numbers after the point.
pd.options.display.float_format = '{:,.3f}'.format
```

4 Data Cleaning

4.1 Loading the Data

In [2]:
```
df_raw = pd.read_csv("judge-1377884607_tweet_product_company.csv",
                     encoding='ANSI')
```

4.1.1 Rename Columns

In [3]:
```
def rename_columns(dforg):
    name = {
        'tweet_text': 'Tweet',
        'emotion_in_tweet_is_directed_at': 'Product',
        'is_there_an_emotion_directed_at_a_brand_or_product': 'Emotion'
        }
    dfone = dforg.copy()
    dfone = dfone.rename(columns=name)
    return dfone.copy()
df = rename_columns(df_raw)
```

In [4]:
```
# The labels for the emotions need to change to short labels for graphing
# purposes.
def abbreviate_emotion(dfone):
    for i in range(len(dfone)):
        if dfone.loc[i, 'Emotion'] == 'No emotion toward brand or product':
            dfone.loc[i, 'Emotion'] = 'Neutral'
        elif dfone.loc[i, 'Emotion'] == 'Negative emotion':
            dfone.loc[i, 'Emotion'] = 'Negative'
        elif dfone.loc[i, 'Emotion'] == 'Positive emotion':
            dfone.loc[i, 'Emotion'] = 'Positive'
        elif dfone.loc[i, 'Emotion'] == "I can't tell":
            dfone.loc[i, 'Emotion'] = 'Indecipherable'
    return dfone.copy()

df = abbreviate_emotion(df)
```

4.2 Preparing Data For Analysis

4.2.1 Standardizing Case

In [5]:
```
df['Tweet'] = df['Tweet'].str.lower()
```

4.2.2 Testing for bad data

In [6]:
```
# Testing for any Tweets that are not strings
bad_tweets = []
for i in range(len(df)):
    typ = type(df.loc[i, 'Tweet'])
    if typ != str:
        print('Index number {} is missing a value in the'.format(i),
              'Tweet column.')
        bad_tweets.append(i)
```

```
Index number 6 is missing a value in the Tweet column.
```

In [7]:
```
# Dropping bad data and reseting the index
df.drop(index=bad_tweets, inplace=True)
df.reset_index(drop=True, inplace=True)
```

4.2.3 Combining compound nouns

In [8]:
```
# This function searches the tweets for keys in a given dictionary and replace
# them with the associated value.
# df = The dataframe to search
# begin_replace = There are some tweets that begin with someone's last name,
#                 this requires separate coding to insert their first name
#                 before their last, so this dictionary will specify those
#                 compound nouns that need this treatment.
# replace_dict  = The dictionary with keys and values that turns more than one
#                 word into compound nouns with underscores.

def string_replace(dfone, begin_replace, replace_dict, to_print = True):
    counts = pd.DataFrame(columns = ['Replaced'])
    for n in range(len(dfone)):
        tweet = dfone.loc[n,'Tweet']
        for fnd, replce in zip(begin_replace.keys(), begin_replace.values()):
            if fnd in tweet[0:len(fnd)]:
                tweet = replce + tweet[len(fnd):]
                replced = fnd + ' / ' + replce
                counts.loc[len(counts),'Replaced'] = replced
                dfone.loc[n,'Tweet'] = ''+tweet
        for find, replace in zip(replace_dict.keys(), replace_dict.values()):
            if find in tweet:
                change = False
                first = tweet[0:len(find)]
                last = tweet[len(tweet) - len(find):]
                if (find in first):
                    change = True
                    tweet = replace + tweet[len(find):]
                    replaced = find + ' / ' + replace
                    counts.loc[len(counts),'Replaced'] = replaced
                if (find in last):
                    change = True
                    tweet = tweet[:len(tweet)-len(find)] + replace
                    replaced = find + ' / ' + replace
                    counts.loc[len(counts),'Replaced'] = replaced
                complete = 0
                while complete < len(tweet) - len(find):
                    for i in range(0, len(tweet) - len(find)):
                        if find in tweet[i:i+len(find)]:
                            change = True
                            first = tweet[:i]
                            last = tweet[i+len(find):]
                            tweet = first + replace + last
                            complete = 0
                            replaced = find + ' / ' + replace
                            counts.loc[len(counts),'Replaced'] = replaced
                            break
                        else:
                            complete += 1
                if change == True:
                    dfone.loc[n,'Tweet'] = ''+tweet
    if to_print == True:
        if counts.empty:
            print('No replacements made.')
```

```
    return dfone.copy()
```

In [9]:
```
first_replace = {'mayer'                : 'marissa_mayer',
                 'vinh'                 : 'khoi_vinh'}

replacements =  {'apple store'          : 'apple_store',
                 'applestore'           : 'apple_store',
                 'apple_stores'         : 'apple_store',
                 'app store'            : 'app_store',
                 'appstore'             : 'app_store',
                 'marissa mayer'        : 'marissa_mayer',
                 'marissa meyer'        : 'marissa_mayer',
                 'marissameyer'         : 'marissa_mayer',
                 'marissamayer'         : 'marissa_mayer',
                 'marisa meyer'         : 'marissa_mayer',
                 'marissa@mention'      : 'marissa_mayer',
                 'melissa mayer'        : 'marissa_mayer',
                 'merissa mayer'        : 'marissa_mayer',
                 'm mayer'              : 'marissa_mayer',
                 'm.mayer'              : 'marissa_mayer',
                 '-mayer'               : 'marissa_mayer',
                 '#mayer'               : '#marissa_mayer',
                 ' mayer'               : 'marissa_mayer',
                 'marissa_mayers'       : 'marissa_mayer',
                 "marissa_mayer's"      : 'marissa_mayer',
                 'mentionmarissa_mayer' : 'mention marissa_mayer',
                 '#mayer'               : '#marissa_mayer',
                 "tim o'reilly"         : "tim_o_reilly",
                 'tim oreilly'          : "tim_o_reilly",
                 "tim_o'reilly's"       : "tim_o_reilly",
                 "tim_o'reillys"        : "tim_o_reilly",
                 "'re"                  : ' are',
                 'tim o areily'         : "tim_o_reilly",
                 'tim soo'              : 'tim_soo',
                 'tim ferris'           : 'tim_ferris',
                 'tim_ferriss'          : 'tim_ferris',
                 'tim wu'               : 'tim_wu',
                 'matt mullenweg'       : 'matt_mullenweg',
                 'matt_mullenwegs'      : 'matt_mullenweg',
                 "matt_mullenweg's"     : 'matt_mullenweg',
                 'jonathan dahl'        : 'jonathan_dahl',
                 'mark belinsky'        : 'mark_belinsky',
                 'maggie mae'           : 'maggie_mae',
                 'maggie may'           : 'maggie_mae',
                 "maggie_mae's"         : 'maggie_mae',
                 "maggie_maes"          : 'maggie_mae',
                 "mike tyson"           : 'mike_tyson',
                 "mike_tyson's"         : 'mike_tyson',
                 "mike_tysons"          : 'mike_tyson',
                 "tyson's"              : 'mike_tyson',
                 'matt damon'           : 'matt_damon',
                 'barry diller'         : 'barry_diller',
                 'nyt'                  : 'ny_times',
                 'new york times'       : 'ny_times',
                 'ny times'             : 'ny_times',
                 "can't"                : 'can not',
                 "won't"                : 'would not',
                 "n't"                  : ' not',
                 "'ve"                  : ' have',
```

```
                    "'ll"                   : ' will',
                    'steve jobs'            : 'steve_jobs',
                    'pop up'                : 'pop_up',
                    'popup'                 : 'pop_up',
                    'pop-up'                : 'pop_up',
                    'pop shop'              : 'pop_up shop',
                    'pop-uitp'              : 'pop_up',
                    'pop- up'               : 'pop_up',
                    'pop-u‰û_'              : 'pop_up',
                    'pop-store'             : 'pop_up store',
                    'pop store'             : 'pop_up store',
                    'wi-fi'                 : 'wifi',
                    'wi fi'                 : 'wifi',
                    '.com'                  : 'com',
                    'dennis crowley'        : 'dennis_crowley',
                    ' crowley'              : 'dennis_crowley',
                    'i-phone'               : 'iphone',
                    'i-pad'                 : 'ipad',
                    'ipads'                 : 'ipad',
                    'iphones'               : 'iphone',
                    'ipad 2'                : 'ipad2',
                    'ipad_2'                : 'ipad2',
                    'ipad2s'                : 'ipad2',
                    'ipad 1'                : 'ipad1',
                    'ipad1s'                : 'ipad1',
                    'iphone app'            : 'iphone_app',
                    'ipad app'              : 'ipad_app',
                    'andoid'                : 'android',
                    'droid app'             : 'droid_app',
                    'iphone_app_store'      : 'iphone app_store',
                    'ipad_app_store'        : 'ipad app_store',
                    'droid_app_store'       : 'droid app_store',
                    'iphone_apps'           : 'iphone_app',
                    'ipad_apps'             : 'ipad_app',
                    'iphone_application'    : 'iphone application',
                    'droid_apps'            : 'droid_app',
                    ' droid_app'            : ' android_app',
                    'ûªll'                  : ' will',
                    "‰ûªs"                  : "'s",
                    "‰ûªre"                 : ' are',
                    "ûªt"                   : ' not',
                    'khoi vinh'             : 'khoi_vinh',
                    'jonathan ive'          : 'jonathan_ive',
                    'winsåêsxsw'            : 'wins sxsw',
                    'matt carlson'          : 'matt_carlson',
                    'matthew davis'         : 'matthew_davis',
                    'william patry'         : 'william_patry',
                    'josh williams'         : 'josh_williams',
                    'david foster'          : 'david_foster',
                    'david carr'            : 'david_carr',
                    'û_please'              : 'please',
                    'adam beckley'          : 'adam_beckley',
                    'paul adams'            : 'paul_adams',
                    'googled'               : 'google_verb_d',
                    'google_verb_d'         : 'googled_verb'}

df = string_replace(df, first_replace, replacements)
```

4.2.4 Tokenize

In [10]:
```
# Creating the tokenizer
basic_token_pattern = r"(?u)\b\w\w+\b"
words_and_hashtags_token_pattern = r"(?u)\#?\b\w\w+\b"
hashtags_token_pattern = r"(?u)\#\b\w\w+\b"

tokenizer = RegexpTokenizer(basic_token_pattern)
hashtag_tokenizer = RegexpTokenizer(hashtags_token_pattern)
words_and_hashtags_tokenizer = RegexpTokenizer(words_and_hashtags_token_patter
```

In [11]:
```
# Creating a new column for the tokenized tweet
def tokenize_columns(dfone):
    dfone['Tokens'] = None
    dfone['Tokens With Hashtags'] = None
    dfone['Hashtags'] = None

    # Creating the token lists. One list will include tokens of each words, th
    # other will only include the words used as Hashtags.
    for i in range(len(dfone)):
        dfone['Tokens'][i] = tokenizer.tokenize(dfone['Tweet'][i])
        dfone['Tokens With Hashtags'
             ][i] = words_and_hashtags_tokenizer.tokenize(dfone['Tweet'][i])
        dfone['Hashtags'][i] = hashtag_tokenizer.tokenize(dfone['Tweet'][i])
    return dfone.copy()

df = tokenize_columns(df)
```

4.3 Purging Retweets Identical to Included Originals

4.3.1 The Retweet Purge Functions

In [12]:
```
# Creating a df of nothing but the retweets
def retweet_trim_one(dfone):
    for i in dfone.index:
        twt = dfone.loc[i, 'Tokens']
        # Twitter parlance: rt = retweet
        # Dropping all tweets that don't start in rt
        if 'rt' != twt[0]:
            dfone.drop(index=i, inplace=True)
    return dfone
```

In [13]:
```
# Creates a dictionary of specified keys with a count as the values, and then
# sorts the list from greatest to least in dictionary form.
def tally(lst):
    tally_dict = {}
    for item in lst:
        if item not in tally_dict:
            tally_dict[item] = 1
        else:
            tally_dict[item] += 1
    # Sorting the dictionaries
    count = sorted(tally_dict.items(), key=lambda x: x[1], reverse=True)
    return count
```

In [14]:
```
# Trimming the terms rt and mention from the beginning of the tweet, counting
# the first and second words of the remaining tweets, and determining which
# tweets to drop.

def retweet_trim_two(dftwo, drop_first_words=['skip'], drop_num=1):
    # Creating a column to put the trimmed retweet
    dftwo['trimmed_rt'] = 'Blank'
    dftwo['Match'] = False

    # I want to test the data to see if there's anything that stands out as
    # repetitive in the first or second words of the newly trimmed tweets.
    # Creating a list of the first and second words of each trimmed tweet.
    firstword_lst = []
    secondword_lst = []

    # Starting a list to drop any retweets.
    retweets_to_drop = []

    for i in dftwo.index:
        # Making the trimmed variable, the tokens of the next tweet to edit
        trimmed = dftwo.loc[i, 'Tokens']
        # Removing the set number of dropped tokens.
        # Default is just the first token, "rt."
        trimmed = trimmed[drop_num:]
        if len(trimmed) == 0:
            if drop_num == 1:
                retweets_to_drop.append(i)
                dftwo.loc[i,'Match'] = True
            continue
        # Several retweets include 'mention' after the rt, so I'll trim that.
        # This while loop quits dropping words with the first token that isn't
        # in the drop list specified by parameter.
        if drop_first_words != ['skip']:
            while trimmed[0] in drop_first_words:
                trimmed = trimmed[1:]
                # Some tweets contained only "rt" and "mention," and are now
                # empty."
                # Adding any empty token lists to the retweets_to_drop list.
                if len(trimmed) == 0:
                    if drop_num == 1:
                        retweets_to_drop.append(i)
                        dftwo.loc[i,'Match'] = True
                    break
        if len(trimmed) == 0:
            continue

        # Adding first and second words of the trimmed tweet to the lists.
        firstword_lst.append(trimmed[0])
        secondword_lst.append(trimmed[1])

        # Replacing the trimmed tokens to a new column, "trimmed_rt".
        dftwo.at[i,'trimmed_rt'] = trimmed

    # I want to test the data to see if there's anything that stands out as
    # repetitive in the first word of the newly trimmed tweets.
    # Creating a dictionary to count the first word in each trimmed tweet.
```

```
    firstword_count = tally(firstword_lst)
    secondword_count = tally(secondword_lst)

    return dftwo, firstword_count, secondword_count, retweets_to_drop
```

In [15]:
```
# Checking to see if any of theses newly trimmed retweets are exact matches of
# Original tweets in the dataset.


def retweet_trim_three(dfthree, dforg, retweets_to_drop):
    original_tweets = []

    for i in dforg.index:
        original_tweets.append(dforg.loc[i, 'Tokens'])

    for i in dfthree.index:
        retweet = dfthree.loc[i, 'trimmed_rt']
        if retweet in original_tweets:
            dfthree.loc[i, 'Match'] = True
            retweets_to_drop.append(i)

    return dfthree, retweets_to_drop
```

In [16]:
```
# Creating a single function that runs all of the previous functions but allow
# a list parameter to choose what words are filtered out of the first words.
# Also allows for deleting the duplicate tweets in one function.
def retweet_trim_all(retweets, word_list=['skip'], delete=True, drop=1):
    retweets = retweet_trim_one(retweets)
    retweets, fw, sw, rtd = retweet_trim_two(retweets, word_list, drop)
    retweets, rtd = retweet_trim_three(retweets, df, rtd)
    # Dropping the tweets.
    if delete == False:
        print(retweets['Match'].value_counts())
        return retweets.loc[retweets['Match'] == True]
    df.drop(index=rtd, inplace=True)
    # df.reset_index(drop=True, inplace=True)
    return retweets['Match'].value_counts()
```

4.3.2 Purge Retweets: Deleting Initial Words "rt" & "mention"

In [17]:
```
# Creating a copy of the dataframe for editing.
retweets = df[['Tweet', 'Tokens']].copy()

# Purging original tweets from the new copy, so that all that is left
# are retweets.
retweets = retweet_trim_one(retweets)

# Creating an initial count of all retweets
retweet_count = len(retweets)
```

In [18]:
```
# Trimming the retweets of the first "rt" and the subsequent word "mention."
retweets, fw, sw, rtd = retweet_trim_two(retweets, ['mention'])
```

In [19]:
```
# Checking to see if any of the trimmed retweets match any original tweets in
# the dataset.
retweets, rtd = retweet_trim_three(retweets, df, rtd)

# Variable that tells me how many matches I've found.
initial_purge = sum(retweets['Match'])

# Variable that tells me how many retweets di not find a match.
initial_remain = sum(~retweets['Match'])
```

In [20]:
```
# Dropping the duplicate retweets from the original dataset.
df.drop(index=rtd, inplace=True)
```

4.3.3 Purge Retweets: Removing all "rt" and "mention" Instances

In [21]:
```
# Before each run through these functions, I need to make a new copy of the
# retweets DafaFrame.
retweets = df[['Tweet', 'Tokens']].copy()

# Checking for multiple subsequent copies of both "rt" and "mention."
roundtwo = retweet_trim_all(retweets, ['rt', 'mention'])
deleted_two = roundtwo.loc[True]
```

In [22]:
```
# Checking for multiple subsequent copies of just "rt."
retweets = df[['Tweet', 'Tokens']].copy()
roundtwo = retweet_trim_all(retweets, ['rt'])
deleted_two += roundtwo.loc[True]
```

4.3.4 Purge Retweets: Removing A Set Number of Words

In [23]:
```
pd.options.display.max_colwidth = 1000
```

In [24]:
```
# Removing matches that occur when removing the first two through the first
# five words

# Variable to count the number of words deleted
deleted_three = 0

# Removing the second through fifth words, checking for matches, and deleting
# the matched retweets from the dataset.
# also, I'm keeping track of how many I delete this round.
for i in range(2, 6):
    retweets = df[['Tweet', 'Tokens']].copy()
    retweet_trim_all(retweets, drop=i)
    deleted_three += len(retweets.loc[retweets['Match'] == True])
```

4.4 Identifying Company

4.4.1 Gleaning Keywords From The Tweets

In [25]:
```
keywords_apple = ['Apple', 'iPad', 'iPad1', 'iPad2', 'iPhone', 'iTunes',
                  'iPhone5', 'iTune', 'Laptop', 'Apple_Store', 'App_Store',
                  'iOS', 'Macbook', 'iPad_App', 'iPhone_App', 'Pop_Up', 'iMac'

