jupyter Team Project - Elon Musk Last Checkpoint: 9 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

In [25]:
```
import pandas as pd
import bs4
import urllib
#bs4 pulls data out of HTML and XML
```

In [26]:
```
import matplotlib.pyplot as plt
import re
from nltk.corpus import stopwords
import os
# NLTK = Natural Language Toolkit, provides tools for loading and cleaning text.
```

In [27]:
```
f = open("Musk.txt", 'r')
text=f.read()
```

In [31]:
```
text
```

Out[31]:
```
'ï»¿I\'m going to talk more about what it takes to become a multi-planet species. And just a brief refresher on why this is i
mportant: I think fundamentally the future is vastly more exciting and interesting if we\'re a space-faring civilization and
a multi-planet species than if we\'re not.\nYou want to be inspired by things. You want to wake up in the morning and think
"the future\'s going to be great". And that\'s what being a space-faring civilization is all about. It\'s about believing in
the future and thinking that the future will be better than the past. And I can\'t think of anything more exciting than going
out there and being among the stars. That\'s why.\nSo let me go into more detail on becoming a multi-planet species. This is
the updated design for the â€" well, we\'re sort of searching for the right name, but the code name, at least, is BFR. Probab
ly the most important thing that I want to convey in this presentation is that I think we have figured out how to pay for it.
This is very important.\nIn last year\'s presentation, we were really searching for what the right way... how do we pay for t
his thing? We went through various ideas, Kickstarter, collecting underpants, these didn\'t pan out. But now we think we\'ve
got a way to do it, which is to have a smaller vehicle â€" it\'s still pretty big â€" but one that can do everything that\'s
needed in the greater-Earth-orbit activity. So essentially, we want to make our current vehicles redundant. We want to have o
ne system, one booster and ship, that replaces Falcon 9, Falcon Heavy, and Dragon.\nIf we can do that, then all the resources
that are used for Falcon 9, Heavy, and Dragon can be applied to this system. That\'s really fundamental.\nSo let\'s see what
progress have we made in this direction.\nLast time you saw the giant tank â€" that\'s actually a 12-meter tank. It\'s 1,000
cubic meters of volume inside. That\'s actually more pressurized volume than an [Airbus] A380, to put that into perspectiv
e.\nWe developed a new carbon fiber matrix that\'s much stronger and more capable at cryo than anything before, and it holds
1,200 tons of liquid oxygen.\nSo we tested it. We successfully tested it up to its design pressure â€" and then went a little
further. So we wanted to see where it would break, and we found out. It shot about 300 feet into the air and landed in the oc
```

jupyter Team Project - Elon Musk Last Checkpoint: 9 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 (ipykernel)

Run Code

In [32]:
```
# By using the regex model (re), we split the document into words by selecting for strings of alphanumeric characters [^a-zA-Z0-9
expression = "[^a-zA-Z0-9 ]"
cleantextCAP = re.sub(expression, '', text)
cleantext = cleantextCAP.lower()
```

In [35]:
```
text_file = open("OutputMusk.txt", "w")
text_file.write(str(cleantext))
text_file.close()
```

In [36]:
```
dat = list(cleantext.split()) #We attempted to split the text into a list. Then, we counted the words.
dict1 = {}
for i in range(len(dat)):
    print(i)
    word = dat[i]
    dict1[word] = dat.count(word)
```

```
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
```

In [37]:
```
#Stopwords do not contribute to the deeper meaning of the phrase. These are the most common used words ("as", "we", "you", "is" e
#However, the stopwords package wasn't enough for our project, because it still didn't filter all the unnecessary words from our
keys = list(dict1)
filtered_words = [word for word in keys if word not in stopwords.words('english')]
dict2  = dict((k, dict1[k]) for k in filtered_words if k in filtered_words)

print(filtered_words)
```

In [37]:
```
#Stopwords do not contribute to the deeper meaning of the phrase. These are the most common used words ("as", "we", "you", "is" e
#However, the stopwords package wasn't enough for our project, because it still didn't filter all the unnecessary words from our
keys = list(dict1)
filtered_words = [word for word in keys if word not in stopwords.words('english')]
dict2  = dict((k, dict1[k]) for k in filtered_words if k in filtered_words)

print(filtered_words)
```

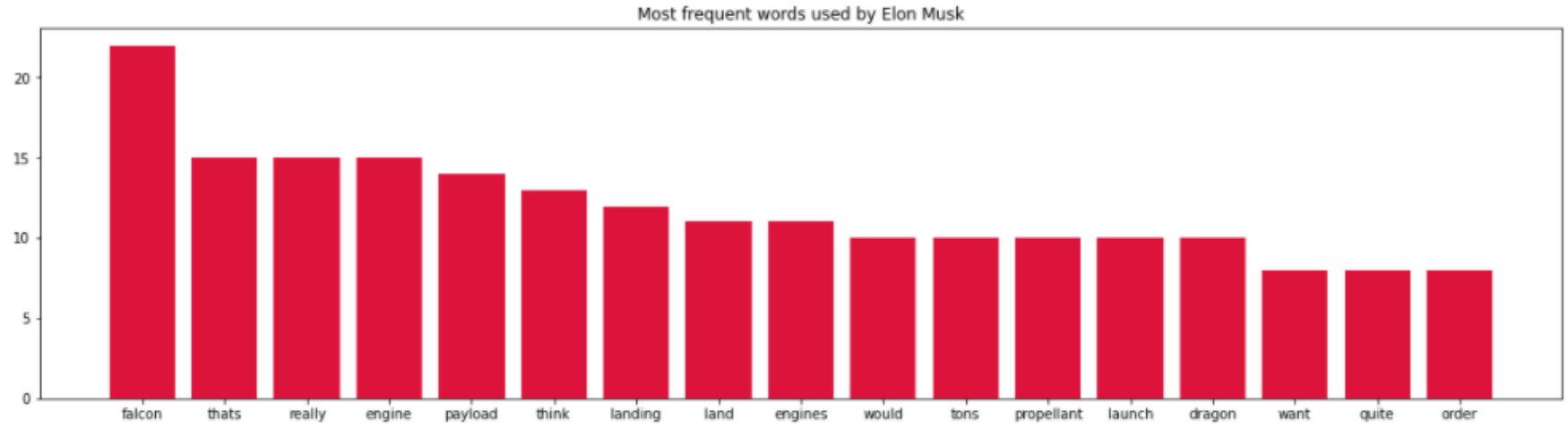
```
['im', 'going', 'talk', 'takes', 'become', 'multiplanet', 'species', 'brief', 'refresher', 'important', 'think', 'fundamentall
y', 'future', 'vastly', 'exciting', 'interesting', 'spacefaring', 'civilization', 'notyou', 'want', 'inspired', 'things', 'wak
e', 'morning', 'futures', 'great', 'thats', 'believing', 'thinking', 'better', 'past', 'cant', 'anything', 'among', 'stars', 'w
hyso', 'let', 'go', 'detail', 'becoming', 'updated', 'design', 'well', 'sort', 'searching', 'right', 'name', 'code', 'least',
'bfr', 'probably', 'thing', 'convey', 'presentation', 'figured', 'pay', 'importantin', 'last', 'years', 'really', 'way', 'wen
t', 'various', 'ideas', 'kickstarter', 'collecting', 'underpants', 'didnt', 'pan', 'weve', 'got', 'smaller', 'vehicle', 'stil
l', 'pretty', 'big', 'one', 'everything', 'needed', 'greaterearthorbit', 'activity', 'essentially', 'make', 'current', 'vehicle
s', 'redundant', 'system', 'booster', 'ship', 'replaces', 'falcon', '9', 'heavy', 'dragonif', 'resources', 'used', 'dragon', 'a
pplied', 'fundamentalso', 'lets', 'see', 'progress', 'made', 'directionlast', 'time', 'saw', 'giant', 'tank', 'actually', '12me
ter', '1000', 'cubic', 'meters', 'volume', 'inside', 'pressurized', 'airbus', 'a380', 'put', 'perspectivewe', 'developed', 'ne
w', 'carbon', 'fiber', 'matrix', 'much', 'stronger', 'capable', 'cryo', 'holds', '1200', 'tons', 'liquid', 'oxygenso', 'teste
d', 'successfully', 'pressure', 'little', 'wanted', 'would', 'break', 'found', 'shot', '300', 'feet', 'air', 'landed', 'ocean',
'fished', 'outweve', 'good', 'sense', 'create', 'huge', 'hold', 'cryogenic', 'extremely', 'making', 'light', 'spaceshipthe', 'n
ext', 'key', 'element', 'engine', 'side', 'efficient', 'raptor', 'highest', 'thrust', 'weight', 'ratio', 'kind', 'ever', 'alrea
dy', 'seconds', 'firing', 'across', '42', 'main', 'tests', 'fired', '100', 'could', 'fire', 'longer', 'size', 'test', 'tanksth
e', 'duration', '40', 'length', 'landing', 'mars', 'currently', 'operates', '200', 'atmospheres', 'bar', 'flight', '250', 'beli
eve', 'get', 'barthe', 'propulsive', 'order', 'land', 'places', 'like', 'moon', 'atmosphere', 'certainly', 'runways', 'thin',
'even', 'wing', 'perfect', 'practicing', 'series', 'videos', 'quite', 'mesmerizing', '16', 'successful', 'landings', 'row', 'ib
id', '12', 'though', 'total', 'without', 'redundancy', 'lands', 'single', 'final', 'always', 'done', 'whereas', 'multiengineou
t', 'capabilityif', 'high', 'reliability', 'either', 'two', 'engines', 'par', 'safest', 'commercial', 'airliners', 'count', 'la
ndingand', 'also', 'precision', 'fact', 'point', 'enough', 'need', 'legs', 'version', 'back', 'launch', 'mountsthe', 'rate', 'i
ncreasing', 'exponentially', 'particularly', 'take', 'refilling', 'onorbit', 'account', 'taking', 'idea', 'establishing', 'self
sustaining', 'base', 'elsewhere', 'seriously', 'thousands', 'ships', 'tens', 'retanking', 'rerefilling', 'operations', 'means',
'many', 'launches', 'per', 'day', 'terms', 'occurring', 'looking', 'watch', 'calendar', 'conventional', 'standards', 'every',
'small', 'compared', 'ultimately', 'neededfor', 'unfamiliar', 'orbital', 'occur', 'year', 'approximately', '60', 'spacex', 'som
ething', '30', 'itll', 'half', 'earththe', 'technology', 'automated', 'rendezvous', 'dockingin', 'retank', 'refill', 'spaceshi
p', 'orbit', 'able', 'dock', 'transfer', 'propellant', 'perfected', '2', 'docking', 'pilot', 'control', 'space', 'station',
'1', 'uses', 'canadarm', 'placement', 'onto', 'use', 'directly', 'zero', 'human', 'intervention', 'press', 'dockdragon', 'allow
ed', 'us', 'heat', 'shield', 'enter', 'velocity', 'melt', 'almost', 'reason', 'meteors', 'reach', 'earth', 'disintegrate', 'gro
und', 'unless', 'theyre', 'sophisticated', 'withstand', 'unbelievably', 'temperatures', 'perfecting', 'part', 'planetcolonizin
g', 'systemso', 'started', 'lot', 'people', 'heard', 'relatively', 'recently', 'may', 'instantly', 'appeared', 'wasnt', 'know',
'rocketsthe', 'ended', 'chief', 'engineer', 'designer', 'couldnt', 'hire', 'anyone', 'nobody', 'join', 'default', 'messed', 'fi
rst', 'three', 'failed', 'fortunately', 'fourth', 'money', 'worked', 'fate', 'liked', 'workedjust', 'today', 'ninth', 'annivers
ary', 'realize', 'told', 'earlier', 'emotional', 'rocket', 'trying', 'figure', 'smallest', 'useful', 'payload', 'thought', 'o
k', 'around', 'ton', 'decentsize', 'satellitebut', 'factor', 'times', 'reuse', 'primary', 'expensive', 'hopefully', 'soon', 'fa
iring', 'nosecone', 'front', 'somewhere', '70', '80', 'reusability', 'towards', 'end', 'launching', 'complex', 'program', 'soun
```