# Google did not make laptops in March 2011, so I'll assume all the laptops
# are Apple.

keywords_google = ['Google', 'Marissa_Mayer', 'MarissaGoogle', 'Googlecom',
                   'Android', 'Droid', 'Circles', 'Blogger', 'Chrome',
                   'Samsung', 'Galaxy', 'Maps', 'GoogleDoodle', 'ChromeOS',
                   'Googlecircles', 'Googletv', 'Android_App']
```

In [26]:
```
# Keywords that I'll use to identify what product the tweet discusses.

def keyword_labels(dfone, kw_apple, kw_google):
    kw_all = kw_apple + kw_google

    # A list of index numbers for tweets that do not include any of the keywor
    no_kw = []

    # Create a column for each keyword, to be purged later.
    for word in kw_all:
        dfone[word] = False

    # Labeling each tweet with what keywords they include.
    # Checks for the same word and the word with a hashtag.
    # The column will be labeled 'True' whether or not there's a hashtag.
    for i in dfone.index:
        for word in kw_all:
            if (word.lower() in dfone['Tokens'][i]) | (
                '#' + word.lower() in dfone['Tokens'][i]):
                dfone.loc[i, word] = True

        # appending no_kw with the index number of all rows
        # without any keywords.
        if sum(dfone.loc[i, kw_all].values) == 0:
            no_kw.append(i)

    return dfone.copy(), no_kw

df, nokeywords = keyword_labels(df, keywords_apple, keywords_google)
```

4.4.2 Identifying The Company By the Keywords

In [27]:
```
# This cell labels each row as "Apple", "Google", "Both" or "Unknown"
def company_organization(dfone, key_apple, key_google):
    dfone['Company'] = 'Blank'
    for i in dfone.index:
        # Checking for "Apple"
        for col in key_apple:
            if dfone[col][i]:
                dfone.loc[i, 'Company'] = 'Apple'
                # Checking for both Apple and Google
                for goog in key_google:
                    if dfone[goog][i]:
                        dfone.loc[i, 'Company'] = 'Both'
                        break
                break
        # Checking for "Google"
        if dfone.loc[i, 'Company'] not in ['Apple', 'Both']:
            for goog in key_google:
                if dfone[goog][i]:
                    dfone.loc[i, 'Company'] = 'Google'
                    break
        # Checking for Unknowns
        if dfone.loc[i, 'Company'] not in ['Apple', 'Both', 'Google']:
            dfone.loc[i, 'Company'] = 'Unknown'
    return dfone.copy()

df = company_organization(df, keywords_apple, keywords_google)
```

In [28]:
```
df.drop(index=df.loc[df['Company'] == 'Both'].index, inplace=True)
```

4.4.3 Comparing the Product Column To The Company Column

In [29]:
```
# Product column values that are Apple products.
Apple = [
    'iPad', 'Apple', 'iPad or iPhone App', 'iPhone',
    'Other Apple product or service'
]
# Product column values that are Google products.
Google = [
    'Google', 'Other Google product or service', 'Android App', 'Android'
]
```

In [30]:
```
# Creating the "Company by Product" column and populating it based on the
# Product column
def company_by_product(dfone):
    dfone['Company by Product'] = dfone['Product'].map(lambda x: 'Apple'
                                                       if x in Apple else np.nan)
    possgoog = dfone['Company by Product'] != 'Apple'

    dfone.loc[possgoog,
              'Company by Product'] = dfone.loc[possgoog, 'Product'].map(
        lambda x: 'Google' if x in Google else np.nan)
    return dfone.copy()

df = company_by_product(df)
```

In [31]:
```
# Checking which rows don't have matching values in columns "Company" and
# "Company by Product"
def no_match(dfone):
    dfone['Pd-Cp Match'] = "Match"
    for i in dfone.index:
        by_keyword = dfone.loc[i, 'Company']
        by_product = dfone.loc[i, 'Company by Product']
        if by_keyword != by_product:
            dfone.loc[i, 'Pd-Cp Match'] = str([by_keyword,
                                               by_product])[1:-1].replace("'", "")
    return dfone.copy()

df = no_match(df)
```

In [32]:
```
def mismatch(dfone):
    dfone.loc[2798,'Pd-Cp Match'] = 'Match'
    dfone.loc[2798, 'Company by Product'] = 'Google'
    return dfone.copy()

df = mismatch(df)
```

4.4.4 Tweets Without Keywords, with a *Product* Column Value

In [33]:
```
# Creating a DataFrame of just the tweets that don't have any of the key words
def no_keywords(dfone, nkw):
    cut = dfone.copy()
    for i in dfone.index:
        if i not in nkw:
            cut.drop(index=i, inplace=True)
    return cut.copy(), nkw

cut_tweets, nokeywords = no_keywords(df, nokeywords)
```

In [34]:
```
# Printing the tweets that don't have any of the keywords, but do have a given
# label in the Product column. Also, saving their index numbers in a list titl
# apple_tweets

def tweets_to_edit(nkw, cut, will_print = True):
    tweets = []
    for i in nkw:
        if cut.loc[i, 'Product'] in ['iPad or iPhone App', 'iPhone', 'Apple']:
            tweets.append(i)
    if will_print == True:
        to_print = cut.loc[tweets,['Tweet','Product']].copy()
        return tweets, to_print
    return tweets
apple_tweets, printing = tweets_to_edit(nokeywords, cut_tweets)
```

In [35]:
```
# This function hand labels a few tweets that have no keywords, but are clearl
# tweets about Apple apps.

def apple_tweet_edit(dfone, nkw, tweets):
    for i in tweets:
        nkw.remove(i)
        dfone.loc[i, 'Company'] = 'Apple'
        dfone.loc[i, 'Pd-Cp Match'] = 'Match'
        dfone.loc[i, 'iPhone_App'] = True
    return dfone.copy(), nkw

df, nokeywords = apple_tweet_edit(df, nokeywords, apple_tweets)
```

4.4.5 Reviewing the Uncategorized Tweets

In [36]:
```
unknowns = [670, 1860, 3859, 4925, 6901]
```

In [37]:
```
# The lists and dictionaries below are hand made after reading through all the
# tweets with unknown companies. This function properly labels them.
def corrections(dfone, nkw):
    unknowns = [670, 1860, 3859, 4925, 6901]

    for i in unknowns:
        nkw.remove(i)

    new_company_labels = {
        670: 'Google',
        1860: 'Google',
        3859: 'Apple',
        4925: 'Google',
        6901: 'Google'
        }

    # Labeling the companies
    new_product_labels = {
        670:  'Circles',
        1860: 'Circles',
        6901: 'Circles'
        }
    for i in new_company_labels.keys():
        dfone.loc[int(i), 'Company'] = new_company_labels[i]
        dfone.loc[int(i), new_company_labels[i]] = True

    # Labeling the products for df
    for num, label in zip(new_product_labels.keys(), new_product_labels.values
        dfone.loc[int(num), label] = True

    dfone.loc[unknowns, ['Apple', 'Google', 'Android', 'Circles', 'Company']]

    return dfone.copy(), nkw

df, nokeywords = corrections(df, nokeywords)
```

In [38]:
```
still_unknown = len(df.loc[df['Company'] == 'Unknown', ['Tweet', 'Company']])
```

In [39]:
```
# Dropping rows with no company known.
df.drop(index=df.loc[df['Company'] == 'Unknown'].index, inplace=True)
```

4.5 Identifying Brand or Product

In [40]:
```
# Create a column for both Apple Keywords and Google Keywords.

def categorize(dfone, kw_apple, kw_google):
    for i in dfone.index:
        # This variable will put each word in list format. I'll be able to
        # easily alphabetize the words in this form.
        keyword_lst = []
        # I want this cell to appear as a string, so I'll store the string her
        keyword_string = ''

        # This function picks the Apple or Google keywords and destination.
        if dfone.loc[i, 'Company'] == 'Apple':
            columns = kw_apple
            place = 'Apple Keyword'
        elif dfone.loc[i, 'Company'] == 'Google':
            columns = kw_google
            place = 'Google Keyword'
        else:
            continue

        # This for loop puts the keywords in the correct column and format.
        for col in columns:
            if dfone.loc[i, col] == True:
                keyword_lst.append(col)

        # Alphabetize the list
        # keyword_lst.sort()

        # Place the words in a string.
        for word in keyword_lst:
            keyword_string += word + ' '
        dfone.loc[i, place] = keyword_string[0:-1]
    return dfone

df = categorize(df, keywords_apple, keywords_google)
```

In [41]:
```
# This labels each tweet as either "Apple Brand", "Apple Products",
#  "Google Brand" or "Google Products".
def keyword_combos(dfone, kw_apple, kw_google):
    # Create lists of words that, if used exclusively, will categorize the
    # tweet as referring to a brand.
    apple_brand = ['Apple', 'Apple_Store', 'Pop_Up']
    google_brand = ['Google', 'Marissa_Mayer', 'MarissaGoogle', 'Googlecom',
                    'Circles', 'Googlecircles']

    # Create a list of keywords that, if used in the tweet, will supercede the
    # brand categorization, and categorize the tweet as a product.
    apple_products = kw_apple.copy()
    google_products = kw_google.copy()

    for prod in apple_products.copy():
        if prod in apple_brand:
            apple_products.remove(prod)

    for prod in google_products.copy():
        if prod in google_brand:
            google_products.remove(prod)

    # Categorizing the tweets
    for i in dfone.index:
        company = dfone.loc[i, 'Company']
        goog_prod_sum = sum(dfone.loc[i, google_products])
        apple_prod_sum = sum(dfone.loc[i, apple_products])
        is_it_a_product = goog_prod_sum + apple_prod_sum
        if (company == 'Apple') | (company == 'Google'):
            if is_it_a_product > 0:
                string = company + ' ' + 'Products'
            else:
                string = company + ' ' + 'Brand'
            dfone.loc[i, 'Keywords'] = dfone.loc[i, company + ' Keyword']
            dfone.loc[i, 'Category'] = string

    return dfone.copy()

df = keyword_combos(df, keywords_apple, keywords_google)
```

4.6 Cutting the DataFrame Columns

In [42]:
```
df = df.loc[:, ['Tweet', 'Emotion', 'Tokens', 'Tokens With Hashtags',
                'Hashtags', 'Keywords', 'Company', 'Category']].copy()
```

4.7 Creating Targets

In [43]:
```
# Dropping rows with no emotion known.
df.drop(index=df.loc[df['Emotion'] == "Indecipherable"].index, inplace=True)

# Resetting the dataset's index. (last deletion and reset)
df.reset_index(drop=True, inplace=True)
```

In [44]:
```
emotion_num = {'Negative': 0, 'Neutral': 1, 'Positive': 2}

category_num = {
    'Apple Brand': 0,
    'Apple Products': 3,
    'Google Brand': 6,
    'Google Products': 9
    }

def targets(dfone):
    # This for loop will label three columns.
    dfone['Target Text'] = ''
    dfone['Target'] = 0
    for i in range(len(dfone)):
        if (dfone.loc[i, 'Company'] != 'Unknown') & (
        dfone.loc[i, 'Company'] != 'Both') & (
        dfone.loc[i, 'Emotion'] != "Indecipherable"):
            dfone.loc[i,'Target Text'] = dfone.loc[
                      i, 'Emotion'] + ' - ' + dfone.loc[i, 'Category']
            dfone.loc[i, 'Target'] = (emotion_num[dfone.loc[i, 'Emotion']] +
                                      category_num[dfone.loc[i, 'Category']])
    return dfone.copy()

df = targets(df)
```

4.8 Removing Stopwords

In [45]:
```
# Downloading the standard stopword list
nltk.download('stopwords', quiet=True)

stopwords_list = stopwords.words('english')
```

In [46]:
```
# Additional stopwords
stopwords_list.extend(['mention', 'rt', 'quot', 'link', 'amp',
                       'ûó', 'ûò', 'ûï', 'û_', 'ã_', 'ûª', 'åç', 'åè', 'ä_',
                       'â_', 'dì'])
```

In [47]:
```
# Returns a token list without the list of specified stopwords.

def remove_stopwords(token_list, stop_list):
    """
    Given a list of tokens, return a list where the tokens
    that are also present in stopwords_list have been
    removed
    """
    stopwords_removed = [
        token for token in token_list if token not in stop_list
    ]
    return stopwords_removed
```

In [48]:
```
def trim_stopwords(dfone):
    dfone.insert(2, 'Tidy Tweet', '')
    dfone.insert(5, 'Tokens Without Stopwords', '')
    dfone["Tokens Without Stopwords"] = dfone[
        "Tokens"].map(lambda x : remove_stopwords(x, stopwords_list))
    for ind in dfone.index:
        dfone.loc[ind, 'Tidy Tweet'] = " ".join(i
                            for i in dfone.loc[ind, 'Tokens Without Stopwords'
    return dfone.copy()

df = trim_stopwords(df)
```

4.9 Stemmer Tokens

In [49]:
```
stemmer = SnowballStemmer(language="english")

def stem_and_tokenize(document):
    tokens = tokenizer.tokenize(document)
    return [stemmer.stem(token) for token in tokens]
```

In [50]:
```
# The stopwords themselves needed to be stemmed.
stemmed_stopwords = [stemmer.stem(word) for word in stopwords_list]
```

In [51]:
```
def stem_tokens(dfone):
    dfone.insert(3, 'Stemmed Tweet', '')
    dfone.insert(7, 'Stemmed Tokens', '')
    dfone["Stemmed Tokens"] = dfone['Tokens Without Stopwords'].apply(
        lambda x: [stemmer.stem(i) for i in x])
    dfone["Stemmed Tokens"] = dfone[
        "Stemmed Tokens"].map(lambda x : remove_stopwords(x,stemmed_stopwords)
    # The stemmer got overzealous with some of the keywords. This fixes that.
    fix_list = {'appl' : 'apple', 'googl' : 'google', 'iphon': 'iphone',
                'apple_stor' : 'apple_store', 'circl': 'circles',
                'marissa_may' : 'marissa_mayer', 'itun' : 'itunes',
                'app_stor' : 'app_store', 'io' : 'ios',
                'marissagoogl' : 'marissagoogle', 'map' : 'maps',
                'googledoodl' : 'googledoodle', 'chromeo' : 'chromeos',
                'googlecircl' : 'googlecircles'}
    for i in dfone["Stemmed Tokens"].index:
        for j in range(len(dfone.loc[i, "Stemmed Tokens"])):
            for fix in fix_list:
                if dfone.loc[i, "Stemmed Tokens"][j] == fix:
                    dfone.loc[i, "Stemmed Tokens"][j] = fix_list[fix]

    for ind in dfone.index:
        dfone.loc[ind, 'Stemmed Tweet'] = " ".join(i
                            for i in dfone.loc[ind, 'Stemmed Tokens'])
    return dfone.copy()

df = stem_tokens(df)
```

4.10 Frequency Distributions (Functions)

In [52]:
```
# Returns a visualization of the distribution of the top ten words in the
# specified data.
def visualize_top_10(freq_dist, title):

    # Extract data for plotting
    top_10 = list(zip(*freq_dist.most_common(10)))
    tokens = top_10[0]
    counts = top_10[1]

    # Set up plot and plot data
    fig, ax = plt.subplots()
    ax.bar(tokens, counts)

    # Customize plot appearance
    ax.set_title(title)
    ax.set_ylabel("Count")
    ax.yaxis.set_major_locator(MaxNLocator(integer=True))
    ax.tick_params(axis="x", rotation=90)
```

In [53]:
```
# Sets up twelve subplots. This is used with other functions.

def setup_twelve_subplots():
    fig = plt.figure(figsize=(9, 12))
    fig.set_tight_layout(True)
    gs = fig.add_gridspec(4, 3)
    ax1 = fig.add_subplot(gs[0, 0])
    ax2 = fig.add_subplot(gs[0, 1])
    ax3 = fig.add_subplot(gs[0, 2])
    ax4 = fig.add_subplot(gs[1, 0])
    ax5 = fig.add_subplot(gs[1, 1])
    ax6 = fig.add_subplot(gs[1, 2])
    ax7 = fig.add_subplot(gs[2, 0])
    ax8 = fig.add_subplot(gs[2, 1])
    ax9 = fig.add_subplot(gs[2, 2])
    ax10 = fig.add_subplot(gs[3, 0])
    ax11 = fig.add_subplot(gs[3, 1])
    ax12 = fig.add_subplot(gs[3, 2])
    return fig, [ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8, ax9, ax10, ax11, ax12
```

In [54]:
```
# Sets up twelve word frequency distributions for all target categories.
def plots(dataf, targ, column, axes, title="Word Frequency for"):
    ax_num = 0

    # Creating a DataFrame of just the target text with their target number as
    # the index.
    df_target = pd.DataFrame()

    for index, category in zip(dataf['Target'].unique(),
                               dataf['Target Text'].unique()):
        df_target.loc[index, 'Target Text'] = category

    df_target = df_target.sort_index()

    for index, category in zip(df_target.index, df_target.values):
        # Calculate frequency distribution for this subset
        all_words = dataf[targ == index][column].explode()
        freq_dist = FreqDist(all_words)
        top_10 = list(zip(*freq_dist.most_common(10)))
        tokens = top_10[0]
        counts = top_10[1]

        # Set up plot
        ax = axes[ax_num]
        ax.bar(tokens, counts)

        # Customize plot appearance
        ax.set_title(f"{title} \n {str(category)[2:-2]}")
        ax.set_ylabel("Count")
        ax.yaxis.set_major_locator(MaxNLocator(integer=True))
        ax.tick_params(axis="x", rotation=90)
        ax_num += 1
```

In [55]:
```
# Plots a single word cloud for the specified data.

def wc(tokens, title='', re = r"(?:\w[\w']+)"):
    if isinstance(tokens, pd.Series):
        text = " ".join(i for i in tokens.explode())
    else:
        text = " ".join(i for i in tokens)
    wordcloud = WordCloud(background_color="white", regexp = re).generate(text
    fig = plt.figure(figsize=(15, 10))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    fig.set_tight_layout(True)

    if title != '':
        fig.suptitle(title, fontsize=48)
    plt.show()
```

In [56]:
```
# Plots twelve word clouds for each specified target.

def word_cloud_grid(dataf, targ, column, title=''):
    tgt_lst = []
    wc_lst = []
    # Creating a DataFrame of just the target text with their target number as
    # the index.
    df_target = pd.DataFrame()
    for index, category in zip(dataf['Target'].unique(),
                               dataf['Target Text'].unique()):
        df_target.loc[index, 'Target Text'] = category
    df_target = df_target.sort_index()

    tgt_num = list(df_target.index)

    for num in tgt_num:
        tgt_lst.append(" ".join(
            i for i in dataf[column].loc[targ == num].explode()))
    for tgt in tgt_lst:
        wc_lst.append(WordCloud(background_color="white").generate(tgt))
    fig = plt.figure(figsize=(9, 9))
    fig.set_tight_layout(True)
    gs = fig.add_gridspec(4, 3)
    ax1 = fig.add_subplot(gs[0, 0])
    ax2 = fig.add_subplot(gs[0, 1])
    ax3 = fig.add_subplot(gs[0, 2])
    ax4 = fig.add_subplot(gs[1, 0])
    ax5 = fig.add_subplot(gs[1, 1])
    ax6 = fig.add_subplot(gs[1, 2])
    ax7 = fig.add_subplot(gs[2, 0])
    ax8 = fig.add_subplot(gs[2, 1])
    ax9 = fig.add_subplot(gs[2, 2])
    ax10 = fig.add_subplot(gs[3, 0])
    ax11 = fig.add_subplot(gs[3, 1])
    ax12 = fig.add_subplot(gs[3, 2])

    i = 0
    for ax in [ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8, ax9, ax10, ax11, ax12]:
        ax.axis("off")
        ax.imshow(wc_lst[i], interpolation='bilinear')
        ref = tgt_num[i]
        ax.set_title(f"Word Cloud \n {df_target.loc[ref,'Target Text']}")
        i += 1
    if title != '':
        fig.suptitle(title, fontsize=36)
    plt.show()
```

4.11 Splitting the data into Train and Test sets

In [57]:
```
# Creating the Test/Train split, 80% / 20%
xtrain, xtest, ytrain, ytest = train_test_split(df.iloc[:, 0:-1],
                                                df.iloc[:, -1],
                                                train_size=.8,
                                                random_state=42)
```

5 Vectorizing The Data

5.1 Build TF-IDF Vectorizer and Evaluate it with a Baseline Model, Multinomial Naive Bayes

In [58]:
```
# Instantiate a MultinomialNB classifier
baseline_model = MultinomialNB()
```

5.2 Build Bag-Of-Words Vectorizer

In [59]:
```
# To avoid meticulous editting, I will use this cell to define the text and
# tokens to be used in the vectorization and modeling below. Changing this cel
# will change the results below.
modstr = 'Stemmed Tweet'  # String to model
modtok = 'Stemmed Tokens' # Tokens to model
```

5.3 Build Word2Vec Vectorizer

In [60]:
```
vsize = 692
def word2vec_model(tokens):
    model = gensim.models.Word2Vec(
        tokens, # Panda series of lists with word tokens in it.
        vector_size = vsize,  # desired no. of features/independent variables
        window=5,  # context window size
        min_count=1,  # Ignores all words with total frequency lower than 2.
        sg=1,  # 1 for skip-gram model
        hs=0,
        negative=10,  # for negative sampling
        workers=32,  # no.of cores
        seed=34)
    model.train(tokens, total_examples=len(tokens), epochs=20)
    return model

model_w2v_df = word2vec_model(df[modtok])
```

In [61]:
```
def word_vector(model, tokens, size):
    vec = np.zeros(size).reshape((1, size))
    count = 0
    for word in tokens:
        try:
            vec += model.wv[word].reshape(1, size)
            count += 1
        except KeyError:  # handling the case where the token is not in
            continue      # vocabulary
    if count != 0:
        vec /= count
    return vec
```

In [62]:
```
def word2Vec(tokens, vector_size):
    wordvec_arrays = np.zeros((len(tokens), vector_size))
    array_index = 0
    for i in tokens.index:
        wordvec_arrays[array_index, :] = word_vector(model_w2v_df,
                                                     tokens[i], vector_size)
        array_index += 1
    wordvec_df = pd.DataFrame(wordvec_arrays)
    print('Word2Vec Shape:', wordvec_df.shape)

    # PredictorScaler=StandardScaler()
    PredictorScaler = MinMaxScaler()

    # Storing the fit object for later reference
    PredictorScalerFit = PredictorScaler.fit(wordvec_df)

    # Generating the standardized values of X
    wordvec_norm = PredictorScalerFit.transform(wordvec_df)

    return wordvec_df, wordvec_norm
```

In [63]:
```
print('X-Train Word2Vec:')
xtrain_w2v, xtrain_w2v_norm = word2Vec(xtrain[modtok],
                                       vsize)

print('X-Test Word2Vec:')
xtest_w2v, xtest_w2v_norm = word2Vec(xtest[modtok],
                                     vsize)
```

```
X-Train Word2Vec:
Word2Vec Shape: (5296, 692)
X-Test Word2Vec:
Word2Vec Shape: (1325, 692)
```

In [64]:
```
mnb_w2v_xtrain_kfold = cross_val_score(baseline_model,
                                       xtrain_w2v_norm, ytrain,
                                       scoring = 'f1_weighted')
```

6 Positive and Negative Word Associations

The Word2Vec vectorization method converts each token to it's own multidimensional vector. With that completed, it doesn't take much to compare the different vectors to see which are pointed in the same direction. Likewise, we can see which vectors are pointed in the opposite direction. That's all there is to word association with this method.

6.1 Cleaned Dataframe With All Original Tweets

As I referenced in the main Jupyter Notebook I wanted the ability to run the entire Jupyter Notebook's data cleaning process on the original raw data with most of the data cleaning intact, but not delete any data.

This dataframe is not in the main Jupyter Notebook because it is not used during modeling, but will be used during word association. Therefore, the only vectorization technique I need to use is Word2Vec. Then I'll be able to use it as a comparison during word association.

Here is the dataframe with all the original Tweets intact.

In [65]:
```
# The input for this parameter is the original df_raw.
def retain(dforg):

    # This function renames the columns.
    df_new = rename_columns(dforg)

    # This function renames the emotions.
    df_new = abbreviate_emotion(df_new)

    # Setting all letters to lowercase.
    df_new['Tweet'] = df_new['Tweet'].str.lower()

    # The single missing value tweet row still has to go. This drops that row.
    df_new.drop(index=6, inplace=True)
    df_new.reset_index(drop=True, inplace=True)

    # This is the larget dictionary found at the beginning that combines the
    # compound noun into firstword_secondword, and cleans up a few misspelled
    # words so the model doesn't get confused.
    df_new = string_replace(df_new, first_replace, replacements,
                            to_print = False)


    # Tokenizes all of the tweets.
    df_new = tokenize_columns(df_new)

    # Creates a column for each, already defined, keyword and creates a list
    # of all rows without a keyword.
    df_new, no_kw_new = keyword_labels(df_new, keywords_apple, keywords_google

    # This cell labels each row as "Apple", "Google", "Both" or "Unknown"
    df_new = company_organization(df_new, keywords_apple, keywords_google)

    # This determines the company each tweet addresses based on the keywords,
    # and adds columns for easily finding tweets from a specific company in
    # later cells.
    df_new = company_by_product(df_new)

    # Checking which rows don't have matching values in columns "Company" and
    # "Company by Product"
    df_new = no_match(df_new)

    # This defines how to label the few tweets that are laveled as one company
    # but keywords label these tweet as the other company. This function
    # properly labels those tweets.
    df_new = mismatch(df_new)

    # This defines which tweets have no keywords, but do have a product
    # category assigned as part of the original data.
    inspect, no_kw_new = no_keywords(df_new, no_kw_new)


    # # Printing the tweets that don't have any of the keywords, but do have a
    # given label in the Product column. Also, saving their index numbers in a
    # list titled apple_tweets
    apple_tweets_new = tweets_to_edit(no_kw_new, inspect, False)
```

```
    # This function hand labels a few tweets that have no keywords, but are
    # clearly tweets about Apple apps.
    df_new, no_kw_new = apple_tweet_edit(df_new,
                                         no_kw_new, apple_tweets_new)

    # The lists and dictionaries below are hand made after reading through all
    # the tweets with unknown companies. This function properly labels them.
    df_new, no_kw_new = corrections(df_new, no_kw_new)

    # Create a column for both Apple Keywords and Google Keywords.
    df_new = categorize(df_new, keywords_apple, keywords_google)

    # This labels each tweet as either "Apple Brand", "Apple Products",
    #  "Google Brand" or "Google Products".
    df_new = keyword_combos(df_new, keywords_apple, keywords_google)

    # This command removes several columns that are unnecessary and resets
    # the index.
    df_new = df_new.loc[:, ['Tweet', 'Emotion', 'Tokens',
                            'Tokens With Hashtags', 'Hashtags', 'Keywords',
                            'Company', 'Category']].copy()
    df_new.reset_index(drop=True, inplace=True)

    # This function sets the targets.
    df_new = targets(df_new)

    # This function trims out all the stopwords.
    df_new = trim_stopwords(df_new)

    # This function stems the words.
    df_new = stem_tokens(df_new)

    # This cell uses Word2Vec to vectorize the whole DataFrame.
    model_w2v_df_new = word2vec_model(df_new[modtok])

    return df_new, model_w2v_df_new

df_retain, model_w2v_df_retain = retain(df_raw)
```

In [66]:
```
df_retain[['Tweet', 'Stemmed Tokens', 'Hashtags',
           'Target Text', 'Target']].head()
```

Out[66]:

	Tweet	Stemmed Tokens	Hashtags	Target Text	Target
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.	[wesley83, 3g, iphone, hrs, tweet, rise_austin, dead, need, upgrad, plugin, station, sxsw]	[#rise_austin, #sxsw]	Negative - Apple Products	3
1	@jessedee know about @fludapp ? awesome ipad/iphone_app that you will likely appreciate for its design. also, they are giving free ts at #sxsw	[jessede, know, fludapp, awesom, ipad, iphone_app, like, appreci, design, also, give, free, ts, sxsw]	[#sxsw]	Positive - Apple Products	5
2	@swonderlin can not wait for #ipad2 also. they should sale them down at #sxsw.	[swonderlin, wait, ipad2, also, sale, sxsw]	[#ipad2, #sxsw]	Positive - Apple Products	5
3	@sxsw i hope this year's festival is not as crashy as this year's iphone_app. #sxsw	[sxsw, hope, year, festiv, crashi, year, iphone_app, sxsw]	[#sxsw]	Negative - Apple Products	3
4	@sxtxstate great stuff on fri #sxsw: marissa_mayer (google), tim_o_reilly (tech books/conferences) & matt_mullenweg (wordpress)	[sxtxstate, great, stuff, fri, sxsw, marissa_mayer, google, tim_o_reilli, tech, book, confer, matt_mullenweg, wordpress]	[#sxsw]	Positive - Google Brand	8

The slicing and interpretation of the many ways you can run the association methods can be tedious. So I set out to build a tool that could easily specify which target the method was pulling the word associations with, and easily switch the word. The construction of this tool began with creating multiple Word2Vec models and dataframes containing its vectors that can be easily referenced.

6.2 Creating Targeted Models and DataFrames

Below is a behemoth of a DataFrame. It slices up the dataset into subsets representing each of the twelve categories, both the DataFrames and Word2Vec models. The titles of each model represent shorthand to identify what it i. Each of the letters between the underscores is separated as such:

df/model = DataFrame or Word2Vec model

w2v = Word2Vec

neg/neu/pos = Negative, Neutral or Positive

ab/ap/gb/gp/a/g = Apple Brand, Apple Products, Google Brand, Google Products, Apple, Google

The different Word2Vec models and DataFrames are used for quick access to create visualizations.

In [67]:
```
# I'm curious if I'll get dramatically different answers if I run the Word2Vec
# Vectorization process separtley on DataFrames created exclusively for each
# target. Only one way to find out.

titles = {}

df_w2v_neg_ab = df.loc[df['Target Text'] == 'Negative - Apple Brand']
model_w2v_neg_ab = word2vec_model(df_w2v_neg_ab[modtok])
df_w2v_neg_ab.name = 'df_w2v_neg_ab'
model_w2v_neg_ab.name = 'model_w2v_neg_ab'
titles[model_w2v_neg_ab.name] = 'Negative - Apple Brand'
titles[df_w2v_neg_ab.name] = 'Negative - Apple Brand'

df_w2v_neu_ab = df.loc[df['Target Text'] == 'Neutral - Apple Brand']
model_w2v_neu_ab = word2vec_model(df_w2v_neu_ab[modtok])
df_w2v_neu_ab.name = 'df_w2v_neu_ab'
model_w2v_neu_ab.name = 'model_w2v_neu_ab'
titles[model_w2v_neu_ab.name] = 'Neutral - Apple Brand'
titles[df_w2v_neu_ab.name] = 'Neutral - Apple Brand'

df_w2v_pos_ab = df.loc[df['Target Text'] == 'Positive - Apple Brand']
model_w2v_pos_ab = word2vec_model(df_w2v_pos_ab[modtok])
df_w2v_pos_ab.name = 'df_w2v_pos_ab'
model_w2v_pos_ab.name = 'model_w2v_pos_ab'
titles[model_w2v_pos_ab.name] = 'Positive - Apple Brand'
titles[df_w2v_pos_ab.name] = 'Positive - Apple Brand'

df_w2v_neg_ap = df.loc[df['Target Text'] == 'Negative - Apple Products']
model_w2v_neg_ap = word2vec_model(df_w2v_neg_ap[modtok])
df_w2v_neg_ap.name = 'df_w2v_neg_ap'
model_w2v_neg_ap.name = 'model_w2v_neg_ap'
titles[model_w2v_neg_ap.name] = 'Negative - Apple Products'
titles[df_w2v_neg_ap.name] = 'Negative - Apple Products'

df_w2v_neu_ap = df.loc[df['Target Text'] == 'Neutral - Apple Products']
model_w2v_neu_ap = word2vec_model(df_w2v_neu_ap[modtok])
df_w2v_neu_ap.name = 'df_w2v_neu_ap'
model_w2v_neu_ap.name = 'model_w2v_neu_ap'
titles[model_w2v_neu_ap.name] = 'Neutral - Apple Products'
titles[df_w2v_neu_ap.name] = 'Neutral - Apple Products'

df_w2v_pos_ap = df.loc[df['Target Text'] == 'Positive - Apple Products']
model_w2v_pos_ap = word2vec_model(df_w2v_pos_ap[modtok])
df_w2v_pos_ap.name = 'df_w2v_pos_ap'
model_w2v_pos_ap.name = 'model_w2v_pos_ap'
titles[model_w2v_pos_ap.name] = 'Positive - Apple Products'
titles[df_w2v_pos_ap.name] = 'Positive - Apple Products'

df_w2v_neg_gb = df.loc[df['Target Text'] == 'Negative - Google Brand']
model_w2v_neg_gb = word2vec_model(df_w2v_neg_gb[modtok])
df_w2v_neg_gb.name = 'df_w2v_neg_gb'
model_w2v_neg_gb.name = 'model_w2v_neg_gb'
titles[model_w2v_neg_gb.name] = 'Negative - Google Brand'
titles[df_w2v_neg_gb.name] = 'Negative - Google Brand'

df_w2v_neu_gb = df.loc[df['Target Text'] == 'Neutral - Google Brand']
model_w2v_neu_gb = word2vec_model(df_w2v_neu_gb[modtok])
```

```
df_w2v_neu_gb.name = 'df_w2v_neu_gb'
model_w2v_neu_gb.name = 'model_w2v_neu_gb'
titles[model_w2v_neu_gb.name] = 'Neutral - Google Brand'
titles[df_w2v_neu_gb.name] = 'Neutral - Google Brand'

df_w2v_pos_gb = df.loc[df['Target Text'] == 'Positive - Google Brand']
model_w2v_pos_gb = word2vec_model(df_w2v_pos_gb[modtok])
df_w2v_pos_gb.name = 'df_w2v_pos_gb'
model_w2v_pos_gb.name = 'model_w2v_pos_gb'
titles[model_w2v_pos_gb.name] = 'Positive - Google Brand'
titles[df_w2v_pos_gb.name] = 'Positive - Google Brand'

df_w2v_neg_gp = df.loc[df['Target Text'] == 'Negative - Google Products']
model_w2v_neg_gp = word2vec_model(df_w2v_neg_gp[modtok])
df_w2v_neg_gp.name = 'df_w2v_neg_gp'
model_w2v_neg_gp.name = 'model_w2v_neg_gp'
titles[model_w2v_neg_gp.name] = 'Negative - Google Products'
titles[df_w2v_neg_gp.name] = 'Negative - Google Products'

df_w2v_neu_gp = df.loc[df['Target Text'] == 'Neutral - Google Products']
model_w2v_neu_gp = word2vec_model(df_w2v_neu_gp[modtok])
df_w2v_neu_gp.name = 'df_w2v_neu_gp'
model_w2v_neu_gp.name = 'model_w2v_neu_gp'
titles[model_w2v_neu_gp.name] = 'Neutral - Google Products'
titles[df_w2v_neu_gp.name] = 'Neutral - Google Products'

df_w2v_pos_gp = df.loc[df['Target Text'] == 'Positive - Google Products']
model_w2v_pos_gp = word2vec_model(df_w2v_pos_gp[modtok])
df_w2v_pos_gp.name = 'df_w2v_pos_gp'
model_w2v_pos_gp.name = 'model_w2v_pos_gp'
titles[model_w2v_pos_gp.name] = 'Positive - Google Products'
titles[df_w2v_pos_gp.name] = 'Positive - Google Products'

df_w2v_neg_a = df.loc[(df['Company'] == 'Apple') &
                      (df['Emotion'] == 'Negative')]
model_w2v_neg_a = word2vec_model(df_w2v_neg_a[modtok])
df_w2v_neg_a.name = 'df_w2v_neg_a'
model_w2v_neg_a.name = 'model_w2v_neg_a'
titles[model_w2v_neg_a.name] = 'Negative - Apple (Products & Brand)'
titles[df_w2v_neg_a.name] = 'Negative - Apple (Products & Brand)'

df_w2v_neu_a = df.loc[(df['Company'] == 'Apple') &
                      (df['Emotion'] == 'Neutral')]
model_w2v_neu_a = word2vec_model(df_w2v_neu_a[modtok])
df_w2v_neu_a.name = 'df_w2v_neu_a'
model_w2v_neu_a.name = 'model_w2v_neu_a'
titles[model_w2v_neu_a.name] = 'Neutral - Apple (Products & Brand)'
titles[df_w2v_neu_a.name] = 'Neutral - Apple (Products & Brand)'

df_w2v_pos_a = df.loc[(df['Company'] == 'Apple') &
                      (df['Emotion'] == 'Positive')]
model_w2v_pos_a = word2vec_model(df_w2v_pos_a[modtok])
df_w2v_pos_a.name = 'df_w2v_pos_a'
model_w2v_pos_a.name = 'model_w2v_pos_a'
titles[model_w2v_pos_a.name] = 'Positive - Apple (Products & Brand)'
titles[df_w2v_pos_a.name] = 'Positive - Apple (Products & Brand)'
```