In [45]:
```
def SequenceSelection(dictionary, length, startindex = 0):
    lengthDict = len(dictionary)
    if length > lengthDict:
        return print("length is longer than dictionary length");
    else:
        d = dictionary #A dictionary comprises of two elements: values (v) and keys (k)
        items = [(v, k) for k, v in d.items()]
        items.sort()
        items.reverse() #By using the reverse method, it modifies the original items rather than creating  new ones.
        itemsOut = [(k, v) for v, k in items]

        highest = itemsOut[startindex:startindex + length]
        dd = dict(highest)
        wanted_keys = dd.keys()
        dictshow = dict((k, d[k]) for k in wanted_keys if k in d)

        return dictshow

        #We tried in multiple ways to return only words with more than 4 letters, in order to increase the relevance of our list,

        #lastDict = dictshow.keys(len(key)>4)
        #output = [''.join([word for word in sentence.split() if len(word) >4]) for sentence in dictshow]
```

In [46]:
```
dictshow = SequenceSelection(dictionary = dict2, length = 20, startindex = 0)
#SequenceSelection is the order in which instructions occur and are processed.
#For our project, we chose the first 20 most used words from each speech.

#long_words = [word for word in dictshow if len(word) > 4]
lastDictionary = {k:dictshow[k] for k in dictshow if len(k)>=4}

print(lastDictionary)
```

```
{'falcon': 22, 'thats': 15, 'really': 15, 'engine': 15, 'payload': 14, 'think': 13, 'landing': 12, 'land': 11, 'engines': 11,
'would': 10, 'tons': 10, 'propellant': 10, 'launch': 10, 'dragon': 10, 'want': 8, 'quite': 8, 'order': 8}
```

In [47]:
```
n = range(len(lastDictionary))
width1 = 20
height1 = 5
width_height_1 = (width1, height1)
plt.figure(figsize=width_height_1)

plt.bar(n, lastDictionary.values(), align ='center', color = 'crimson')
plt.xticks(n, lastDictionary.keys())
plt.title("Most frequent words used by Elon Musk")
plt.savefig("FrequentWords-Musk.png", transparent=True)
```



In [48]:
```
from os import path
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

In [49]:
```
! pip install wordcloud
```

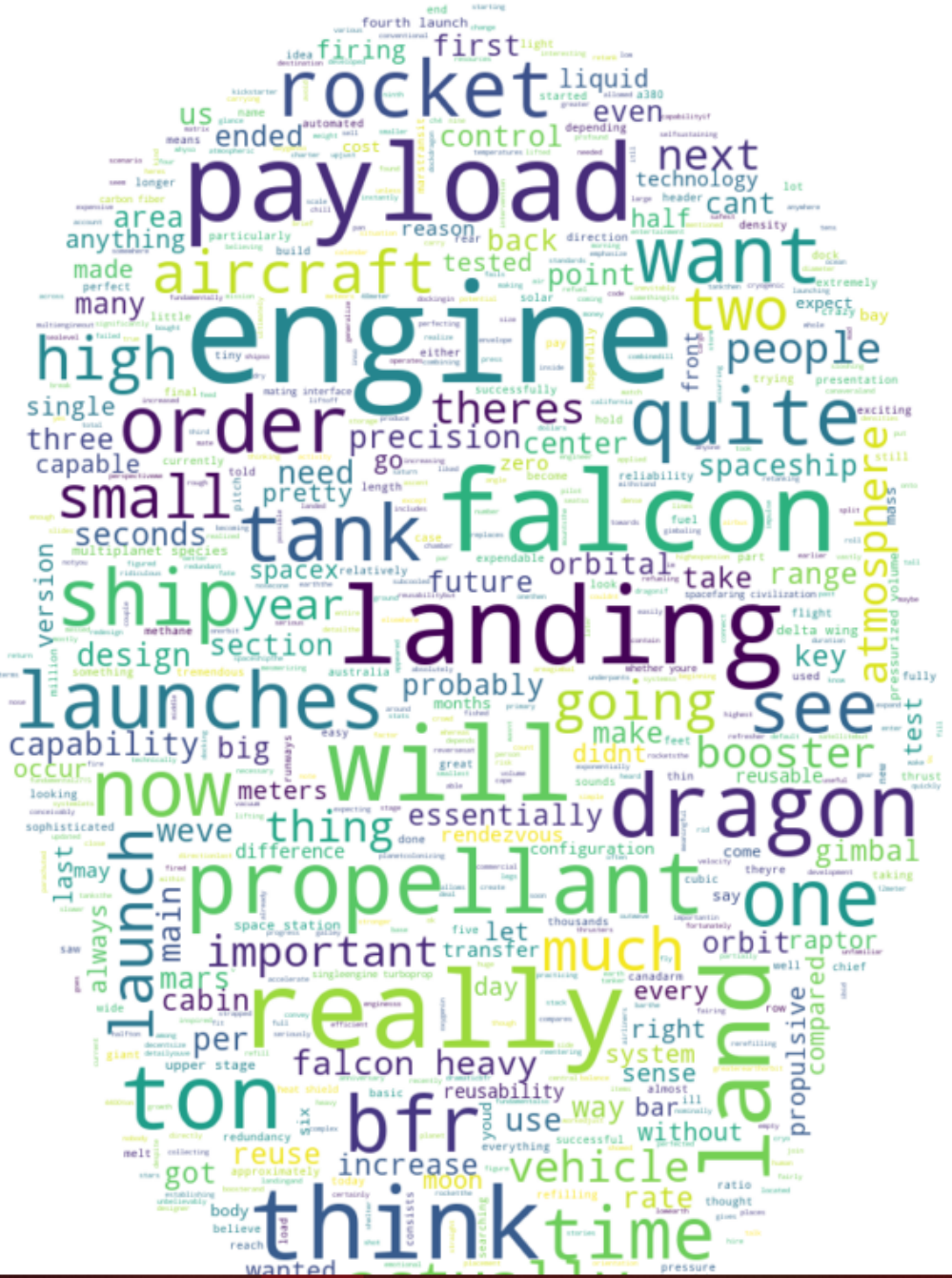
```
Requirement already satisfied: wordcloud in c:\users\c_sab\anaconda3\lib\site-packages (1.8.1)
Requirement already satisfied: matplotlib in c:\users\c_sab\anaconda3\lib\site-packages (from wordcloud) (3.4.3)
Requirement already satisfied: pillow in c:\users\c_sab\anaconda3\lib\site-packages (from wordcloud) (8.4.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\c_sab\anaconda3\lib\site-packages (from wordcloud) (1.20.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\c_sab\anaconda3\lib\site-packages (from matplotlib->wordcloud) (3.
0.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\c_sab\anaconda3\lib\site-packages (from matplotlib->wordcloud) (1.
3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\c_sab\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.10.0)
```

In [50]:
```
from wordcloud import WordCloud, STOPWORDS
```

In [82]:
```
root_path = os.getcwd()