```
df_w2v_neg_g = df.loc[(df['Company'] == 'Google') &
                      (df['Emotion'] == 'Negative')]
model_w2v_neg_g = word2vec_model(df_w2v_neg_g[modtok])
df_w2v_neg_g.name = 'df_w2v_neg_g'
model_w2v_neg_g.name = 'model_w2v_neg_g'
titles[model_w2v_neg_g.name] = 'Negative - Google (Products & Brand)'
titles[df_w2v_neg_g.name] = 'Negative - Google (Products & Brand)'

df_w2v_neu_g = df.loc[(df['Company'] == 'Google') &
                      (df['Emotion'] == 'Neutral')]
model_w2v_neu_g = word2vec_model(df_w2v_neu_g[modtok])
df_w2v_neu_g.name = 'df_w2v_neu_g'
model_w2v_neu_g.name = 'model_w2v_neu_g'
titles[model_w2v_neu_g.name] = 'Neutral - Google (Products & Brand)'
titles[df_w2v_neu_g.name] = 'Neutral - Google (Products & Brand)'

df_w2v_pos_g = df.loc[(df['Company'] == 'Google') &
                      (df['Emotion'] == 'Positive')]
model_w2v_pos_g = word2vec_model(df_w2v_pos_g[modtok])
df_w2v_pos_g.name = 'df_w2v_pos_g'
model_w2v_pos_g.name = 'model_w2v_pos_g'
titles[model_w2v_pos_g.name] = 'Positive - Google (Products & Brand)'
titles[df_w2v_pos_g.name] = 'Positive - Google (Products & Brand)'
```

This next cell puts the models into lists that allow easy access for the Word2Vec functions below.

In [68]:
```
apple_brand_wv =     [model_w2v_neg_ab,
                      model_w2v_neu_ab,
                      model_w2v_pos_ab]
apple_products_wv =  [model_w2v_neg_ap,
                      model_w2v_neu_ap,
                      model_w2v_pos_ap]
google_brand_wv =    [model_w2v_neg_gb,
                      model_w2v_neu_gb,
                      model_w2v_pos_gb]
google_products_wv = [model_w2v_neg_gp,
                      model_w2v_neu_gp,
                      model_w2v_pos_gp]
apple_both_wv =       [model_w2v_neg_a,
                      model_w2v_neu_a,
                      model_w2v_pos_a]
google_both_wv =      [model_w2v_neg_g,
                      model_w2v_neu_g,
                      model_w2v_pos_g]

apple_neg_wv =       [model_w2v_neg_ab,
                      model_w2v_neg_ap,
                      model_w2v_neg_a]
apple_neu_wv =       [model_w2v_neu_ab,
                      model_w2v_neu_ap,
                      model_w2v_neu_a]
apple_pos_wv =       [model_w2v_pos_ab,
                      model_w2v_pos_ap,
                      model_w2v_pos_a]
google_neg_wv =      [model_w2v_neg_gb,
                      model_w2v_neg_gp,
                      model_w2v_neg_g]
google_neu_wv =      [model_w2v_neu_gb,
                      model_w2v_neu_gp,
                      model_w2v_neu_g]
google_pos_wv =      [model_w2v_pos_gb,
                      model_w2v_pos_gp,
                      model_w2v_pos_g]


apple_all_wv = []
apple_all_wv.extend(apple_brand_wv)
apple_all_wv.extend(apple_products_wv)
apple_all_wv.extend(apple_both_wv)

google_all_wv = []
google_all_wv.extend(google_brand_wv)
google_all_wv.extend(google_products_wv)
google_all_wv.extend(google_both_wv)
all_wv = []
all_wv.extend(apple_all_wv)
all_wv.extend(google_all_wv)
all_wv.append(model_w2v_df_retain)
model_w2v_df_retain.name = 'model_w2v_df_retain'
df_retain.name = 'df_retain'
titles[model_w2v_df_retain.name] = 'Entire Dataset'
```

```
titles[df_retain.name] = 'Entire Dataset'
```

Same for the DataFrames.

In [69]:
```
apple_brand_df =     [df_w2v_neg_ab,
                      df_w2v_neu_ab,
                      df_w2v_pos_ab]
apple_products_df =  [df_w2v_neg_ap,
                      df_w2v_neu_ap,
                      df_w2v_pos_ap]
google_brand_df =    [df_w2v_neg_gb,
                      df_w2v_neu_gb,
                      df_w2v_pos_gb]
google_products_df = [df_w2v_neg_gp,
                      df_w2v_neu_gp,
                      df_w2v_pos_gp]
apple_both_df =       [df_w2v_neg_a,
                      df_w2v_neu_a,
                      df_w2v_pos_a]
google_both_df =      [df_w2v_neg_g,
                      df_w2v_neu_g,
                      df_w2v_pos_g]

apple_neg_df =       [df_w2v_neg_ab,
                      df_w2v_neg_ap,
                      df_w2v_neg_a]
apple_neu_df =       [df_w2v_neu_ab,
                      df_w2v_neu_ap,
                      df_w2v_neu_a]
apple_pos_df =       [df_w2v_pos_ab,
                      df_w2v_pos_ap,
                      df_w2v_pos_a]
google_neg_df =      [df_w2v_neg_gb,
                      df_w2v_neg_gp,
                      df_w2v_neg_g]
google_neu_df =      [df_w2v_neu_gb,
                      df_w2v_neu_gp,
                      df_w2v_neu_g]
google_pos_df =      [df_w2v_pos_gb,
                      df_w2v_pos_gp,
                      df_w2v_pos_g]


apple_all_df = []
apple_all_df.extend(apple_brand_df)
apple_all_df.extend(apple_products_df)
apple_all_df.extend(apple_both_df)

google_all_df = []
google_all_df.extend(google_brand_df)
google_all_df.extend(google_products_df)
google_all_df.extend(google_both_df)
all_df = []
all_df.extend(apple_all_df)
all_df.extend(google_all_df)
all_df.append(df_retain)

google_df = [df_w2v_neg_g, df_w2v_neu_g, df_w2v_pos_g]
```

```
apple_df  = [df_w2v_neg_a, df_w2v_neu_a, df_w2v_pos_a]
```

6.3 Word Association Functions

These next three functions create word clouds for a given word and slice of the dataset. The word clouds also list the word association correlation values.

In [70]:
```
# Checks to see if the word is in the give model, and returns the pair of
# association tuple lists.

def word_association(mdl, wrd):
    if wrd not in mdl.wv.key_to_index:
        string = 'The word "' + wrd.capitalize() + '" is not in the model '
        string += titles[mdl.name]
        return  string, ''
    pos = mdl.wv.most_similar(positive=wrd)
    neg = mdl.wv.most_similar(negative=wrd)
    return pos, neg
```

In [71]:
```
# Creates a pair of clouds that that show the top ten positively and negativel
# associated word in the given association list of association tuples, the wor
# and the title of the graph.

def word_assoc_cloud(pos_assoc, neg_assoc, word, title = '', subs = []):
    word_cap = word.capitalize()
    pos_lst = []
    neg_lst = []
    for i in range(len(pos_assoc)):
        pos_lst.append(pos_assoc[i][0])
        neg_lst.append(neg_assoc[i][0])
    pos_text = " ".join(i for i in pos_lst)
    neg_text = " ".join(i for i in neg_lst)
    wc_lst = []

    wc_lst.append(WordCloud(background_color="white").generate(pos_text))
    wc_lst.append(WordCloud(background_color="white").generate(neg_text))

    fig = plt.figure(figsize=(9, 4))
    fig.set_tight_layout(True)
    gs = fig.add_gridspec(1, 2)
    ax1 = fig.add_subplot(gs[0, 0])
    ax2 = fig.add_subplot(gs[0, 1])

    if subs == []:
        subtitles = ['Associated With '     + word_cap,
                     'Not Associated With ' + word_cap]
    else:
        subtitles = subs
    i = 0
    for ax in [ax1, ax2]:
        ax.spines['bottom'].set_color('0.5')
        ax.spines['top'].set_color('0.5')
        ax.spines['right'].set_color('0.5')
        ax.spines['left'].set_color('0.5')
        ax.patch.set_facecolor('0.1')
        ax.tick_params(axis='x', which='both', bottom = False,
                       labelbottom = False)
        ax.tick_params(axis='y', which='both', left = False,
                       labelleft = False)
        ax.yaxis.label.set_color('0.9')
        ax.xaxis.label.set_color('0.9')
        ax.margins(0.5)
        ax.imshow(wc_lst[i], interpolation='bilinear')
        ax.set_title(subtitles[i], fontsize=18)
        i += 1
    if title != '':
        fig.suptitle(title, fontsize=24)
    plt.show()
```

In [72]:
```
# Creates a word cloud graphic and lists underneath the words and their
# association similarity for the given word and model.

def word_association_visual(model_lst, word):
    if type(model_lst) is not list:
        temp = []
        temp.append(model_lst)
        model_lst = temp
    for model in model_lst:
        pos_tup_lst, neg_tup_lst = word_association(model, word)
        if type(pos_tup_lst) is str:
            print(pos_tup_lst)
            continue

        first = 'Word Clouds for\n'
        mid = titles[model.name] + ' Tweets\n'
        last  =  'Associated With The Word "{}"'.format(word.capitalize())
        model_title = first + mid + last
        word_assoc_cloud(pos_tup_lst, neg_tup_lst, word,
            title = model_title)

        max_len = 0
        for i in range(len(pos_tup_lst)):
            if len(pos_tup_lst[i][0]) > max_len:
                max_len = len(pos_tup_lst[i][0])
            if len(neg_tup_lst[i][0]) > max_len:
                max_len = len(neg_tup_lst[i][0])
        for i in range(len(pos_tup_lst)):
            pos_txt = pos_tup_lst[i][0]
            neg_txt = neg_tup_lst[i][0]
            spaces_pos = ''
            spaces_neg = ''
            between_space = ''
            pos_num = str('%.3f' % round(pos_tup_lst[i][1],3))
            neg_num = str('%.3f' % round(neg_tup_lst[i][1],3))
            num_spaces_pos = max_len - len(pos_txt) + 4
            num_spaces_neg = max_len - len(neg_txt) + 4
            if float(pos_num) < 0:
                num_spaces_pos -= 1
            if float(neg_num) < 0:
                num_spaces_neg -= 1
            for i in range(num_spaces_pos):
                spaces_pos += ' '
            for i in range(num_spaces_neg):
                spaces_neg += ' '
            txt =  '(' + pos_txt + spaces_pos + pos_num + ')'
            between = 40 - len(txt)
            for i in range(between):
                between_space += ' '
            txt += between_space + '(' + neg_txt + spaces_neg + neg_num + ')'
            print(txt)
        print()
    return
```

The function below lists the top ten hashtags used for the given model slice and word. Similar to the function above, it gives a word cloud of the hashtags and a tally below that shows how many times that hastag was used in the given model.

In [73]:
```
def hashtag_search(model_lst, word):
    if type(model_lst) is not list:
        temp = []
        temp.append(model_lst)
        model_lst = temp
    for dfone in model_lst:
        hashtagbag = []
        num_tweets = 0
        for i in dfone.index:
            if (word in dfone.loc[i, 'Tokens']):
                num_tweets += 1
                for hashtag in dfone.loc[i, 'Hashtags']:
                    hashtagbag.append(hashtag)
        count = tally(hashtagbag)
        t = ('Word Cloud For Included Hashtags In\n' +
             titles[dfone.name] + ' Tweets\n' +
             'That Include The Word "' + word.capitalize() + '"\n')
        tot = str(len(dfone))
        if len(hashtagbag) !=0:
            wc(hashtagbag, title = t, re = r"#(?:\w[\w']+)")
            print('For the', titles[dfone.name], 'set of', tot,'tweets:')
            print('The word "' + word.capitalize() +'" is included in',
                  str(num_tweets) + '/' + tot, 'tweets.')
            print()
            print('Top ten Hashtags included with the word "' +
                  word.capitalize() +'" in this dataset:')
            for pair in count[:10]:
                print(pair)
        else:
            string = 'The word "' + word.capitalize() + '" is not in the '
            string += 'DataFrame ' + titles[dfone.name]
            print(string)

    return count
```

6.4 Word Association Tool – Workspace

Here is the fruit of my labor. With only two terms, I can now provide word clouds and a list of ten positive association and negative association words. I can do this for multiple slices of the dataset at once, and I can do it for all the data in the set.

I performed multiple searches through multiple words and it's clear the search could continue forever. To limit the scope to what is appropriate for the time I'm given, I performed analysis on a single keyword for four words: The #1 keyword in each of the following categories:

1. Google Brand : Google
2. Google Products : Android
3. Apple Brand : Apple

4. Apple Products : iPad

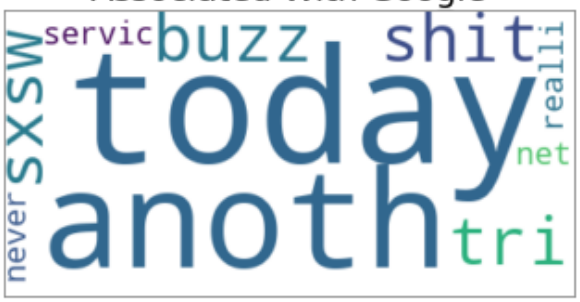
Below I list the word associations for each word for all the available slices of the DataFrame they exist in.

In [74]:
```
word_association_visual(all_wv, 'google');
```

```
The word "Google" is not in the model Negative - Apple Brand
The word "Google" is not in the model Neutral - Apple Brand
The word "Google" is not in the model Positive - Apple Brand
The word "Google" is not in the model Negative - Apple Products
The word "Google" is not in the model Neutral - Apple Products
The word "Google" is not in the model Positive - Apple Products
The word "Google" is not in the model Negative - Apple (Products & Brand)
The word "Google" is not in the model Neutral - Apple (Products & Brand)
The word "Google" is not in the model Positive - Apple (Products & Brand)
```

Word Clouds for Negative - Google Brand Tweets Associated With The Word "Google"

Associated With Google



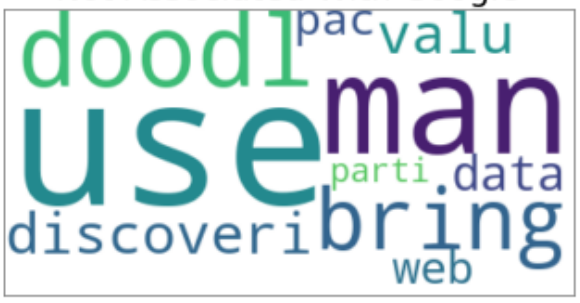
Not Associated With Google

(today	1.000)	(rais	-0.999)
(anoth	1.000)	(hand	-0.999)
(buzz	1.000)	(room	-0.999)
(shit	1.000)	(solut	-0.999)
(sxsw	1.000)	(pack	-0.999)
(tri	1.000)	(care	-0.999)
(servic	1.000)	(busi	-0.999)
(net	1.000)	(nobodi	-0.999)
(realli	1.000)	(socialview	-0.999)
(never	1.000)	(much	-0.999)

Word Clouds for Neutral - Google Brand Tweets Associated With The Word "Google"

Associated With Google

Not Associated With Google



(sxsw	0.957)
(sxswi	0.729)
(south	0.694)
(pi	0.693)
(via	0.689)
(southwest	0.689)
(kek	0.687)
(block	0.683)
(despit	0.680)
(paypal	0.678)

(use	-0.161)
(man	-0.222)
(doodl	-0.267)
(bring	-0.273)
(discoveri	-0.275)
(valu	-0.278)
(data	-0.281)
(pac	-0.283)
(web	-0.291)
(parti	-0.291)

Word Clouds for Positive - Google Brand Tweets Associated With The Word "Google"

Associated With Google

Not Associated With Google

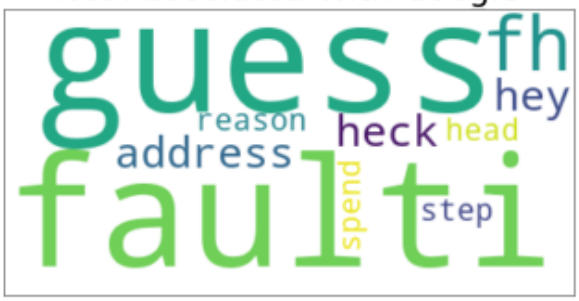
(peep	0.991)
(happen	0.990)
(close	0.990)
(expect	0.990)
(lucki	0.989)
(folk	0.989)
(debut	0.989)
(tip	0.989)
(appear	0.989)
(word	0.989)

(save	-0.498)
(peopl	-0.543)
(traffic	-0.547)
(rout	-0.565)
(work	-0.576)
(brain	-0.583)
(world	-0.587)
(around	-0.604)
(digit	-0.608)
(physic	-0.608)

Word Clouds for Negative - Google Products Tweets Associated With The Word "Google"

Associated With Google

Not Associated With Google



```
(sxsw       1.000)
(android    1.000)
(app        1.000)
(maps       1.000)
(imag       1.000)
(nativ      1.000)
(peopl      0.999)
(seem       0.999)
(realli     0.999)
(horribl    0.999)
```

```
(guess     -0.987)
(faulti    -0.988)
(fh        -0.997)
(address   -0.997)
(heck      -0.997)
(hey       -0.998)
(reason    -0.998)
(spend     -0.998)
(step      -0.998)
(head      -0.998)
```

Word Clouds for Neutral - Google Products Tweets Associated With The Word "Google"

Associated With Google

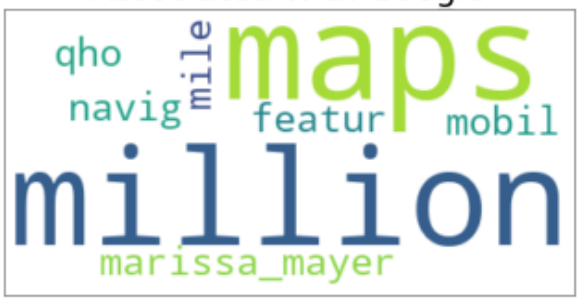
Not Associated With Google


```
(usag            0.999)
(mobil           0.999)
(150             0.999)
(user            0.999)
(maps            0.998)
(million         0.997)
(40              0.997)
(marissa_mayer   0.995)
(use             0.989)
(stat            0.988)
```

```
(android        -0.880)
(hilton         -0.890)
(meetup         -0.892)
(30pm           -0.894)
(dev            -0.895)
(team           -0.899)
(galaxi         -0.903)
(launch         -0.904)
(615ab          -0.905)
(pearl          -0.906)
```

Word Clouds for Positive - Google Products Tweets Associated With The Word "Google"

Associated With Google

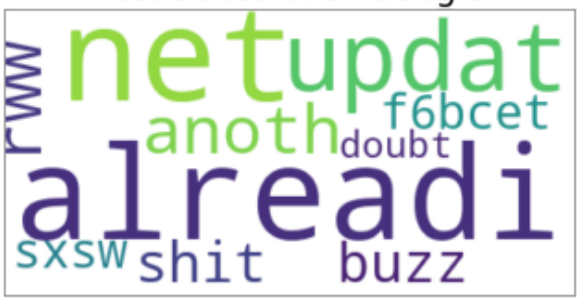


Not Associated With Google

(maps	0.999)	(choic	-0.792)
(million	0.997)	(win	-0.837)
(150	0.997)	(team	-0.843)
(marissa_mayer	0.996)	(best	-0.849)
(featur	0.996)	(award	-0.850)
(40	0.996)	(yes	-0.853)
(mobil	0.996)	(thank	-0.858)
(navig	0.995)	(gowalla	-0.886)
(mile	0.995)	(android_app	-0.898)
(qho	0.994)	(android	-0.940)

Word Clouds for Negative - Google (Products & Brand) Tweets Associated With The Word "Google"

Associated With Google



Not Associated With Google

(net	1.000)	(network	-0.998)
(alreadi	1.000)	(solut	-0.998)
(updat	1.000)	(hand	-0.998)
(anoth	1.000)	(room	-0.998)
(rww	1.000)	(rais	-0.998)
(shit	1.000)	(social	-0.998)
(buzz	1.000)	(call	-0.999)
(f6bcet	1.000)	(new	-0.999)
(sxsw	1.000)	(great	-0.999)
(doubt	1.000)	(pack	-0.999)

Word Clouds for Neutral - Google (Products & Brand) Tweets Associated With The Word "Google"

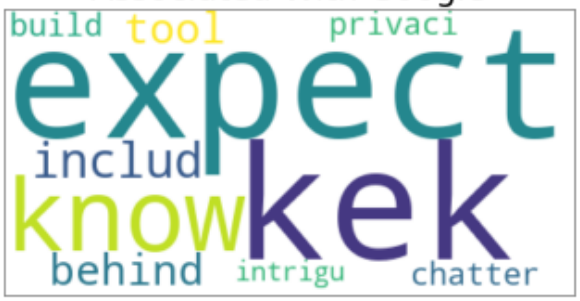
Associated With Google

Not Associated With Google

(sxsw	0.879)	(team	-0.053)
(via	0.796)	(develop	-0.070)
(sxswi	0.738)	(android	-0.088)
(take	0.722)	(meetup	-0.114)
(fb	0.712)	(leagu	-0.123)
(surpris	0.712)	(extraordinari	-0.126)
(focus	0.711)	(dev	-0.134)
(interact	0.685)	(chrome	-0.135)
(appar	0.684)	(know	-0.144)
(integr	0.683)	(hacker	-0.160)

Word Clouds for Positive - Google (Products & Brand) Tweets Associated With The Word "Google"

Associated With Google



Not Associated With Google

(expect	0.963)	(win	-0.331)
(kek	0.962)	(choic	-0.349)
(know	0.961)	(award	-0.350)
(includ	0.960)	(team	-0.359)
(behind	0.957)	(gowalla	-0.366)
(tool	0.957)	(yes	-0.370)
(privaci	0.955)	(best	-0.370)
(build	0.954)	(android_app	-0.391)
(chatter	0.954)	(thank	-0.403)
(intrigu	0.954)	(brain	-0.426)

Word Clouds for Entire Dataset Tweets Associated With The Word "Google"

Associated With Google

Not Associated With Google

```
(kek                  0.556)          (chevi                 0.018)
(incorrect            0.554)          (16gb                 -0.008)
(tim_ferri            0.542)          (pure                 -0.021)
(increas              0.539)          (ly                   -0.028)
(euphoria             0.528)          (mp3                  -0.028)
(njdoc                0.522)          (64gb                 -0.030)
(stevecas             0.520)          (black                -0.030)
(nadja                0.519)          (extra                -0.033)
(paul_adam            0.518)          (sunni                -0.035)
(justinjustinjustin   0.516)          (autocorrect          -0.045)
```

In [75]:
```
word_association_visual(all_wv, 'android');
```

```
The word "Android" is not in the model Negative - Apple Brand
The word "Android" is not in the model Neutral - Apple Brand
The word "Android" is not in the model Positive - Apple Brand
The word "Android" is not in the model Negative - Apple Products
The word "Android" is not in the model Neutral - Apple Products
The word "Android" is not in the model Positive - Apple Products
The word "Android" is not in the model Negative - Apple (Products & Brand)
The word "Android" is not in the model Neutral - Apple (Products & Brand)
The word "Android" is not in the model Positive - Apple (Products & Brand)
The word "Android" is not in the model Negative - Google Brand
The word "Android" is not in the model Neutral - Google Brand
The word "Android" is not in the model Positive - Google Brand
```

(google	1.000)
(sxsw	1.000)
(imag	0.999)
(app	0.999)
(maps	0.999)
(nativ	0.999)
(realli	0.999)
(crash	0.999)
(well	0.999)
(meet	0.999)

(guess	-0.987)
(faulti	-0.988)
(address	-0.996)
(fh	-0.997)
(heck	-0.997)
(hey	-0.998)
(step	-0.998)
(head	-0.998)
(spend	-0.998)
(reason	-0.998)

Word Clouds for Neutral - Google Products Tweets Associated With The Word "Android"

Associated With Android

Not Associated With Android

(detail	0.997)	(40	-0.846)
(need	0.996)	(maps	-0.854)
(team	0.996)	(usag	-0.876)
(austin	0.996)	(150	-0.878)
(get	0.996)	(google	-0.880)
(know	0.996)	(mobil	-0.886)
(let	0.996)	(user	-0.889)
(check	0.996)	(million	-0.906)
(free	0.995)	(marissa_mayer	-0.920)
(dev	0.995)	(use	-0.939)

Word Clouds for Positive - Google Products Tweets Associated With The Word "Android"

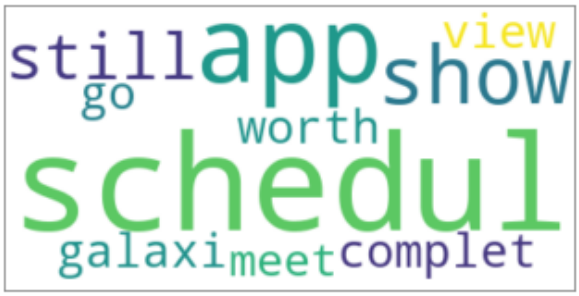
Associated With Android

Not Associated With Android

(congrat	0.997)	(save	-0.881)
(job	0.996)	(year	-0.900)
(congratul	0.995)	(traffic	-0.901)
(nice	0.995)	(everi	-0.903)
(huzzah	0.994)	(rout	-0.916)
(check	0.993)	(user	-0.916)
(look	0.993)	(around	-0.919)
(awesom	0.993)	(day	-0.922)
(far	0.993)	(maps	-0.933)
(androidsxsw	0.992)	(choic	-0.939)

Word Clouds for Negative - Google (Products & Brand) Tweets Associated With The Word "Android"

Associated With Android



Not Associated With Android

(schedul	1.000)
(app	1.000)
(show	1.000)
(still	1.000)
(complet	1.000)
(galaxi	1.000)
(go	1.000)
(worth	1.000)
(view	1.000)
(meet	1.000)

(network	-0.996)
(social	-0.997)
(call	-0.997)
(new	-0.998)
(major	-0.998)
(launch	-0.998)
(possibl	-0.998)
(solut	-0.998)
(room	-0.998)
(hand	-0.998)

Word Clouds for Neutral - Google (Products & Brand) Tweets Associated With The Word "Android"

Associated With Android

Not Associated With Android

(fcc	0.885)
(lightbox	0.882)
(team	0.874)
(chrome	0.873)
(develop	0.871)
(award	0.867)
(samsung	0.867)
(startup	0.864)
(finalist	0.863)
(ii	0.859)

(google	-0.088)
(site	-0.124)
(hotpot	-0.125)
(search	-0.150)
(product	-0.157)
(els	-0.172)
(local	-0.181)
(plenti	-0.185)
(interest	-0.189)
(ht	-0.197)

Word Clouds for Positive - Google (Products & Brand) Tweets Associated With The Word "Android"

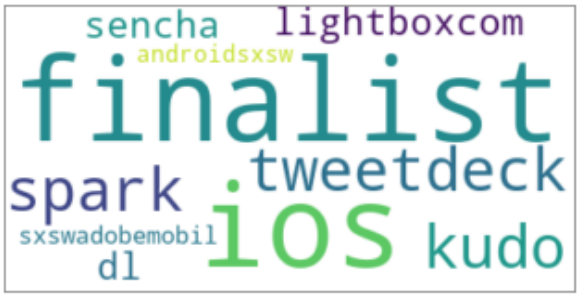
Associated With Android

Not Associated With Android

(date	0.969)	(40	-0.231)
(thank	0.966)	(mobil	-0.239)
(dev	0.963)	(maps	-0.245)
(pearl	0.957)	(around	-0.265)
(app	0.956)	(150	-0.266)
(androidsxsw	0.955)	(million	-0.269)
(lustr	0.953)	(world	-0.282)
(award	0.951)	(view	-0.282)
(bizzi	0.949)	(connect	-0.286)
(team	0.949)	(user	-0.292)

Word Clouds for Entire Dataset Tweets Associated With The Word "Android"

Associated With Android



Not Associated With Android

(ios	0.578)	(instrument	0.034)
(finalist	0.562)	(fedora	0.015)
(tweetdeck	0.557)	(avoid	0.010)
(kudo	0.548)	(virgin	0.008)
(spark	0.547)	(idea	0.006)
(lightboxcom	0.529)	(front	0.003)
(sencha	0.524)	(high	-0.008)
(dl	0.520)	(likeabl	-0.012)
(sxswadobemobil	0.518)	(may	-0.014)
(androidsxsw	0.513)	(trustworthi	-0.018)

In [76]:
```
word_association_visual(all_wv, 'apple');
```

(one	1.000)	(ridic	-0.999)
(like	1.000)	(stylish	-0.999)
(come	1.000)	(facist	-0.999)
(apple_store	1.000)	(stay	-1.000)
(barry_dil	1.000)	(run	-1.000)
(guy	1.000)	(prize	-1.000)
(write	1.000)	(suck	-1.000)
(store	1.000)	(shop	-1.000)
(grab	1.000)	(queue	-1.000)
(subscript	1.000)	(boo	-1.000)

(geekfest	0.997)	(win	-0.613)
(via	0.996)	(day	-0.620)
(head	0.996)	(new	-0.695)
(technolog	0.994)	(product	-0.723)
(sxsw	0.994)	(tv	-0.726)
(bjdproduct	0.992)	(use	-0.726)
(rumor	0.991)	(guy	-0.729)
(downtown	0.991)	(kawasaki	-0.731)
(west	0.989)	(someth	-0.742)
(store	0.985)	(regist	-0.746)

Word Clouds for Positive - Apple Brand Tweets Associated With The Word "Apple"

Associated With Apple

Not Associated With Apple

(sxsw	0.998)	(ever	-0.809)
(tech	0.989)	(heard	-0.819)
(lt	0.982)	(come	-0.821)
(excus	0.982)	(confer	-0.823)
(ausxsw	0.980)	(set	-0.830)
(via	0.979)	(cool	-0.834)
(model	0.979)	(one	-0.834)
(cuz	0.979)	(technolog	-0.837)
(yup	0.979)	(core	-0.841)
(true	0.979)	(go	-0.842)

Word Clouds for Negative - Apple Products Tweets Associated With The Word "Apple"

Associated With Apple

Not Associated With Apple

(open	0.999)	(fade	-0.623)
(austin	0.999)	(fast	-0.629)
(obvious	0.998)	(heard	-0.634)
(peopl	0.998)	(gave	-0.636)
(temporari	0.998)	(among	-0.641)
(noth	0.997)	(japan	-0.643)
(store	0.997)	(money	-0.647)
(want	0.997)	(weekend	-0.655)
(shell	0.997)	(digit	-0.659)
(macbook	0.997)	(novelti	-0.664)

Word Clouds for Neutral - Apple Products Tweets Associated With The Word "Apple"

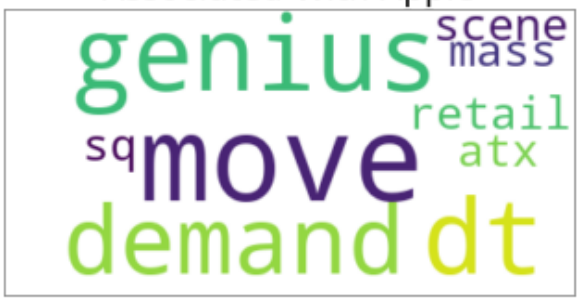
Associated With Apple

Not Associated With Apple

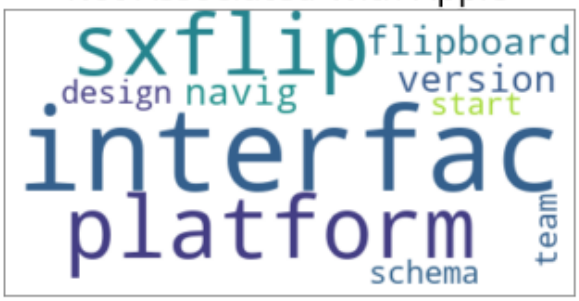
(apple2	0.937)	(platform	-0.034)
(geekfest	0.937)	(flipboard	-0.051)
(ha	0.935)	(team	-0.066)
(omarg	0.933)	(develop	-0.071)
(dt	0.932)	(web	-0.104)
(shut	0.931)	(indi	-0.120)
(handl	0.931)	(tune	-0.149)
(toptweet	0.929)	(pack	-0.149)
(conveni	0.927)	(young	-0.155)
(3rd	0.925)	(surviv	-0.155)

Word Clouds for Positive - Apple Products Tweets Associated With The Word "Apple"

Associated With Apple



Not Associated With Apple



(move	0.967)	(interfac	-0.119)
(genius	0.964)	(platform	-0.186)
(000	0.961)	(sxflip	-0.188)
(dt	0.961)	(flipboard	-0.192)
(demand	0.957)	(version	-0.197)
(retail	0.954)	(navig	-0.201)
(scene	0.954)	(design	-0.204)
(mass	0.954)	(schema	-0.220)
(atx	0.953)	(team	-0.221)
(sq	0.953)	(start	-0.225)

Word Clouds for Negative - Apple (Products & Brand) Tweets Associated With The Word "Apple"

Associated With Apple

Not Associated With Apple

(exist	0.993)
(facist	0.987)
(today	0.987)
(stylish	0.987)
(rji	0.987)
(flipboard	0.985)
(corpor	0.985)
(flip	0.983)
(sxflip	0.981)
(board	0.977)

(fast	-0.524)
(fade	-0.529)
(among	-0.534)
(novelti	-0.542)
(news	-0.550)
(digit	-0.553)
(deleg	-0.554)
(japan	-0.561)
(gave	-0.562)
(money	-0.565)

Word Clouds for Neutral - Apple (Products & Brand) Tweets Associated With The Word "Apple"

Associated With Apple

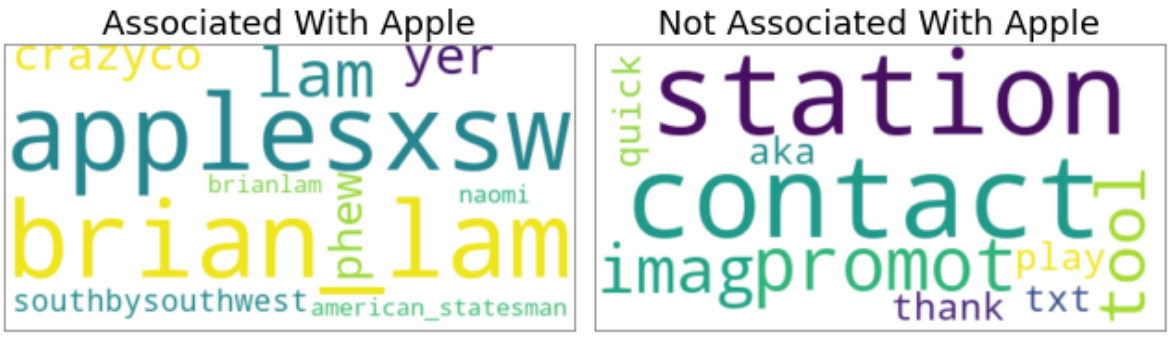
Not Associated With Apple

(realest	0.813)
(foot	0.810)
(rage	0.808)
(demand	0.805)
(bjdproduct	0.805)
(geekfest	0.802)
(brian_lam	0.802)
(campbel	0.800)
(apple2	0.799)
(bait	0.799)

(platform	-0.111)
(flipboard	-0.139)
(pack	-0.142)
(grill	-0.149)
(offici	-0.162)
(magazin	-0.172)
(juic	-0.173)
(young	-0.179)
(fundament	-0.180)
(photo	-0.182)

(texa	0.850)	(song	-0.148)
(tmsxsw	0.841)	(version	-0.158)
(attent	0.837)	(look	-0.159)
(technew	0.836)	(iphone_app	-0.163)
(debut	0.836)	(updat	-0.175)
(despit	0.834)	(info	-0.175)
(amex	0.833)	(ubersoci	-0.181)
(omg	0.827)	(download	-0.185)
(geekfest	0.827)	(sampler	-0.194)
(brilliant	0.826)	(interfac	-0.200)