#Using os.path.join helped us to combine more path names into a single path.
with open(os.path.join(root_path, 'OutputMusk.txt'), 'r', errors='ignore') as output_file:
    text = output_file.readlines()

Elon_face = np.array(Image.open(path.join(root_path, "Elon_face1.jpg")))

#We opted to eliminate some additional stopwords for our word cloud
stopwords = set(STOPWORDS)
stopwords.add("thats")

wc = WordCloud(max_words=700,
               stopwords=stopwords, mask=Elon_face, mode='RGBA', background_color= "white", max_font_size=100, width=Elon_face.sh
               height=Elon_face.shape[0])

wc.generate(text[0])
wc.to_file(path.join(root_path, "Musk.png"))

plt.figure()
plt.figure(figsize=width_height_1)
width1 = 15
height1 = 15
width_height_1 = (width1, height1)
title = "Wordcloud Elon Musk speech"
plt.title(title)
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.imshow(wc,cmap=plt.cm.gray, interpolation='bilinear')
```

Out[82]:
```
<matplotlib.image.AxesImage at 0x2154d856190>

<Figure size 432x288 with 0 Axes>
```

Wordcloud Elon Musk speech



In [52]:
```
! pip install textblob
```

```
Requirement already satisfied: textblob in c:\users\c_sab\anaconda3\lib\site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in c:\users\c_sab\anaconda3\lib\site-packages (from textblob) (3.6.5)
Requirement already satisfied: click in c:\users\c_sab\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (8.0.3)
Requirement already satisfied: joblib in c:\users\c_sab\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (1.1.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\c_sab\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (2021.
8.3)
Requirement already satisfied: tqdm in c:\users\c_sab\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (4.62.3)
Requirement already satisfied: colorama in c:\users\c_sab\anaconda3\lib\site-packages (from click->nltk>=3.1->textblob) (0.4.4)
```

In [80]:
```
from textblob import TextBlob

sentiment = TextBlob(cleantext)
print("Sentiment Score: ", sentiment.sentiment.polarity)

#The score indicates how negative or positive the overall text analyzed is. Values below a score of -0.05 are considered negative
```

```
Sentiment Score:  0.1204426898858716
```

In [ ]:

In [1]:
```
import pandas as pd
import bs4
import urllib
```

In [2]:
```
import matplotlib.pyplot as plt
import re
from nltk.corpus import stopwords
import os
```

In [3]:
```
f = open("Steve.txt", encoding="latin-1")
text=f.read()
```

In [4]:
```
text
```

Out[4]:
```
'ï»¿This is the day Iâ\x80\x99ve been looking forward to for two and a half years.\nEvery once in a while, a revolutionary pr
oduct comes along that changes everything and Apple has been â\x80\x93 well, first of all, oneâ\x80\x99s very fortunate if yo
u get to work on just one of these in your career. Apple has been very fortunate. Itâ\x80\x99s been able to introduce a few o
f these into the world.\n\n\n1984 â\x80\x93 we introduced the Macintosh. It didnâ\x80\x99t just change Apple. It changed the
whole computer industry.\n\n\nIn 2001, we introduced the first iPod. And it didnâ\x80\x99t just change the way we all listen
to music, it changed the entire music industry.\n\n\nWell, today weâ\x80\x99re introducing three revolutionary products of th
is class. The first one is a widescreen iPod with touch controls. The second is a revolutionary mobile phone. And the third i
s a breakthrough Internet communications device.\n\n\nSo, three things: a widescreen iPod with touch controls; a revolutionar
y mobile phone; and a breakthrough Internet communications device. An iPod, a phone, and an Internet communicator. An iPod, a
phoneâ\x80¦ are you getting it? These are not three separate devices, this is one device, and we are calling it iPhone. Toda
y, Apple is going to reinvent the phone, and here it is.\n\n\nNo, actually here it is, but weâ\x80\x99re going to leave it th
ere for now.\n\n\nSo, before we get into it, let me talk about a category of things. The most advanced phones are called smar
t phones, so they say. And they typically combine a phone plus some e-mail capability, plus they say itâ\x80\x99s the Interne
t. Itâ\x80\x99s sort of the baby Internet, into one device, and they all have these little plastic keyboards on them. And the
problem is that theyâ\x80\x99re not so smart and theyâ\x80\x99re not so easy to use, and so if you kind of make a Business Sc
hool 101 graph of the smart axis and the easy-to-use axis, phones, regular cell phones are right there, theyâ\x80\x99re not s
o smart, and theyâ\x80\x99re not so easy to use. But smart phones are definitely a little smarter, but they actually are hard
er to use. Theyâ\x80\x99re really complicated. Just for the basic stuff people have a hard time figuring out how to use them.
Well, we donâ\x80\x99t want to do either one of these things. What we want to do is make a leapfrog product that is way smart
```

In [5]:
```
expression = "[^a-zA-Z0-9 ]"
cleantextCAP = re.sub(expression, '', text)
cleantext = cleantextCAP.lower()
```

In [6]:
```
text_file = open("OutputSteve.txt", "w")
text_file.write(str(cleantext))
text_file.close()
```

In [7]:
```
dat = list(cleantext.split())
dict1 = {}
for i in range(len(dat)):
    print(i)
    word = dat[i]
    dict1[word] = dat.count(word)
```

```
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
```

In [10]:
```
keys = list(dict1)
filtered_words = [word for word in keys if word not in stopwords.words('english')]
dict2  = dict((k, dict1[k]) for k in filtered_words if k in filtered_words)

print(filtered_words)
```

```
['day', 'ive', 'looking', 'forward', 'two', 'half', 'yearsevery', 'revolutionary', 'product', 'comes', 'along', 'changes', 'eve
rything', 'apple', 'well', 'first', 'ones', 'fortunate', 'get', 'work', 'one', 'career', 'able', 'introduce', 'world1984', 'int
roduced', 'macintosh', 'didnt', 'change', 'changed', 'whole', 'computer', 'industryin', '2001', 'ipod', 'way', 'listen', 'musi
c', 'entire', 'industrywell', 'today', 'introducing', 'three', 'products', 'class', 'widescreen', 'touch', 'controls', 'secon
d', 'mobile', 'phone', 'third', 'breakthrough', 'internet', 'communications', 'deviceso', 'things', 'device', 'communicator',
'getting', 'separate', 'devices', 'calling', 'iphone', 'going', 'reinvent', 'isno', 'actually', 'leave', 'nowso', 'let', 'tal
k', 'category', 'advanced', 'phones', 'called', 'smart', 'say', 'typically', 'combine', 'plus', 'email', 'capability', 'sort',
'baby', 'little', 'plastic', 'keyboards', 'problem', 'theyre', 'easy', 'use', 'kind', 'make', 'business', 'school', '101', 'gra
ph', 'axis', 'easytouse', 'regular', 'cell', 'right', 'definitely', 'smarter', 'harder', 'really', 'complicated', 'basic', 'stu
ff', 'people', 'hard', 'time', 'figuring', 'dont', 'want', 'either', 'leapfrog', 'ever', 'supereasy', 'okso', 'start', 'user',
'interface', 'result', 'years', 'research', 'development', 'course', 'interplay', 'hardware', 'software', 'need', 'heres', 'fou
r', 'motorola', 'q', 'blackberry', 'palm', 'treo', 'nokia', 'e62', 'usual', 'suspects', 'whats', 'wrong', 'interfaces', 'botto
m', '40', 'whether', 'control', 'buttons', 'fixed', 'every', 'application', 'wants', 'slightly', 'different', 'optimized', 'se
```

```
'saves', 'accelerometer', 'tell', 'portrait', 'landscape', 'pretty', 'cool', 'minute', 'inso', 'turn', 'size', 'fits', 'beautif
ully', 'youve', 'even', 'faster', 'album', 'art', 'builtin', 'cover', 'flow', 'rather', 'youalrighty', 'special', 'iphones', 't
heyve', 'board', 'digital', 'cord', 'goes', 'projectors', 'images', 'im', 'finger', 'seconds', 'picture', 'within', 'sleepwak
e', 'unlock', 'slide', 'wanted', 'couldnt', 'accident', 'pocket', 'lower', 'corner', 'icon', 'playlists', 'songs', 'artists',
'scroll', 'lists', 'isnt', 'rubber', 'banding', 'edgeand', 'pick', 'somebody', 'beatles', 'tap', 'albums', 'play', 'sgt', 'pepp
ers', 'hit', 'help', 'friends', 'artwork', 'upper', 'righthand', 'flip', 'lovely', 'rita', 'simple', 'stars', 'setting', 'arrow
s', 'star', 'yeah']
```

In [11]:

```
def SequenceSelection(dictionary, length, startindex = 0):
    lengthDict = len(dictionary)
    if length > lengthDict:
        return print("length is longer than dictionary length");
    else:
        d = dictionary
        items = [(v, k) for k, v in d.items()]
        items.sort()
        items.reverse()
        itemsOut = [(k, v) for v, k in items]

        highest = itemsOut[startindex:startindex + length]
        dd = dict(highest)
        wanted_keys = dd.keys()
        dictshow = dict((k, d[k]) for k in wanted_keys if k in d)

        return dictshow;
```

In [12]:

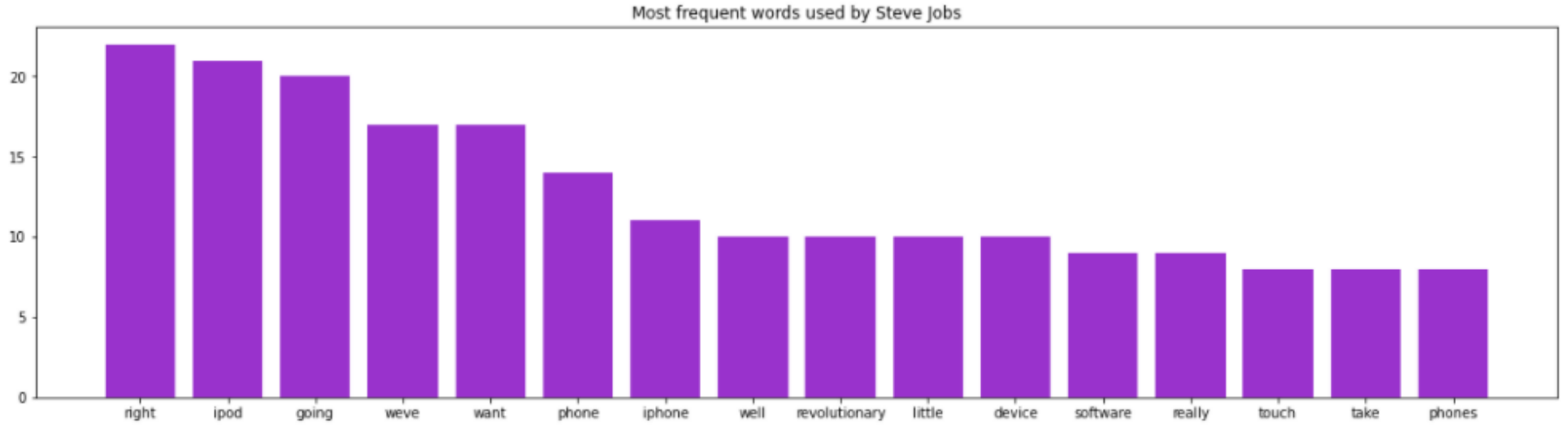
```
dictshow = SequenceSelection(dictionary = dict2, length = 20, startindex = 0)

lastDictionary = {k:dictshow[k] for k in dictshow if len(k)>=4}

print(lastDictionary)
```

```
{'right': 22, 'ipod': 21, 'going': 20, 'weve': 17, 'want': 17, 'phone': 14, 'iphone': 11, 'well': 10, 'revolutionary': 10, 'lit
tle': 10, 'device': 10, 'software': 9, 'really': 9, 'touch': 8, 'take': 8, 'phones': 8}
```

In [13]:

```
n = range(len(lastDictionary))
width1 = 20
height1 = 5
width_height_1 = (width1, height1)
plt.figure(figsize=width_height_1)

plt.bar(n, lastDictionary.values(), align ='center', color = 'darkorchid')
plt.xticks(n, lastDictionary.keys())
plt.title("Most frequent words used by Steve Jobs")
plt.savefig("FrequentWords-Jobs.png", transparent=True)
```

In [13]:
```
n = range(len(lastDictionary))
width1 = 20
height1 = 5
width_height_1 = (width1, height1)
plt.figure(figsize=width_height_1)

plt.bar(n, lastDictionary.values(), align ='center', color = 'darkorchid')
plt.xticks(n, lastDictionary.keys())
plt.title("Most frequent words used by Steve Jobs")
plt.savefig("FrequentWords-Jobs.png", transparent=True)
```



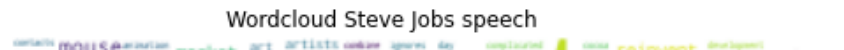
In [14]:
```
from os import path
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

In [14]:
```
from os import path
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

In [15]:
```
from wordcloud import WordCloud, STOPWORDS
```

In [40]:
```
root_path = os.getcwd()

with open(os.path.join(root_path, 'OutputSteve.txt'), 'r', errors='ignore') as output_file:
    text = output_file.readlines()

Steve_face = np.array(Image.open(path.join(root_path, "Steve_face.jpg")))

stopwords = set(STOPWORDS)
stopwords.add("weve")
stopwords.add("got")

wc = WordCloud(max_words=700,
               stopwords=stopwords, mask=Steve_face, mode='RGBA', background_color= "white", max_font_size=100, width=Steve_face.
               height=Steve_face.shape[0])

wc.generate(text[0])

wc.to_file(path.join(root_path, "Jobs.png"))

plt.figure()
plt.figure(figsize=width_height_1)
width1 = 15
height1 = 15
width_height_1 = (width1, height1)
title = "Wordcloud Steve Jobs speech"
plt.title(title)
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.imshow(wc,cmap=plt.cm.gray, interpolation='bilinear')
```

Out[40]:
```
<matplotlib.image.AxesImage at 0x20b43286910>

<Figure size 432x288 with 0 Axes>
```



Wordcloud Steve Jobs speech

In [39]:
```
from textblob import TextBlob

sentiment = TextBlob(cleantext)
print("Sentiment Score: ", sentiment.sentiment.polarity)
```