```
The word "Apple" is not in the model Negative - Google Brand
The word "Apple" is not in the model Neutral - Google Brand
The word "Apple" is not in the model Positive - Google Brand
The word "Apple" is not in the model Negative - Google Products
The word "Apple" is not in the model Neutral - Google Products
The word "Apple" is not in the model Positive - Google Products
The word "Apple" is not in the model Negative - Google (Products & Brand)
The word "Apple" is not in the model Neutral - Google (Products & Brand)
The word "Apple" is not in the model Positive - Google (Products & Brand)
```



```
(applesxsw             0.621)          (contact             -0.012)
(brian_lam             0.609)          (station             -0.021)
(lam                   0.608)          (promot              -0.034)
(yer                   0.603)          (imag                -0.043)
(crazyco               0.600)          (tool                -0.044)
(phew                  0.600)          (thank               -0.045)
(southbysouthwest      0.596)          (play                -0.046)
(american_statesman    0.593)          (txt                 -0.047)
(brianlam              0.593)          (quick               -0.049)
(naomi                 0.590)          (aka                 -0.049)
```

In [77]:
```
word_association_visual(all_wv, 'ipad');
```

```
The word "Ipad" is not in the model Negative - Apple Brand
The word "Ipad" is not in the model Neutral - Apple Brand
The word "Ipad" is not in the model Positive - Apple Brand
```

(via	0.992)	(heard	-0.490)
(media	0.989)	(gave	-0.494)
(aron	0.987)	(japan	-0.500)
(pilhof	0.982)	(money	-0.501)
(2011	0.980)	(weekend	-0.505)
(app	0.975)	(relief	-0.519)
(xipad	0.972)	(best	-0.544)
(ny_tim	0.969)	(thing	-0.546)
(agre	0.968)	(ipad2	-0.616)
(khoi_vinh	0.962)	(need	-0.619)

(learn	0.692)	(come	-0.150)
(may	0.689)	(join	-0.176)
(class	0.684)	(event	-0.180)
(bluefli	0.680)	(poster	-0.181)
(percent	0.677)	(iphone_app	-0.214)
(bliss	0.673)	(pile	-0.216)
(structur	0.670)	(year	-0.217)
(ten	0.668)	(last	-0.226)
(40	0.666)	(game	-0.232)
(piec	0.666)	(ipad_app	-0.233)

Word Clouds for Positive - Apple Products Tweets Associated With The Word "Ipad"

Associated With Ipad

Not Associated With Ipad

(futur	0.911)	(video	-0.286)
(view	0.905)	(via	-0.376)
(mom	0.901)	(itunes	-0.395)
(attend	0.895)	(take	-0.397)
(ratio	0.893)	(stream	-0.398)
(envi	0.891)	(free	-0.398)
(forgot	0.890)	(guid	-0.420)
(haul	0.889)	(song	-0.428)
(type	0.887)	(launch	-0.428)
(comput	0.886)	(mashabl	-0.430)

Word Clouds for Negative - Apple (Products & Brand) Tweets Associated With The Word "Ipad"

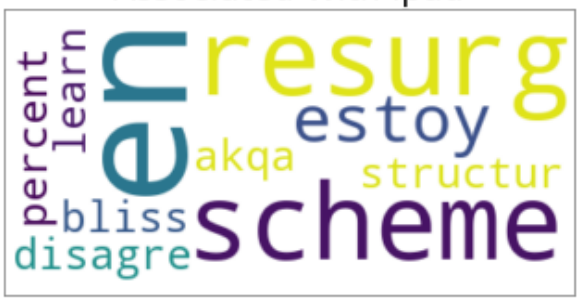
Associated With Ipad

Not Associated With Ipad

(media	0.988)	(money	-0.488)
(aron	0.987)	(japan	-0.491)
(ny_tim	0.984)	(gave	-0.504)
(pilhof	0.982)	(weekend	-0.513)
(agre	0.979)	(relief	-0.515)
(khoi_vinh	0.973)	(heard	-0.521)
(xipad	0.972)	(compani	-0.535)
(via	0.972)	(thing	-0.537)
(bad	0.967)	(best	-0.538)
(type	0.955)	(america	-0.546)

Word Clouds for Neutral - Apple (Products & Brand) Tweets Associated With The Word "Ipad"

Associated With Ipad

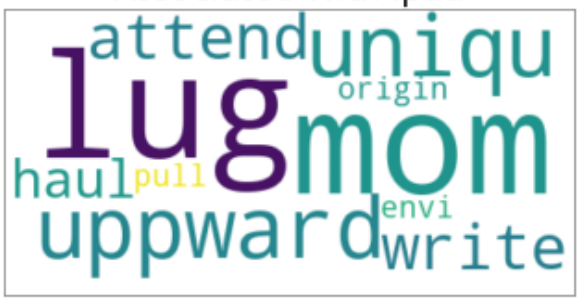


Not Associated With Ipad

(en	0.656)	(join	-0.092)
(resurg	0.656)	(poster	-0.114)
(scheme	0.653)	(iphone_app	-0.137)
(estoy	0.653)	(event	-0.142)
(structur	0.650)	(pile	-0.146)
(disagre	0.650)	(launch	-0.147)
(percent	0.648)	(line	-0.162)
(akqa	0.647)	(week	-0.164)
(bliss	0.645)	(social	-0.174)
(learn	0.644)	(bit	-0.188)

Word Clouds for Positive - Apple (Products & Brand) Tweets Associated With The Word "Ipad"

Associated With Ipad

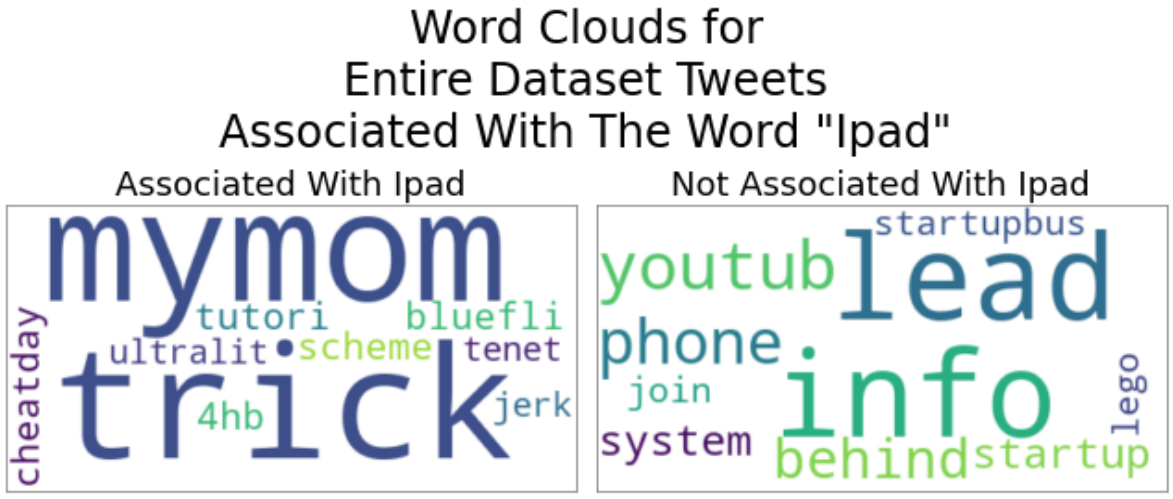


Not Associated With Ipad

```
(mom        0.834)
(lug        0.818)
(uppward    0.812)
(uniqu      0.811)
(attend     0.809)
(write      0.808)
(haul       0.807)
(origin     0.805)
(pull       0.805)
(envi       0.805)
```

```
(launch    -0.246)
(smart     -0.261)
(market    -0.270)
(via       -0.287)
(ly        -0.324)
(expert    -0.327)
(video     -0.329)
(school    -0.337)
(free      -0.340)
(guid      -0.343)
```

```
The word "Ipad" is not in the model Negative - Google Brand
The word "Ipad" is not in the model Neutral - Google Brand
The word "Ipad" is not in the model Positive - Google Brand
The word "Ipad" is not in the model Negative - Google Products
The word "Ipad" is not in the model Neutral - Google Products
The word "Ipad" is not in the model Positive - Google Products
The word "Ipad" is not in the model Negative - Google (Products & Brand)
The word "Ipad" is not in the model Neutral - Google (Products & Brand)
The word "Ipad" is not in the model Positive - Google (Products & Brand)
```




```
(mymom        0.465)
(trick        0.461)
(bluefli      0.459)
(4hb          0.452)
(scheme       0.450)
(tutori       0.447)
(cheatday     0.444)
(ultralit     0.444)
(tenet        0.440)
(jerk         0.439)
```

```
(info        -0.001)
(lead        -0.003)
(youtub      -0.012)
(phone       -0.017)
(behind      -0.018)
(startup     -0.025)
(system      -0.031)
(startupbus  -0.032)
(join        -0.037)
(lego        -0.041)
```

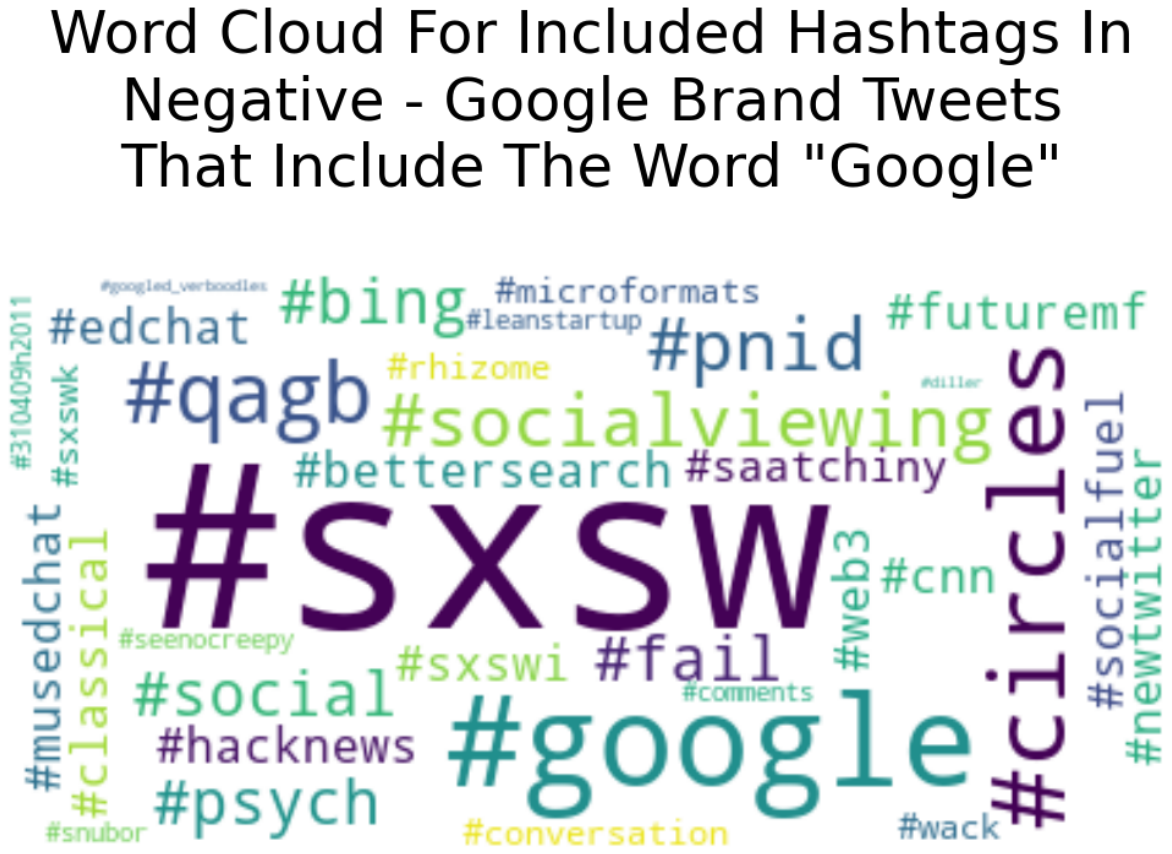
6.5 Hashtag Counter Tool – Workspace

Word associations throughout a dataset are great, but what about intentional associations to link up with other like minded people on Twitter? Enter the hashtag.

As described above, this function lists all the top ten hashtags used in the given word for each data slice. Below I list the hashtags for each word. I'm performing this analysis on for all the

In [78]:
```
hash_count_google = hashtag_search(all_df, 'google')
```

```
The word "Google" is not in the DataFrame Negative - Apple Brand
The word "Google" is not in the DataFrame Neutral - Apple Brand
The word "Google" is not in the DataFrame Positive - Apple Brand
The word "Google" is not in the DataFrame Negative - Apple Products
The word "Google" is not in the DataFrame Neutral - Apple Products
The word "Google" is not in the DataFrame Positive - Apple Products
The word "Google" is not in the DataFrame Negative - Apple (Products & Brand)
The word "Google" is not in the DataFrame Neutral - Apple (Products & Brand)
The word "Google" is not in the DataFrame Positive - Apple (Products & Brand)
```




```
For the Negative - Google Brand set of 98 tweets:
The word "Google" is included in 98/98 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 98)
('#google', 10)
('#circles', 6)
('#qagb', 4)
('#pnid', 3)
('#socialviewing', 3)
('#social', 2)
('#psych', 2)
('#bing', 2)
('#fail', 2)
```

Word Cloud For Included Hashtags In Neutral - Google Brand Tweets That Include The Word "Google"

For the Neutral - Google Brand set of 1207 tweets:
The word "Google" is included in 1177/1207 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 1184)
('#google', 186)
('#circles', 55)
('#sxswi', 46)
('#qagb', 26)
('#gsdm', 24)
('#facebook', 17)
('#socialmedia', 16)
('#marissagoogle', 12)
('#tech', 11)

Word Cloud For Included Hashtags In Positive - Google Brand Tweets That Include The Word "Google"

For the Positive - Google Brand set of 457 tweets:
The word "Google" is included in 455/457 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 455)
('#google', 49)
('#sxswi', 14)
('#circles', 12)
('#marissagoogle', 9)
('#googled_verboodle', 8)
('#gsdm', 8)
('#qagb', 6)
('#911tweets', 5)
('#mobile', 5)

Word Cloud For Included Hashtags In Negative - Google Products Tweets That Include The Word "Google"

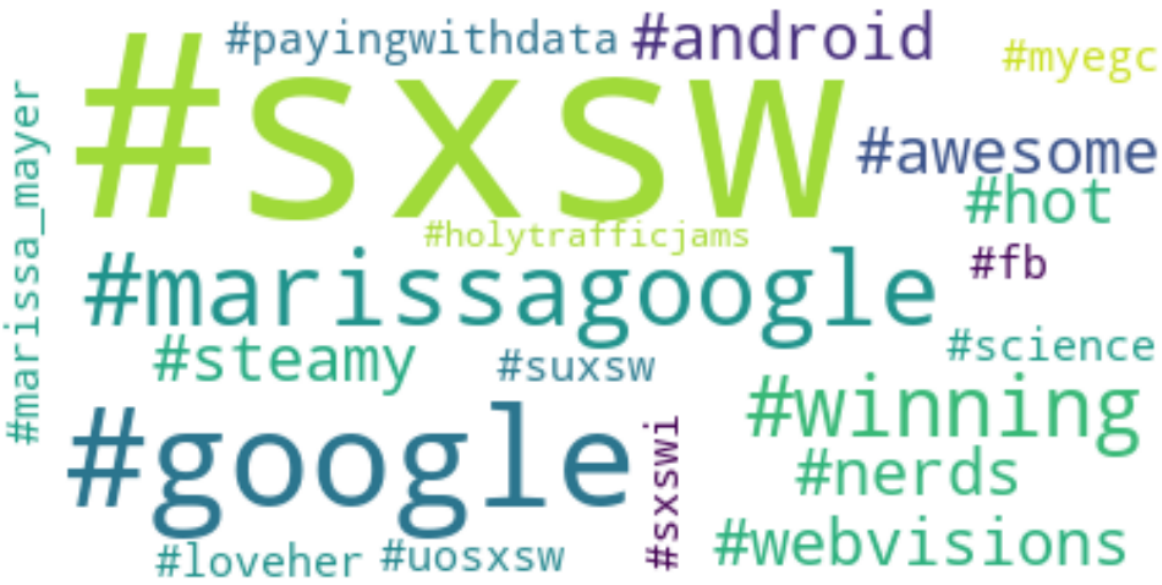