```
Sentiment Score:  0.20558020683020686
```

In [1]:
```
import pandas as pd
import bs4
import urllib
```

In [2]:
```
import matplotlib.pyplot as plt
import re
from nltk.corpus import stopwords
import os
```

In [3]:
```
f = open("Mark transcript.txt", 'r')
text=f.read()
```

In [4]:
```
text
```

Out[4]:
```
'ï»¿Desktop to web to phones, from text to photos to video. But this isnâ€™t the end of the line. The next platform and medium
will be even more immersive, an embodied internet where youâ€™re in the experience, not just looking at it, and we call this th
e metaverse. And youâ€™re going to be able to do almost anything you can imagine, get together with friends and family, work, l
earn, play, shop, create as well as entirely new categories that donâ€™t really fit how we think about computers or phones toda
y. Now, since weâ€™re doing this remotely today, I figured letâ€™s make this special.\n \nSo weâ€™ve put together something tha
t I think is really going to give you a feeling for what this future could be like. We believe the metaverse will be the succes
sor to the mobile internet. Weâ€™ll be able to feel present like weâ€™re right there with people no matter how far apart we act
ually are. Weâ€™ll be able to express ourselves in new, joyful, completely immersive ways and thatâ€™s to unlock a lot of amazi
ng new experiences. When I send my parents a video of my kids, theyâ€™re going to feel like theyâ€™re right in the moment with
us not peering through a little window.\n \nWhen you play a game with your friends, youâ€™ll feel like youâ€™re right there tog
ether in a different world, not just on your computer by yourself. And when youâ€™re in a meeting in the metaverse, itâ€™ll fee
l like youâ€™re right in the room together, making eye contact, having a shared sense of space and not just looking at a grid o
f faces on a screen. Thatâ€™s what we mean by an embodied internet. Instead of looking at a screen, youâ€™re going to be in the
se experiences. Everything we do online today connecting socially, entertainment, games, work is going to be more natural and v
ivid.\n \nThis isnâ€™t about spending more time on screens. Itâ€™s about making the time that we already spend better. Screens
just canâ€™t convey the full range of human expression and connection. They canâ€™t deliver that deep feeling of presence, but
the next version of the internet can. Thatâ€™s what we should be working towards. Technology thatâ€™s built around people and h
ow we actually experience the world and interact with each other. Thatâ€™s what the metaverse is all about. Now, the best way t
o understand the metaverse is to experience it yourself.\n \nBut itâ€™s a little tough because it doesnâ€™t â€¦ Emerge as we sp
eak, weâ€™re starting to get a sense of how it could all come together and what it could feel like. So today weâ€™re going to d
o something a little bit different. Rather than just focusing on this yearâ€™s products, like a normal keynote, weâ€™re going t
o talk about the future. So letâ€™s start by exploring what different kinds of metaverse experiences could feel like, starting
with the most important experience of all, connecting with people.\n \nImagine you put on your glasses or headset, and youâ€™re
instantly in your home space. It has parts of your physical home recreated virtually, it has things that are only possible virt
ually, and it has an incredibly inspiring view of whatever you find most beautiful.\n \n \nAll right. So thatâ€™s a glimpse of
a few ways that weâ€™re going to be able to get together and socialize in the metaverse. Itâ€™s a ways off, but you can start t
o see some of the fundamental building blocks take shape. First, the feeling of presence. This is the defining quality of the m
etaverse. Youâ€™re going to really feel like youâ€™re there with other people. Youâ€™ll see their face expressions. Youâ€™ll se
e their body language. Maybe figure out if theyâ€™re actually holding a winning hand. All the subtle ways that we communicate t
hat todayâ€™s technology canâ€™t quite deliver.\n \nNext, there are avatars, and thatâ€™s how weâ€™re going to represent oursel
```

In [5]:
```
expression = "[^a-zA-Z0-9 ]"
cleantextCAP = re.sub(expression, '', text)
cleantext = cleantextCAP.lower()
```

In [6]:
```
text_file = open("OutputMark.txt", "w")
text_file.write(str(cleantext))
text_file.close()
```

In [7]:
```
dat = list(cleantext.split())
dict1 = {}
for i in range(len(dat)):
    print(i)
    word = dat[i]
    dict1[word] = dat.count(word)
```

```
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
```

In [8]:
```
keys = list(dict1)
filtered_words = [word for word in keys if word not in stopwords.words('english')]
dict2  = dict((k, dict1[k]) for k in filtered_words if k in filtered_words)

print(filtered_words)
```

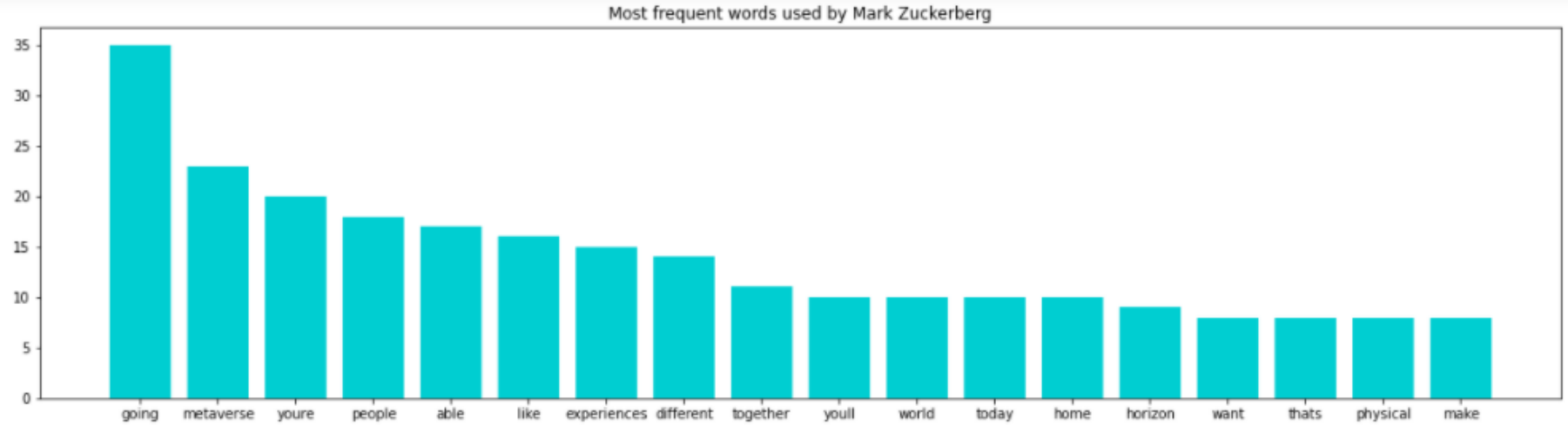
```
['desktop', 'web', 'phones', 'text', 'photos', 'video', 'isnt', 'end', 'line', 'next', 'platform', 'medium', 'even', 'immersiv
e', 'embodied', 'internet', 'youre', 'experience', 'looking', 'call', 'metaverse', 'going', 'able', 'almost', 'anything', 'imag
ine', 'get', 'together', 'friends', 'family', 'work', 'learn', 'play', 'shop', 'create', 'well', 'entirely', 'new', 'categorie
s', 'dont', 'really', 'fit', 'think', 'computers', 'today', 'since', 'remotely', 'figured', 'lets', 'make', 'special', 'weve',
'put', 'something', 'give', 'feeling', 'future', 'could', 'like', 'believe', 'successor', 'mobile', 'feel', 'present', 'right',
'people', 'matter', 'far', 'apart', 'actually', 'express', 'joyful', 'completely', 'ways', 'thats', 'unlock', 'lot', 'amazing',
'experiences', 'send', 'parents', 'kids', 'theyre', 'moment', 'us', 'peering', 'little', 'window', 'game', 'youll', 'differen
t', 'world', 'computer', 'meeting', 'itll', 'room', 'making', 'eye', 'contact', 'shared', 'sense', 'space', 'grid', 'faces', 's
creen', 'mean', 'instead', 'everything', 'online', 'connecting', 'socially', 'entertainment', 'games', 'natural', 'vivid', 'spe
nding', 'time', 'screens', 'already', 'spend', 'better', 'cant', 'convey', 'full', 'range', 'human', 'expression', 'connectio
n', 'deliver', 'deep', 'presence', 'version', 'working', 'towards', 'technology', 'built', 'around', 'interact', 'best', 'way',
'understand', 'tough', 'doesnt', 'emerge', 'speak', 'starting', 'come', 'bit', 'rather', 'focusing', 'years', 'products', 'norm
al', 'keynote', 'talk', 'start', 'exploring', 'kinds', 'important', 'glasses', 'headset', 'instantly', 'home', 'parts', 'physic
al', 'recreated', 'virtually', 'things', 'possible', 'incredibly', 'inspiring', 'view', 'whatever', 'find', 'beautiful', 'glimp
se', 'socialize', 'see', 'fundamental', 'building', 'blocks', 'take', 'shape', 'first', 'defining', 'quality', 'face', 'express
ions', 'body', 'language', 'maybe', 'figure', 'holding', 'winning', 'hand', 'subtle', 'communicate', 'todays', 'quite', 'avatar
s', 'represent', 'common', 'profile', 'pictures', 'static', 'image', 'living', '3d', 'representations', 'gestures', 'interactio
ns', 'much', 'richer', 'probably', 'photo', 'realistic', 'avatar', 'stylized', 'one', 'hanging', 'fantasy', 'gaming', 'wardrob
e', 'virtual', 'clothes', 'occasions', 'designed', 'creators', 'apps', 'importantly', 'bring', 'digital', 'items', 'across', 'b
eyond', 'design', 'look', 'want', 'videos', 'store', 'goods', 'invite', 'hang', 'also', 'office', 'personal', 'teleport', 'anyw
here', 'speaking', 'teleporting', 'spaces', 'rooms', 'ones', 'saw', 'whole', 'worlds', 'whenever', 'clicking', 'link', 'open',
'standard', 'order', 'potential', 'needs', 'interoperability', 'goes', 'taking', 'api', 'support', 'know', 'buy', 'useful', 'co
ntexts', 'locked', 'require', 'technical', 'projects', 'crypto', 'nfts', 'community', 'ecosystem', 'norm', 'setting', 'forms',
'governance', 'focus', 'privacy', 'safety', 'need', 'day', 'decide', 'block', 'someone', 'appearing', 'break', 'private', 'bubb
le', 'alone', 'type', 'media', 'represented', 'digitally', 'art', 'music', 'movies', 'books', 'name', 'lots', 'holograms', 'won
t', 'tv', '1', 'hologram', 'high', 'school', 'kid', 'halfway', 'project', 'augmented', 'reality', 'move', 'devices', 'sometime
s', 'using', 'fully', 'immersed', 'phone', 'quickly', 'jump', 'existing', 'platforms', 'interacting', 'typing', 'tapping', 'ges
ture', 'hands', 'say', 'words', 'happen', 'thinking', 'focal', 'point', 'attention', 'anymore', 'getting', 'basic', 'concepts',
'may', 'sound', 'science', 'fiction', 'technologies', 'coming', 'five', '10', 'mainstream', 'creating', 'inhabiting', 'detaile
d', 'convincing', 'daily', 'basis', 'though', 'still', 'long', 'foundational', 'horizon', 'social', 'part', 'early', 'vision',
'thing', 'quest', 'bunch', 'options', 'choose', 'anyone', 'called', 'missing', 'soon', 'introducing', 'join', 'watch', 'build',
'everyone', 'seeing', 'interesting', 'throwing', 'surprise', 'parties', 'vr', 'started', 'rolling', 'beta', 'last', 'year', 'ad
ding', 'every', 'launched', 'workrooms', 'earlier', 'collaboration', 'easier', 'layers', 'bringing', 'messenger', 'calls', 'exp
lore', 'somewhere', 'tools', 'pretty', 'set', 'enable', 'letting', 'add', 'artists', 'connect', 'audiences', 'theres', 'piece
s', 'spark', 'ar', 'use', 'place', 'objects', 'let', 'simple', 'visual', 'effects', 'creator', 'capabilities', 'respond', 'reac
t', 'realistically', 'including', 'depth', 'occlusion', 'ability', 'locations', 'cohesive', 'storytelling', 'guided', 'tours',
'scavenger', 'hunts', 'marketplace', 'sell', 'share', 'hope', 'commerce', 'help', 'grow', 'overall', 'economy', 'developers',
'real', 'sure', 'sustain', 'hundreds', 'thousands', 'life', 'critical', 'good', 'ask', 'thought', 'would', 'spiderman', 'movi
e']
```

In [87]:
```
def SequenceSelection(dictionary, length, startindex = 0):
    lengthDict = len(dictionary)
    if length > lengthDict:
        return print("length is longer than dictionary length");
    else:
        d = dictionary
        items = [(v, k) for k, v in d.items()]
        items.sort()
        items.reverse()
        itemsOut = [(k, v) for v, k in items]

        highest = itemsOut[startindex:startindex + length]
        dd = dict(highest)
        wanted_keys = dd.keys()
        dictshow = dict((k, d[k]) for k in wanted_keys if k in d)

        return dictshow;
```

In [88]:
```
dictshow = SequenceSelection(dictionary = dict2, length = 20, startindex = 0)

lastDictionary = {k:dictshow[k] for k in dictshow if len(k)>=4}

print(lastDictionary)
```

```
{'going': 35, 'metaverse': 23, 'youre': 20, 'people': 18, 'able': 17, 'like': 16, 'experiences': 15, 'different': 14, 'togethe
r': 11, 'youll': 10, 'world': 10, 'today': 10, 'home': 10, 'horizon': 9, 'want': 8, 'thats': 8, 'physical': 8, 'make': 8}
```

In [89]:
```
n = range(len(lastDictionary))
width1 = 20
height1 = 5
width_height_1 = (width1, height1)
plt.figure(figsize=width_height_1)

plt.bar(n, lastDictionary.values(), align ='center', color = 'darkturquoise')
plt.xticks(n, lastDictionary.keys())
plt.title("Most frequent words used by Mark Zuckerberg")
plt.savefig("FrequentWords-Zuckerberg.png", transparent=True)
```



In [90]:
```
from os import path
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

In [91]:
```
from wordcloud import WordCloud, STOPWORDS
```

In [94]:
```
root_path = os.getcwd()

with open(os.path.join(root_path, 'OutputMark.txt'), 'r', errors='ignore') as output_file:
    text = output_file.readlines()

Mark_face = np.array(Image.open(path.join(root_path, "Mark_face6.jpg")))

stopwords = set(STOPWORDS)
stopwords.add("going")
stopwords.add("youre")
stopwords.add("youll")

wc = WordCloud(max_words=700,
               stopwords=stopwords, mask=Mark_face, mode='RGBA', background_color= "white", max_font_size=100, width=Mark_face.sh
               height=Mark_face.shape[0])

wc.generate(text[0])
```

```
wc.generate(text[0])

wc.to_file(path.join(root_path, "Zuckerberg.png"))

plt.figure()
plt.figure(figsize=width_height_1)
width1 = 15
height1 = 15
width_height_1 = (width1, height1)
title = "Wordcloud Mark Zuckerberg speech"
plt.title(title)
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.imshow(wc,cmap=plt.cm.gray, interpolation='bilinear')
```

Out[94]: <matplotlib.image.AxesImage at 0x17905f94880>

<Figure size 432x288 with 0 Axes>

In [93]:
```
from textblob import TextBlob

sentiment = TextBlob(cleantext)
print("Sentiment Score: ", sentiment.sentiment.polarity)
```

```
Sentiment Score:  0.1634718044040078
```

In [192]:
```
import pandas as pd

from datetime import datetime

import matplotlib.pyplot as plt
import re
import seaborn as sns
import itertools
import collections

import networkx as nx
from nltk.corpus import stopwords
import os

! pip install vaderSentiment

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

```
Requirement already satisfied: vaderSentiment in c:\users\c_sab\anaconda3\lib\site-packages (3.3.2)
Requirement already satisfied: requests in c:\users\c_sab\anaconda3\lib\site-packages (from vaderSentiment) (2.26.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\c_sab\anaconda3\lib\site-packages (from requests->vaderSentiment)
(2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\c_sab\anaconda3\lib\site-packages (from requests->vaderSentiment) (3.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\c_sab\anaconda3\lib\site-packages (from requests->vaderSentime
nt) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\c_sab\anaconda3\lib\site-packages (from requests->vaderSen
timent) (2.0.4)
```

In [193]:
```
df = pd.read_excel('Speeches combined1.xlsx')
df=df.dropna()
```

In [194]:
```
df['Date'] =  pd.to_datetime(df['Date']).dt.strftime('%Y-%m-%d')

df= df[df['Content'].notna()]
print(df)
```

```
         Date          Speaker                                          Title  \
0  2017-10-25        Elon Musk                    Making Life Multiplanetary
1  2007-09-01       Steve Jobs                        Introducing the Iphone
2  2021-10-28  Mark Zuckerberg  The Metaverse and how we'll build it together

                                        Content
0  I'm going to talk more about what it takes to ...
1  This is the day I've been looking forward to f...
2  Desktop to web to phones, from text to photos ...
```

jupyter Speeches Analysis Last Checkpoint: 9 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

In [195]:
```
import matplotlib.pyplot as plt
import re
from nltk.corpus import stopwords

stop_words = stopwords.words("english")

from nltk import bigrams


df['clean_text'] = df['Content'].apply(lambda x: ' '.join([item for item in x.split()
                                                           if item not in stop_words]))
df['clean_text'] = df['clean_text'].str.replace('\d+', '')
```