```
For the Negative - Google Products set of 24 tweets:
The word "Google" is included in 10/24 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 10)
('#fh', 1)
('#startupbus', 1)
('#uosxsw', 1)
```

Word Cloud For Included Hashtags In Neutral - Google Products Tweets That Include The Word "Google"

For the Neutral - Google Products set of 235 tweets:
The word "Google" is included in 84/235 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 84)
('#google', 5)
('#mobile', 3)
('#tc', 2)
('#donline', 2)
('#blogger', 2)
('#ie9', 2)
('#marissagoogle', 2)
('#virtualwallet', 1)
('#digitalid', 1)

Word Cloud For Included Hashtags In Positive - Google Products Tweets That Include The Word "Google"



For the Positive - Google Products set of 206 tweets:
The word "Google" is included in 86/206 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 86)
('#google', 6)
('#marissagoogle', 3)
('#winning', 2)
('#android', 1)
('#awesome', 1)
('#webvisions', 1)
('#steamy', 1)
('#hot', 1)
('#nerds', 1)

Word Cloud For Included Hashtags In Negative - Google (Products & Brand) Tweets That Include The Word "Google"

For the Negative - Google (Products & Brand) set of 122 tweets:
The word "Google" is included in 108/122 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 108)
('#google', 10)
('#circles', 6)
('#qagb', 4)
('#pnid', 3)
('#socialviewing', 3)
('#social', 2)
('#psych', 2)
('#bing', 2)
('#fail', 2)

Word Cloud For Included Hashtags In Neutral - Google (Products & Brand) Tweets That Include The Word "Google"

For the Neutral - Google (Products & Brand) set of 1442 tweets:
The word "Google" is included in 1261/1442 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 1268)
('#google', 191)
('#circles', 55)
('#sxswi', 47)
('#qagb', 26)
('#gsdm', 24)
('#facebook', 17)
('#socialmedia', 16)
('#marissagoogle', 14)
('#tech', 11)

Word Cloud For Included Hashtags In Positive - Google (Products & Brand) Tweets That Include The Word "Google"

For the Positive - Google (Products & Brand) set of 663 tweets:
The word "Google" is included in 541/663 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 541)
('#google', 55)
('#sxswi', 15)
('#circles', 12)
('#marissagoogle', 12)
('#googled_verboodle', 8)
('#gsdm', 8)
('#qagb', 6)
('#911tweets', 5)
('#mobile', 5)

Word Cloud For Included Hashtags In Entire Dataset Tweets That Include The Word "Google"

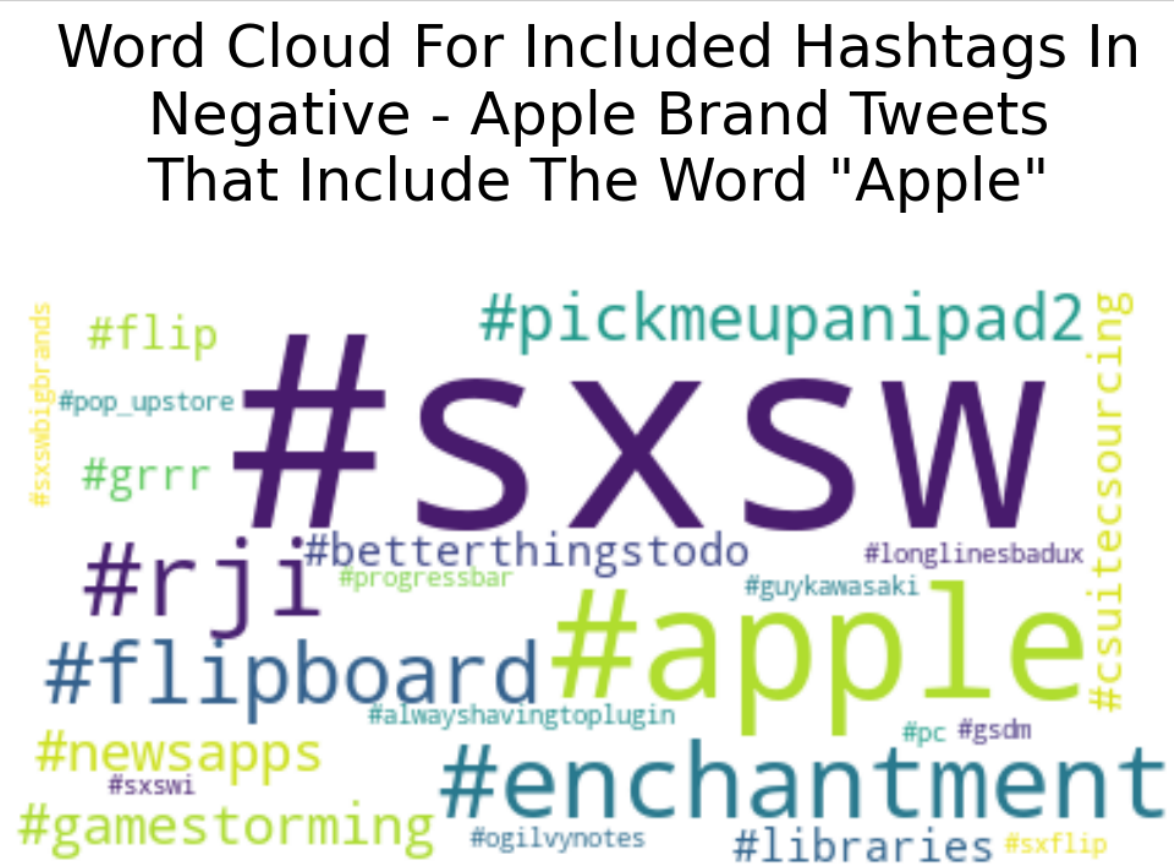
For the Entire Dataset set of 9092 tweets:
The word "Google" is included in 2455/9092 tweets.

Top ten Hashtags included with the word "Google" in this dataset:
('#sxsw', 2463)
('#google', 322)
('#circles', 98)
('#sxswi', 85)
('#qagb', 50)
('#gsdm', 37)
('#marissagoogle', 27)
('#911tweets', 22)
('#socialmedia', 21)
('#facebook', 21)

In [79]:
```
hash_count_android = hashtag_search(all_df, 'android')
```

```
The word "Android" is not in the DataFrame Negative - Apple Brand
The word "Android" is not in the DataFrame Neutral - Apple Brand
The word "Android" is not in the DataFrame Positive - Apple Brand
The word "Android" is not in the DataFrame Negative - Apple Products
The word "Android" is not in the DataFrame Neutral - Apple Products
The word "Android" is not in the DataFrame Positive - Apple Products
The word "Android" is not in the DataFrame Negative - Apple (Products & Br
and)
The word "Android" is not in the DataFrame Neutral - Apple (Products & Bra
nd)
The word "Android" is not in the DataFrame Positive - Apple (Products & Br
and)
The word "Android" is not in the DataFrame Negative - Google Brand
The word "Android" is not in the DataFrame Neutral - Google Brand
The word "Android" is not in the DataFrame Positive - Google Brand
```

In [80]:
```
hash_count_apple = hashtag_search(all_df, 'apple')
```



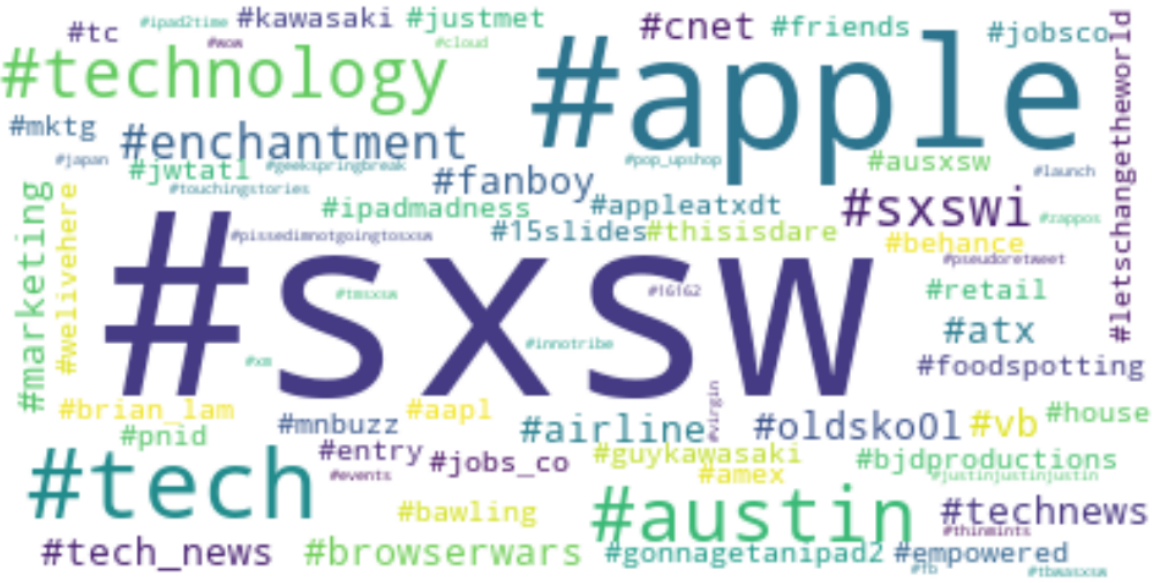

```
For the Negative - Apple Brand set of 60 tweets:
The word "Apple" is included in 47/60 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 47)
('#apple', 6)
('#rji', 2)
('#enchantment', 2)
('#flipboard', 2)
('#pickmeupanipad2', 1)
('#gamestorming', 1)
('#newsapps', 1)
('#betterthingstodo', 1)
('#grrr', 1)
```

Word Cloud For Included Hashtags In Neutral - Apple Brand Tweets That Include The Word "Apple"

For the Neutral - Apple Brand set of 548 tweets:
The word "Apple" is included in 387/548 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 388)
('#apple', 100)
('#tech', 21)
('#austin', 13)
('#technology', 12)
('#enchantment', 6)
('#winning', 5)
('#bjdproductions', 5)
('#news', 4)
('#gamestorming', 3)

Word Cloud For Included Hashtags In Positive - Apple Brand Tweets That Include The Word "Apple"



For the Positive - Apple Brand set of 361 tweets:
The word "Apple" is included in 287/361 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 291)
('#apple', 61)
('#tech', 15)
('#austin', 9)
('#technology', 7)
('#sxswi', 4)
('#enchantment', 3)
('#cnet', 2)
('#technews', 2)
('#tech_news', 2)

Word Cloud For Included Hashtags In Negative - Apple Products Tweets That Include The Word "Apple"

For the Negative - Apple Products set of 292 tweets:
The word "Apple" is included in 29/292 tweets.

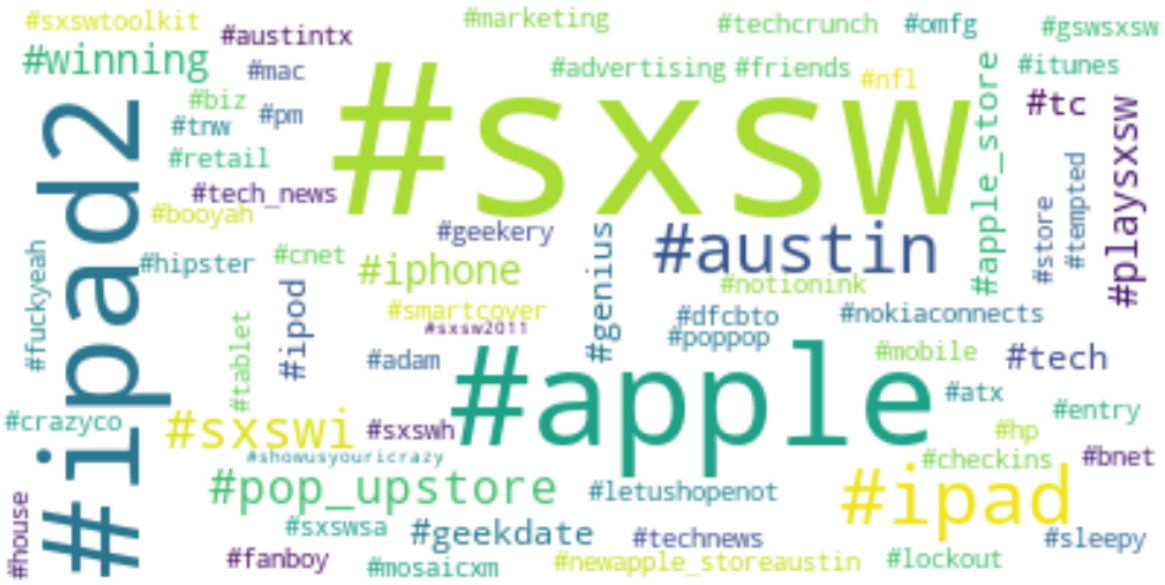
Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 29)
('#apple', 8)
('#ipad2', 3)
('#iphone', 2)
('#ipad', 2)
('#precommerce', 1)
('#jpmobilesummit', 1)
('#wth', 1)
('#whowillrise', 1)
('#nowhammies', 1)

Word Cloud For Included Hashtags In Neutral - Apple Products Tweets That Include The Word "Apple"

For the Neutral - Apple Products set of 1733 tweets:
The word "Apple" is included in 267/1733 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 267)
('#ipad2', 85)
('#apple', 65)
('#ipad', 11)
('#austin', 11)
('#tech', 9)
('#sxswi', 6)
('#fb', 6)
('#technology', 4)
('#sxswsa', 4)

Word Cloud For Included Hashtags In Positive - Apple Products Tweets That Include The Word "Apple"



For the Positive - Apple Products set of 1400 tweets:
The word "Apple" is included in 266/1400 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 268)
('#apple', 76)
('#ipad2', 72)
('#ipad', 13)
('#austin', 10)
('#sxswi', 5)
('#pop_upstore', 4)
('#winning', 3)
('#playsxsw', 3)
('#iphone', 3)

Word Cloud For Included Hashtags In Negative - Apple (Products & Brand) Tweets That Include The Word "Apple"

For the Negative - Apple (Products & Brand) set of 352 tweets:
The word "Apple" is included in 76/352 tweets.

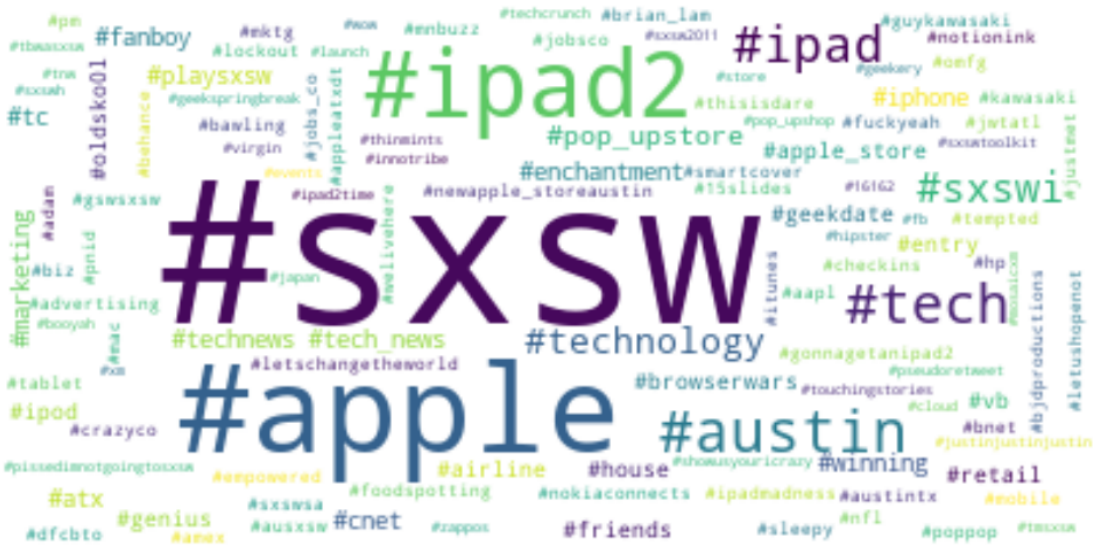
Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 76)
('#apple', 14)
('#ipad2', 3)
('#rji', 3)
('#flipboard', 3)
('#iphone', 2)
('#enchantment', 2)
('#sxswi', 2)
('#ipad', 2)
('#pickmeupanipad2', 1)

Word Cloud For Included Hashtags In Neutral - Apple (Products & Brand) Tweets That Include The Word "Apple"

For the Neutral - Apple (Products & Brand) set of 2281 tweets:
The word "Apple" is included in 654/2281 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 655)
('#apple', 165)
('#ipad2', 85)
('#tech', 30)
('#austin', 24)
('#technology', 16)
('#ipad', 11)
('#sxswi', 8)
('#winning', 7)
('#fb', 7)

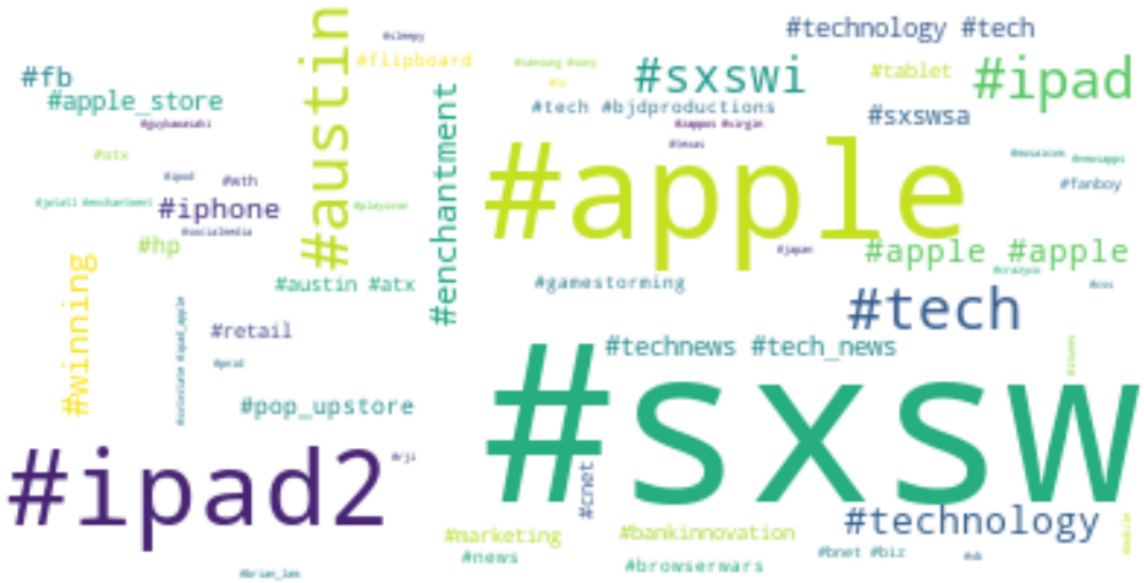
Word Cloud For Included Hashtags In Positive - Apple (Products & Brand) Tweets That Include The Word "Apple"



For the Positive - Apple (Products & Brand) set of 1761 tweets:
The word "Apple" is included in 553/1761 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 559)
('#apple', 137)
('#ipad2', 72)
('#austin', 19)
('#tech', 17)
('#ipad', 13)
('#sxswi', 9)
('#technology', 7)
('#pop_upstore', 4)
('#enchantment', 3)
The word "Apple" is not in the DataFrame Negative - Google Brand
The word "Apple" is not in the DataFrame Neutral - Google Brand
The word "Apple" is not in the DataFrame Positive - Google Brand
The word "Apple" is not in the DataFrame Negative - Google Products
The word "Apple" is not in the DataFrame Neutral - Google Products
The word "Apple" is not in the DataFrame Positive - Google Products
The word "Apple" is not in the DataFrame Negative - Google (Products & Brand)
The word "Apple" is not in the DataFrame Neutral - Google (Products & Brand)
The word "Apple" is not in the DataFrame Positive - Google (Products & Brand)

Word Cloud For Included Hashtags In Entire Dataset Tweets That Include The Word "Apple"



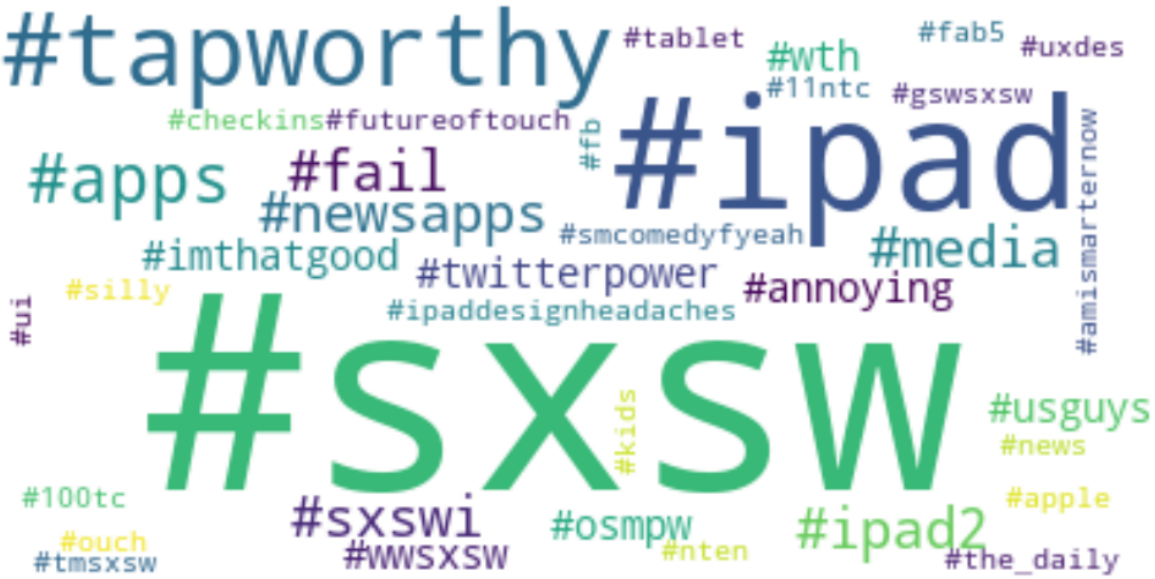
For the Entire Dataset set of 9092 tweets:
The word "Apple" is included in 1606/9092 tweets.

Top ten Hashtags included with the word "Apple" in this dataset:
('#sxsw', 1615)
('#apple', 373)
('#ipad2', 196)
('#austin', 52)
('#tech', 50)
('#ipad', 33)
('#sxswi', 24)
('#technology', 24)
('#enchantment', 16)
('#winning', 12)

In [81]:
```
hash_count_ipad = hashtag_search(all_df, 'ipad')
```

```
The word "Ipad" is not in the DataFrame Negative - Apple Brand
The word "Ipad" is not in the DataFrame Neutral - Apple Brand
The word "Ipad" is not in the DataFrame Positive - Apple Brand
```

Word Cloud For Included Hashtags In Negative - Apple Products Tweets That Include The Word "Ipad"




```
For the Negative - Apple Products set of 292 tweets:
The word "Ipad" is included in 98/292 tweets.

Top ten Hashtags included with the word "Ipad" in this dataset:
('#sxsw', 98)
('#ipad', 12)
('#tapworthy', 9)
('#apps', 3)
('#ipad2', 2)
('#fail', 2)
('#media', 2)
('#newsapps', 2)
('#sxswi', 2)
('#osmpw', 1)
```

Word Cloud For Included Hashtags In Neutral - Apple Products Tweets That Include The Word "Ipad"

For the Neutral - Apple Products set of 1733 tweets:
The word "Ipad" is included in 584/1733 tweets.

Top ten Hashtags included with the word "Ipad" in this dataset:
('#sxsw', 585)
('#ipad', 101)
('#uxdes', 23)
('#sxswi', 19)
('#apple', 17)
('#tapworthy', 15)
('#ipad2', 12)
('#iphone', 9)
('#touchingstories', 9)
('#art', 8)

Word Cloud For Included Hashtags In Positive - Apple Products Tweets That Include The Word "Ipad"

For the Positive - Apple Products set of 1400 tweets:
The word "Ipad" is included in 444/1400 tweets.

Top ten Hashtags included with the word "Ipad" in this dataset:
('#sxsw', 447)
('#ipad', 58)
('#apple', 17)
('#tapworthy', 14)
('#sxswi', 13)
('#ipad2', 9)
('#poursite', 7)
('#newsapps', 5)
('#uxdes', 5)
('#cbatsxsw', 5)

Word Cloud For Included Hashtags In Negative - Apple (Products & Brand) Tweets That Include The Word "Ipad"

For the Negative - Apple (Products & Brand) set of 352 tweets:
The word "Ipad" is included in 98/352 tweets.

Top ten Hashtags included with the word "Ipad" in this dataset:
('#sxsw', 98)
('#ipad', 12)
('#tapworthy', 9)
('#apps', 3)
('#ipad2', 2)
('#fail', 2)
('#media', 2)
('#newsapps', 2)
('#sxswi', 2)
('#osmpw', 1)

Word Cloud For Included Hashtags In Neutral - Apple (Products & Brand) Tweets That Include The Word "Ipad"

For the Neutral - Apple (Products & Brand) set of 2281 tweets:
The word "Ipad" is included in 584/2281 tweets.

Top ten Hashtags included with the word "Ipad" in this dataset:
('#sxsw', 585)
('#ipad', 101)
('#uxdes', 23)
('#sxswi', 19)
('#apple', 17)
('#tapworthy', 15)
('#ipad2', 12)
('#iphone', 9)
('#touchingstories', 9)
('#art', 8)

Word Cloud For Included Hashtags In Positive - Apple (Products & Brand) Tweets That Include The Word "Ipad"

For the Positive - Apple (Products & Brand) set of 1761 tweets:
The word "Ipad" is included in 444/1761 tweets.

Top ten Hashtags included with the word "Ipad" in this dataset:
('#sxsw', 447)
('#ipad', 58)
('#apple', 17)
('#tapworthy', 14)
('#sxswi', 13)
('#ipad2', 9)
('#poursite', 7)
('#newsapps', 5)
('#uxdes', 5)
('#cbatsxsw', 5)
The word "Ipad" is not in the DataFrame Negative - Google Brand
The word "Ipad" is not in the DataFrame Neutral - Google Brand
The word "Ipad" is not in the DataFrame Positive - Google Brand
The word "Ipad" is not in the DataFrame Negative - Google Products
The word "Ipad" is not in the DataFrame Neutral - Google Products
The word "Ipad" is not in the DataFrame Positive - Google Products
The word "Ipad" is not in the DataFrame Negative - Google (Products & Brand)
The word "Ipad" is not in the DataFrame Neutral - Google (Products & Brand)
The word "Ipad" is not in the DataFrame Positive - Google (Products & Brand)

Word Cloud For Included Hashtags In Entire Dataset Tweets That Include The Word "Ipad"


```
For the Entire Dataset set of 9092 tweets:
The word "Ipad" is included in 1381/9092 tweets.

Top ten Hashtags included with the word "Ipad" in this dataset:
('#sxsw', 1388)
('#ipad', 213)
('#tapworthy', 46)
('#apple', 44)
('#sxswi', 42)
('#uxdes', 39)
('#ipad2', 32)
('#newsapps', 23)
('#iphone', 19)
('#fb', 13)
```

6.6 Word Associations Conclusions.

I read each of the above lists and grabbed words and hashtags that were most practically applicable to the word triggering the lists. Here is a list of all of the associated words and hashtags.

In [82]:
```
# Google
key_assoc_negative_google  = ['trend', 'buzz', 'app', 'android',
                             'imag', 'maps', 'marissa_mayer']
key_assoc_neutral_google   = ['marissa_mayer', 'maps']
key_assoc_positive_google  = ['maps', 'navi', 'expect', 'evolv',
                             'fast', 'cool', 'video', 'circles']

hashtags_negative_google   = ['#circles', '#social', '#bing']
hashtags_neutral_google    = ['#circles', '#facebook', '#socialmedia',
                             '#marissagoogle']
hashtags_positive_google   = ['#circles', '#marissagoogle']

# Android
key_assoc_negative_android = ['google', 'app', 'phon',
                              'updat', 'vers', 'opt']
key_assoc_neutral_android  = ['team', 'award', 'samsung', 'chrome', 'detail']
key_assoc_positive_android = ['congrat', 'job', 'huzzah', 'awesom', 'thank',
                              'android_app', 'choic']

hashtags_negative_android  = ['#fail']
hashtags_neutral_android   = ['#tech', '#app']
hashtags_positive_android  = ['#teamandroid']

# Apple
key_assoc_negative_apple   = ['apple_store', 'store', 'macbook', 'facist',
                              'stylish', 'flip']
key_assoc_neutral_apple    = ['geekfest', 'technolog', 'rumor', 'store',
                              'realest', 'bait']
key_assoc_positive_apple   = ['sxsw', 'excus', 'model', 'genius', 'demand',
                              'retail', 'scene', 'debut', 'brilliant']

hashtags_negative_apple    = ['#sxsw', '#apple', '#flipboard', '#ipad2']
hashtags_neutral_apple     = ['#sxsw', '#apple', '#ipad2', '#tech',
                              '#apple_store']
hashtags_positive_apple    = ['#sxsw', '#apple', '#tech', '#technology']

# iPad
key_assoc_negative_ipad    = ['media', 'app', ]
key_assoc_neutral_ipad     = ['bliss', 'structur', 'book', 'ultim', 'present',
                              'scheme']
key_assoc_positive_ipad    = ['futur', 'mom', 'envi', 'media']

hashtags_negative_ipad     = ['#sxsw', '#ipad', '#tapworthy']
hashtags_neutral_ipad      = ['#sxsw', '#ipad', '#apple', '#tapworthy',
                              '#touchingstories', '#art']
hashtags_positive_ipad     = ['#sxsw', '#ipad', '#apple', '#tapworthy',
                              '#poursite', '#newsapps']

# Combinations for function below.
google_associations  = [key_assoc_negative_google, key_assoc_neutral_google,
                        key_assoc_positive_google]
android_associations = [key_assoc_negative_android, key_assoc_neutral_android,
                        key_assoc_positive_android]

apple_associations   = [key_assoc_negative_apple, key_assoc_neutral_apple,
                        key_assoc_positive_apple]
```

```
ipad_associations    = [key_assoc_negative_ipad, key_assoc_neutral_ipad,
                        key_assoc_positive_ipad]
```

The function below displays lists of how often each word in a list appears in tweets that include the associated word. It splits the lists into the three emotions, negative, neutral and positive. For example, these lists will tell you how many times the word "Trend" appears in negative tweets that include the word "Google."

In [83]:
```
def association_count(list_of_lists, model_list, word):
    emo = ['Negative', 'Neutral', 'Positive']
    colors = ['Reds', 'Greens', 'Blues']
    for e, lst, mdl, c in zip(emo, list_of_lists, model_list, colors):
        tweet_count = 0
        count = []
        for i in mdl.index:
            for stm in mdl.loc[i,'Stemmed Tokens']:
                if word == stm:
                    tweet_count +=1
                    for wrd in lst:
                        if wrd in mdl.loc[i,'Stemmed Tweet']:
                            count.append(wrd)
        if tweet_count == 0:
            print('There are no occurences of "' + word.capitalize() +
                  '" in the ' + e +' emotion tweets.')
        else:
            string = 'Associated word occurrences found in the '
            string += str(tweet_count) + ' ' + e +' emotion tweets'
            string_two = 'that include the word "' + word.capitalize() + '":'
            print(string)
            print(string_two)
            column_title = e + ' Tweets that include "'+ word.capitalize() + '
            count_total = pd.DataFrame(count)
            count_total.rename(columns = {0:'Word'}, inplace = True)
            count_total = count_total.groupby('Word')['Word'].count()
            count_total = pd.DataFrame(count_total)
            count_total.rename(columns = {'Word':column_title}, inplace = True
            count_total = count_total.sort_values(column_title,
                                                  ascending = False)
            count_total.reset_index(drop=False, inplace=True)
            display(count_total.style.background_gradient(cmap=c))
    return None
```

6.6.1 "Google" Word Association

In [84]:
```
association_count(google_associations, google_df, 'google')
```

```
Associated word occurrences found in the 122 Negative emotion tweets
that include the word "Google":
```

	Word	Negative Tweets that include "Google"
0	maps	10
1	app	8
2	marissa_mayer	8
3	buzz	6
4	android	1
5	imag	1
6	trend	1

```
Associated word occurrences found in the 1368 Neutral emotion tweets
that include the word "Google":
```

	Word	Neutral Tweets that include "Google"
0	marissa_mayer	125
1	maps	97

```
Associated word occurrences found in the 581 Positive emotion tweets
that include the word "Google":
```

	Word	Positive Tweets that include "Google"
0	circles	96
1	maps	93
2	cool	26
3	fast	11
4	navi	7
5	video	5
6	evolv	4
7	expect	4

"Google" Word Association Analysis Takeaways:

1. People like to tweet about Google Maps. Their opinions of it are mostly positive.
2. Tweets regarding Marissa Meyer are more neutral than anything else.
3. People think Google is "cool" and associate it with being "fast".

6.6.2 "Android" Word Association

In [85]:
```
association_count(android_associations, google_df, 'android')
```

```
Associated word occurrences found in the 12 Negative emotion tweets
that include the word "Android":
```

	Word	Negative Tweets that include "Android"
0	app	5
1	phon	3
2	opt	2
3	google	1
4	updat	1
5	vers	1

```
Associated word occurrences found in the 144 Neutral emotion tweets
that include the word "Android":
```

	Word	Neutral Tweets that include "Android"
0	team	20
1	samsung	9
2	chrome	8
3	detail	7
4	award	6

```
Associated word occurrences found in the 108 Positive emotion tweets
that include the word "Android":
```

	Word	Positive Tweets that include "Android"
0	thank	12
1	choic	11
2	android_app	10
3	congrat	5
4	awesom	3
5	job	3
6	huzzah	1

"Android" Word Association Analysis Takeaways:

1. People tweet complaints about apps on android, sometimes.
2. People who have android products seem to be happy with their choice.

6.6.3 "Apple" Word Association

In [86]:
```
association_count(apple_associations, apple_df, 'apple')
```

```
Associated word occurrences found in the 81 Negative emotion tweets
that include the word "Apple":
```

	Word	Negative Tweets that include "Apple"
0	store	20
1	flip	5
2	apple_store	2
3	macbook	2
4	facist	1
5	stylish	1

```
Associated word occurrences found in the 712 Neutral emotion tweets
that include the word "Apple":
```

	Word	Neutral Tweets that include "Apple"
0	store	402
1	rumor	47
2	technolog	25
3	geekfest	16
4	realest	2
5	bait	1

```
Associated word occurrences found in the 600 Positive emotion tweets
that include the word "Apple":
```

	Word	Positive Tweets that include "Apple"
0	sxsw	600
1	genius	16
2	brilliant	14
3	retail	11
4	demand	3
5	model	3
6	scene	2
7	debut	1
8	excus	1

"Apple" Word Association Analysis Takeaways:

1. People are enjoying the experience of being at SXSW and associate it with the Apple store.
2. People associate a lot of positive words with Apple, like genius and brilliant.

6.6.4 "iPad" Word Association

In [87]:
```
association_count(ipad_associations, apple_df, 'ipad')
```

```
Associated word occurrences found in the 106 Negative emotion tweets
that include the word "Ipad":
```

	Word	Negative Tweets that include "Ipad"
0	app	33
1	media	3

```
Associated word occurrences found in the 603 Neutral emotion tweets
that include the word "Ipad":
```

	Word	Neutral Tweets that include "Ipad"
0	book	19
1	present	5
2	scheme	2
3	ultim	2
4	bliss	1
5	structur	1

```
Associated word occurrences found in the 462 Positive emotion tweets
that include the word "Ipad":
```

	Word	Positive Tweets that include "Ipad"
0	envi	11
1	mom	7
2	futur	6
3	media	3

"iPad" Word Association Analysis Takeaways:

1. People are complaining about apps on iPad, likely they want apps that are not available yet.
2. People plan to read books on their iPad.
3. People want to give iPads as presents, to their mom and others.
4. People associate the future with iPad.

So here are the word analyses of the top keyword in each of the company / brand or product categories. Rather than digging into the set any further, I think it makes a lot more sense to empower the client, Google, to perform these searches on their own. I propose that Google hires our company to turn this analysis into a tool of its own. It will empower not just data scientists to search for word associations, but brand managers, marketers, public relations specialists, even HR managers looking for qualities in who to hire.

I recommend we move forward to make this tool accessible to multiple, non-python coders, at Google.