```
C:\Users\c_sab\AppData\Local\Temp/ipykernel_12508/1880088210.py:12: FutureWarning: The default value of regex will change from
True to False in a future version.
  df['clean_text'] = df['clean_text'].str.replace('\d+', '')
```

In [196]:
```
print(df['clean_text'])
```

```
0    I'm going talk takes become multi-planet speci...
1    This day I've looking forward two half years. ...
2    Desktop web phones, text photos video. But isn...
Name: clean_text, dtype: object
```

In [197]:
```
def remove_url(txt):
    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt).split())

df_further_cleaning = [remove_url(text) for text in df['clean_text']]
df_further_cleaning[:1]

lower_case = [word.lower() for word in df['clean_text']]
sentences = df['clean_text']

df_further_cleaning[0].split()

words_in_df = [text.lower().split() for text in df_no_urls]
words_in_df[:2]
```

Out[197]:
```
[['im',
  'going',
  'talk',
  'takes',
  'become',
  'multiplanet',
  'species',
  'and'
```

In [198]:
```
#We created a list with all the words used in the three speeches, combined, and we created a counter in order to find out which w
all_words_further_cleaning = list(itertools.chain(*words_in_df))

counts_further_cleaning = collections.Counter(all_words_further_cleaning)

counts_further_cleaning.most_common(20)
```

Out[198]:
```
[('and', 62),
 ('going', 61),
 ('so', 56),
 ('its', 47),
 ('i', 43),
 ('were', 41),
 ('got', 36),
 ('want', 33),
 ('it', 32),
 ('get', 32),
 ('right', 31),
 ('really', 31),
 ('the', 29),
 ('thats', 29),
 ('one', 27),
 ('like', 27),
 ('people', 27),
 ('now', 27),
 ('weve', 25),
 ('metaverse', 23)]
```

In [199]:
```
stop_words = stopwords.words('english')
stop_words.extend(["also","now","yet","thing","things","thats", "get", "good","the","im","weve","would","its","going","talk","tak

#We wanted to eliminate the stop words from all the lists of words.
df_other_stop_words = [[word for word in df_words if not word in stop_words]
              for df_words in words_in_df]
df_other_stop_words[0]
```

Out[199]:
```
['become',
 'multiplanet',
 'species',
 'brief',
 'refresher',
 'important',
 'think',
```

In [200]:
```
#After we obtained the list, we wanted to count its content, in order to have a better view of the most used words.
all_words_other_stop_words_nc = list(itertools.chain(*df_other_stop_words))

counts_other_stop_words_nc = collections.Counter(all_words_other_stop_words_nc)

counts_other_stop_words_nc.most_common(20)
```

Out[200]:
```
[('right', 31),
 ('really', 31),
 ('one', 27),
 ('people', 27),
 ('see', 19),
 ('able', 19),
 ('think', 17),
 ('today', 16),
 ('home', 16),
 ('little', 15),
 ('engine', 15),
 ('use', 15),
 ('phone', 15),
 ('different', 15),
 ('experiences', 15),
 ('make', 14),
 ('first', 14),
 ('payload', 14),
 ('new', 13),
 ('landing', 13)]
```

In [201]:
```
#Next, we created a table with the list of words.
clean_df_other_stop_words = pd.DataFrame(counts_other_stop_words_nc.most_common(30),
                                         columns=['words', 'repetition'])
clean_df_other_stop_words
```

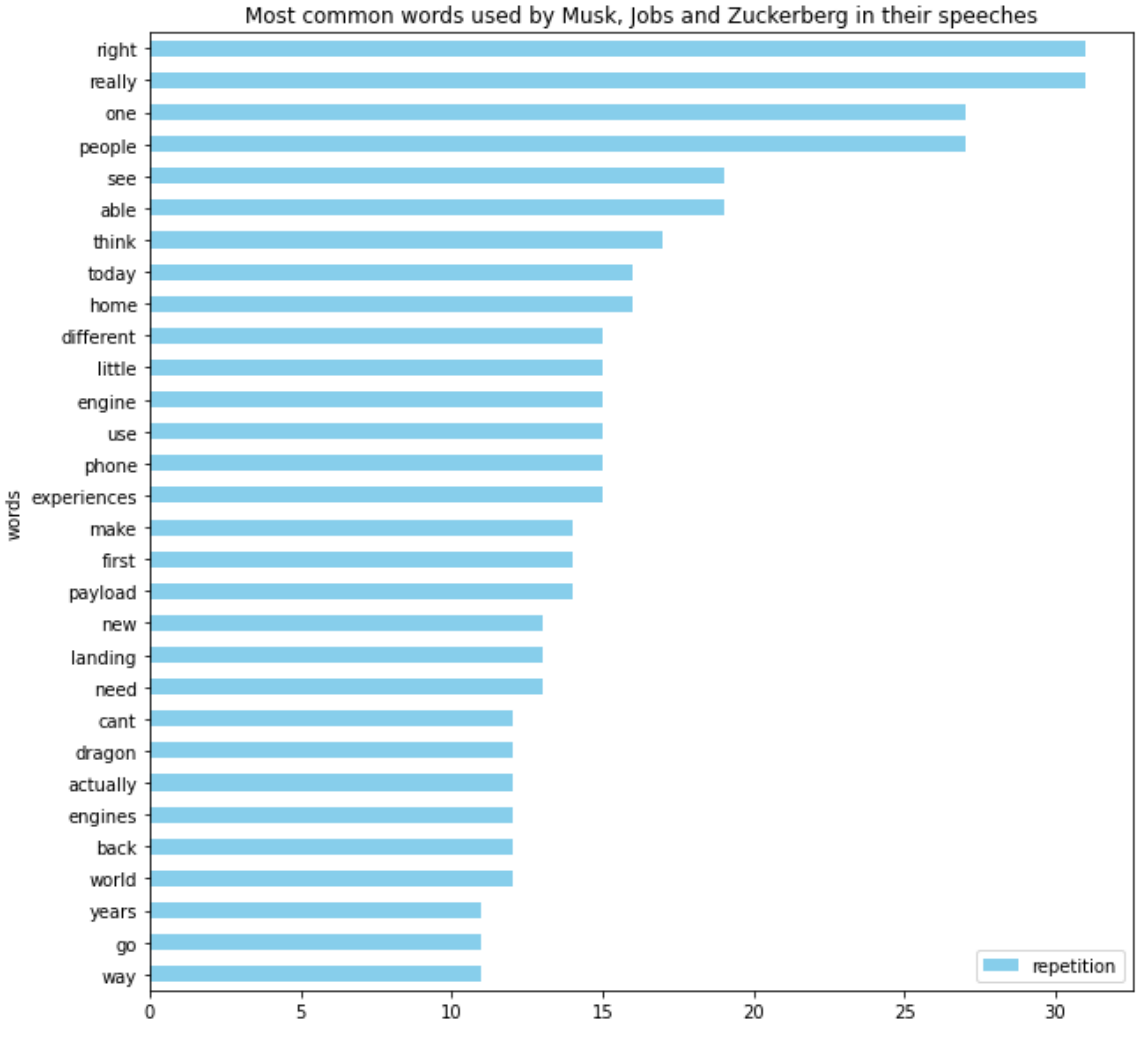
Out[201]:

	words	repetition
0	right	31
1	really	31
2	one	27
3	people	27
4	see	19
5	able	19
6	think	17

Out[201]:

	words	repetition
0	right	31
1	really	31
2	one	27
3	people	27
4	see	19
5	able	19
6	think	17
7	today	16
8	home	16
9	little	15
10	engine	15
11	use	15
12	phone	15
13	different	15
14	experiences	15
15	make	14
16	first	14
17	payload	14
18	new	13
19	landing	13
20	need	13
21	cant	12
22	dragon	12
23	actually	12
24	engines	12
25	back	12
26	world	12
27	go	11
28	years	11

In [276]:
```
fig, ax = plt.subplots(figsize=(10, 10))

#We created a plot with horizontal lines
clean_df_other_stop_words.sort_values(by ='repetition').plot.barh(x='words',
                      y ='repetition',
                      ax=ax,
                      color="skyblue")

ax.set_title("Most common words used by Musk, Jobs and Zuckerberg in their speeches")

plt.show()
```



jupyter Speeches Analysis Last Checkpoint: 9 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [220]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

df = pd.read_excel('Speeches combined1.xlsx')
df=df.dropna()

df['year'] = pd.DatetimeIndex(df['Date']).year
df['Date'] =  pd.to_datetime(df['Date']).dt.strftime('%Y-%m-%d')

df= df[df['Content'].notna()]
print(df)

def sentimentScore(text):
    analyzer = SentimentIntensityAnalyzer()
    results = []
    for sentence in text:
        vs = analyzer.polarity_scores(sentence)
        results.append(vs)
    return results
```

```
         Date          Speaker                                          Title  \
0  2017-10-25        Elon Musk                       Making Life Multiplanetary
1  2007-09-01       Steve Jobs                           Introducing the Iphone
2  2021-10-28  Mark Zuckerberg  The Metaverse and how we'll build it together

                                      Content  year
0  I'm going to talk more about what it takes to ...  2017
1  This is the day I've been looking forward to f...  2007
2  Desktop to web to phones, from text to photos ...  2021
```

In [221]:
```
df_results = pd.DataFrame(sentimentScore(df['Content']))
```

In [222]:
```
df=pd.merge(df, df_results, left_index=True, right_index=True)
df.set_index('Date', inplace=True)
```

In [223]:
```
df
```

Out[223]:

Date	Speaker	Title	Content	year	neg	neu	pos	compound
2017-10-25	Elon Musk	Making Life Multiplanetary	I'm going to talk more about what it takes to ...	2017	0.026	0.889	0.085	0.9995
2007-09-01	Steve Jobs	Introducing the Iphone	This is the day I've been looking forward to f...	2007	0.022	0.856	0.121	0.9998
2021-10-28	Mark Zuckerberg	The Metaverse and how we'll build it together	Desktop to web to phones, from text to photos ...	2021	0.007	0.856	0.136	0.9998

In [223]:
```
df
```

Out[223]:

Date	Speaker	Title	Content	year	neg	neu	pos	compound
2017-10-25	Elon Musk	Making Life Multiplanetary	I'm going to talk more about what it takes to ...	2017	0.026	0.889	0.085	0.9995
2007-09-01	Steve Jobs	Introducing the Iphone	This is the day I've been looking forward to f...	2007	0.022	0.856	0.121	0.9998
2021-10-28	Mark Zuckerberg	The Metaverse and how we'll build it together	Desktop to web to phones, from text to photos ...	2021	0.007	0.856	0.136	0.9998

In [256]:
```
df['polarity']=(df.pos-df.neg)/(df.pos+df.neg)
df
#"Neg","Neu","Pos" sum up to 1 and the scores show the proportion of text falling in the category. "Compound" ranges from -1 (the
```

Out[256]:

Date	Speaker	Title	Content	year	neg	neu	pos	compound	polarity
2017-10-25	Elon Musk	Making Life Multiplanetary	I'm going to talk more about what it takes to ...	2017	0.026	0.889	0.085	0.9995	0.531532
2007-09-01	Steve Jobs	Introducing the Iphone	This is the day I've been looking forward to f...	2007	0.022	0.856	0.121	0.9998	0.692308
2021-10-28	Mark Zuckerberg	The Metaverse and how we'll build it together	Desktop to web to phones, from text to photos ...	2021	0.007	0.856	0.136	0.9998	0.902098

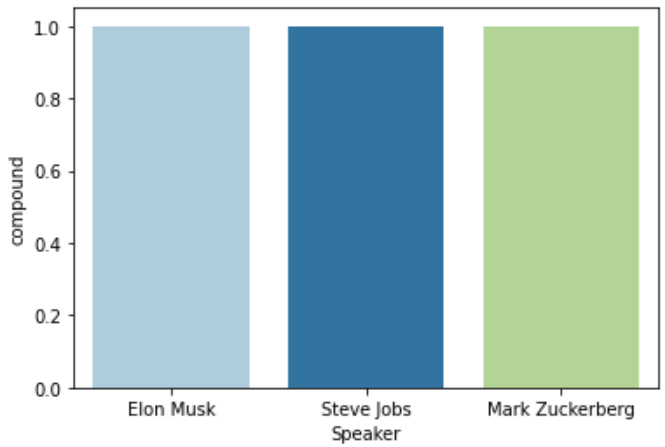
In [257]:
```
df.info
```

Out[257]:
```
<bound method DataFrame.info of                    Speaker                                         Title  \
Date
2017-10-25        Elon Musk                    Making Life Multiplanetary
2007-09-01       Steve Jobs                        Introducing the Iphone
2021-10-28  Mark Zuckerberg  The Metaverse and how we'll build it together

                                                Content  year    neg  \
Date
2017-10-25  I'm going to talk more about what it takes to ...  2017  0.026
2007-09-01  This is the day I've been looking forward to f...  2007  0.022
2021-10-28  Desktop to web to phones, from text to photos ...  2021  0.007
```

```
             neu    pos  compound  polarity
Date
2017-10-25  0.889  0.085    0.9995  0.531532
2007-09-01  0.856  0.121    0.9998  0.692308
2021-10-28  0.856  0.136    0.9998  0.902098  >
```

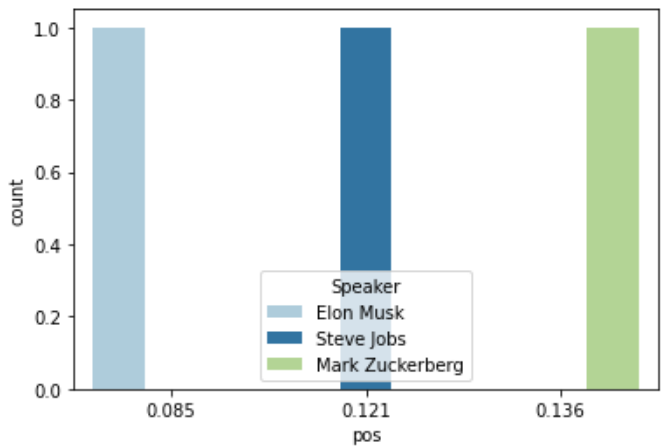
In [280]:
```
import seaborn as sns
sns.barplot(x='Speaker',y='compound',data=df, palette='Paired')
```

Out[280]: <AxesSubplot:xlabel='Speaker', ylabel='compound'>



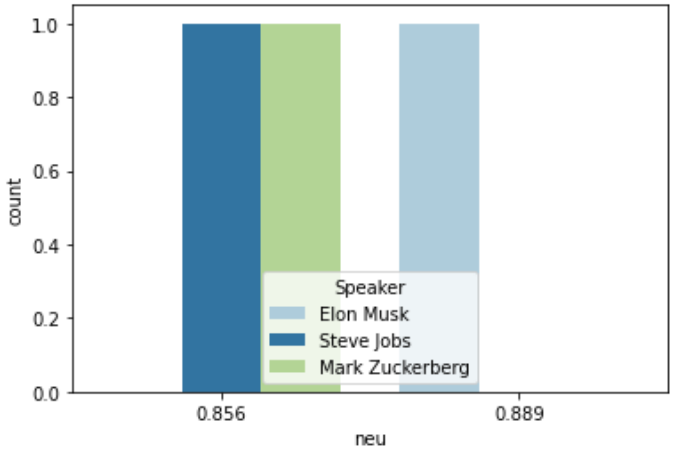
In [272]:
```
sns.countplot(x='pos', hue='Speaker',data=df, palette='Paired')
```

Out[272]: <AxesSubplot:xlabel='pos', ylabel='count'>



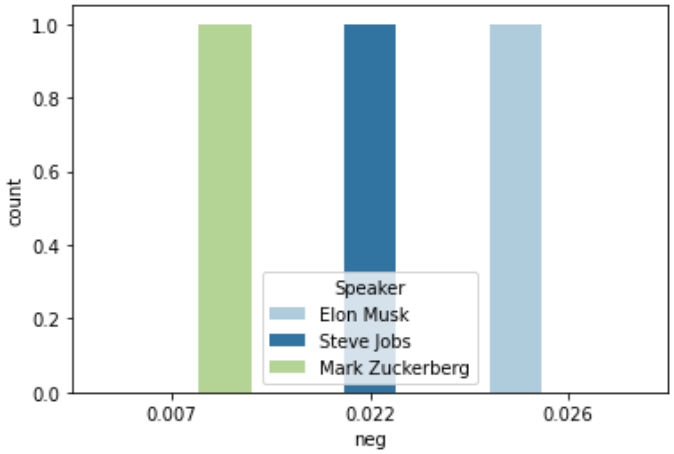
In [273]:
```
sns.countplot(x='neu', hue='Speaker',data=df, palette='Paired')
```

Out[273]: <AxesSubplot:xlabel='neu', ylabel='count'>



In [274]:
```
sns.countplot(x='neg', hue='Speaker',data=df, palette='Paired')
```

Out[274]: <AxesSubplot:xlabel='neg', ylabel='count'>



In [ ]:

In [ ]: