Public Sentiment of Commercial Versus Government Space Entities

Kimberly Gahn

Western Governors University

A. Research Question

As Neil Armstrong became the first man to step on the surface of the moon in 1969, he famously proclaimed that the act was “one small step for a man, one giant leap for mankind”. Until this past decade, developments in the exploration of outer space have been more modest than the initial days of the Cold War-fueled space race. Early pioneers such as Wernher von Braun, the scientist behind some of the first rockets to launch mankind into space, planned for the moon landings to be a stepping stone for manned missions to Mars and beyond (Baker, 2021). During the decades following the last Apollo mission, endeavors outside of the Earth’s atmosphere were smaller in scale. Extraterrestrial activities were restricted to the government domain for countries with economies that could afford such exploration. This began to change in the decade beginning in 2010, as a handful of the world’s billionaires set their sights on the stars, bringing back a renewed public interest in outer space. Elon Musk of SpaceX hopes to build a colony on Mars, Amazon’s Jeff Bezos would like to move much of the world’s industrial complex to space, and Richard Branson wants Virgin Galactic to be the first “spaceline” for commercial space travel.

In the past, private industry’s involvement in the space program was in the form of competing for government contracts. For example in the United States, the government would manage the contracts and provide the financial backing, and companies would provide any outside materials and resources required to complete the mission. Rockets were built by the companies with the government approving the plans at each step from design to production. This changed as SpaceX began to privately develop rockets, which it could then sell the ability to move satellites and other cargo to space for any interested customer (Grush, 2019). Commercial entities now had the leverage to bring more innovative solutions to an industry that

had previously been limited by bureaucracy. This has caused the cost of getting to and operating in space to decrease (Knowledge@Wharton, 2019).

Beyond the scientific advancements that have been made due to space exploration, including advancements in weather prediction and knowledge gained about the history of our universe, mankind has benefitted from many technical advancements. NASA programs have led to the development of technology such as GPS, laptops, and cell phone cameras. The medical field can contribute CAT scans, artificial limbs, and LASIK eye surgery to developments created to monitor and assist astronauts (NASA Center for AeroSpace Information (CASI), 2008). Private industries hope to add to the value of going to space. More efforts are planned to use satellites to expand Earth-based communications and data collection. Private corporations are even looking at mining asteroids for rare valuable elements that may be used to keep up with the production demands for high-tech items as a more environmentally friendly option to mining on Earth (Knowledge@Wharton, 2019). Beyond that, companies like SpaceX, Blue Origin, and Virgin Atlantic have plans to begin ferrying civilians into space, which is sure to usher in a promising future for space tourism.

To keep momentum, the companies and agencies involved in the space realm will need to keep the general public on their side. Agencies like NASA have experienced a loss in budgeted funding when American citizens lost interest in the past. In the 1960s when there was a race between the USA and the Soviet Union to get to the moon, the government was willing to spare no expense for the space program. Once that race was over and public interest waned, budgetary priorities shifted (Hayward, n.d.). Public interest again took a downward turn in 1986, after the explosion of the Space Shuttle Challenger. This mission had been highly televised as it was the first to include a civilian, and American school teacher, and the tragedy and subsequent investigation halted missions for two years (Hayward, n.d.), and the program fell out

of public favor. Since both government space programs and private corporations rely on the use of money from taxpayers to continue operation, it could be advantageous to monitor public attitudes.

This study will use messages written by the public on the social media network Twitter to gauge the public sentiment towards entities in the space industry to answer the question "Is there a significant difference in the sentiment expressed in tweets involving commercial entities compared to government agencies involved space exploration?" Data will be collected for a period of two years (April 2019 through March 2021). The contribution of this study to the field of Data Analytics and the MSDA program is to create an application that can view the sentiment being expressed towards entities within the space industry, and compare the government and commercial sectors. This study will utilize natural language processing (NLP) techniques to determine the tone of the sentiment expressed in "tweets", or messages posted on Twitter. The data will then be segregated into two sets, based on if the entity being discussed is a government agency or a private corporation. Finally, the chi-square test of independence will be used to compare the public sentiment between sectors. This test compares categorically coded data with the expected frequencies due to chance alone (Freedman, Pisani, & Purves, 2014). For the test, the null hypothesis will be "There is no difference in sentiment between commercial and government entities", and the alternative hypothesis is "There is a significant difference in sentiment between commercial and government entities."

B. Data Collection

To perform the analysis, data was gathered in the form of tweets from the Twitter website (https://twitter.com/). Besides the text, each tweet has an associated object with more than thirty attributes that can be collected, which include information about the time that the

tweet was posted, user geographic location, and reactions from other users (Twitter, n.d.). For this project the date and time of the tweet, the individual tweet ID, the user name of the author, and the actual text of the tweet were the only attributes collected.

The data was collected using the tool Python tool snscrape, which was designed to collect data from several social media networks (JustAnotherArchivist, n.d.). To get a feel for the sentiment over a period of time, the first 2,000 tweets were collected for each month per entity for a period of two years.

To achieve this goal, the parameters for the snscrape tool were set to loop through a list of months (from April 2019 through March 2021) and another list of entities ('SpaceX', 'Virgin Galactic', 'Blue Origin', 'NASA', 'ESA', 'JPL'), collecting the first 2,000 tweets for each. During the loop, the entity name was added to the tweet data. The code in the figure below resulted in a dataframe with 285,167 rows and five columns. Due to the nature of how the tweet object is created, data sparsity is 0%.

```
# Create lists of entities and start/end dates
entities = ['SpaceX', 'Virgin Galactic', 'Blue Origin', 'NASA', 'ESA', 'JPL']
month_start = ['2021-03-01', '2021-02-01', '2021-01-01', '2020-12-01', '2020-11-01', '2020-10-01', '2020-09-01',
               '2020-08-01', '2020-07-01', '2020-06-01', '2020-05-01', '2020-04-01', '2020-03-01', '2020-02-01',
               '2020-01-01', '2019-12-01', '2019-11-01', '2019-10-01', '2019-09-01', '2019-08-01', '2019-07-01',
               '2019-06-01', '2019-05-01', '2019-04-01']
month_end   = ['2021-03-31', '2021-02-28', '2021-01-31', '2020-12-31', '2020-11-30', '2020-10-31', '2020-09-30',
               '2020-08-31', '2020-07-31', '2020-06-30', '2020-05-31', '2020-04-30', '2020-03-31', '2020-02-29',
               '2020-01-31', '2019-12-31', '2019-11-30', '2019-10-31', '2019-09-30', '2019-08-31', '2019-07-31',
               '2019-06-30', '2019-05-31', '2019-04-30']
```

```
# Create list to append tweet data to
tweets_list = []
```

```
# Using TwitterSearchScraper to scrape data and append tweets to list

import snscrape.modules.twitter as sntwitter

for entity in entities:
    for j in range(len(month_start)):
        for i,tweet in enumerate(sntwitter.TwitterSearchScraper(f"'{entity} since:{month_start[j]}
                                                                 until:{month_end[j]}'").get_items()):
            if i>2000:
                break
            tweets_list.append([tweet.date, tweet.id, tweet.content, tweet.user.username, entity])
```

```
# Creating a dataframe from the tweets list above
tweets_df = pd.DataFrame(tweets_list, columns=['Datetime', 'Tweet Id', 'Text', 'Username', 'Company'])
```

Figure 1: Data Collection and Dataframe Creation

Initially, the plan was to use the Twitter developer API interface to collect the data required. This requires the researcher to create a developer account that includes specifying the intentions for the project. Twitter then provides individual credentials that can be used to extract the data. To protect their servers from being inundated with requests, Twitter enforces "rate limits" which limit the amount of data that can be requested in a given time. When the rate limit is reached, the extraction will pause for several minutes before resuming. Initial attempts to retrieve the data in this manner took hours. One advantage of using the snscrape tool for extraction is that was able to retrieve more data in much less time than requesting it from the API.

Another downside to using the Twitter API is that it will only allow you to collect data for the last seven days. This created a challenge as it is not useful when wanting to analyze sentiment since a single event could change the perception of public opinion for a short period.

To eliminate bias, it is necessary to spread out the length of time over a longer duration. This challenge was overcome with the use of the snscrape method, as it has the flexibility to collect information from as far back as the inception of Twitter in any increment.

A disadvantage of the methodology for data collection is the inability to choose the monthly sampling at random. Some entities may be mentioned more often, therefore the 2,000 tweet limit may only capture a period of a few days. The odd number of records in the collected data means that some entities may not have had 2,000 tweets in particular months. Without having an equal amount of tweets for each entity dispersed randomly throughout each month, the analysis could be impacted by bias.

C. Data Extraction and Preparation

To extract and prepare the data for analysis, Python was used in a Jupyter Notebook. Python was chosen for data processing and manipulation based on its flexible nature due to the abundance of libraries that can be imported. Python contains more than two hundred standard libraries and a much larger quantity of third-party libraries that can be implemented, including several libraries dedicated to NLP (Bansa, 2020). The Jupyter Notebook was chosen as the development environment also due to its flexibility. Changes can be quickly made in a single block of code, then only the portions of code that need to be updated are executed after editing. This can conserve processing time.

The first step in the data preparation is to add a column for the sector of each entity, commercial or government. This was accomplished by applying a simple function using an if statement to test the name of the entity.

```
# Define a function to return the sector based on the entity name
def sector(name):
    return_type = ''
    if name == 'SpaceX' or name == 'Virgin Galactic' or name == 'Blue Origin':
        return_type = 'Commercial'
    else: return_type = 'Government'
    return return_type
```

```
# Apply sector function to each row of the dataframe and view head
df['Type'] = df['Company'].apply(sector)
df.head()
```

	Datetime	Tweet Id	Text	Username	Company	Type
0	2021-03-30 23:59:49+00:00	1377047947259576322	SpaceX Starship SN11 Rocket Fails to Land Safe...	toolsness	SpaceX	Commercial
1	2021-03-30 23:59:47+00:00	1377047938938060803	@danahjon @NASAKennedy @ArceneauxHayley @DrSia...	rookisaacman	SpaceX	Commercial
2	2021-03-30 23:59:38+00:00	1377047902044872706	@SpaceTfrs So grateful for your contribution t...	BlueJ11274903	SpaceX	Commercial
3	2021-03-30 23:59:34+00:00	1377047884101656576	Guess What Just Happened to The Latest SpaceX ...	Pauloaep	SpaceX	Commercial
4	2021-03-30 23:59:31+00:00	1377047870424104964	@WhatAFool6 @PushTheFrontier @TooChill_Javi No...	RubenJHenriquez	SpaceX	Commercial

Figure 2: Adding the Sector

Next, any rows with duplicate text were dropped from the dataframe. Duplicate text may occur when an item was re-tweeted by another user and might skew the results of the analysis by inflating the text to be labeled for sentiment. After eliminating the duplicates in the “Text” column, 276,208 rows of data remain from the original data set.

```
# Remove duplicate text
df_clean = df.drop_duplicates(subset='Text')
```

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 276208 entries, 0 to 285166
Data columns (total 6 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Datetime  276208 non-null  object
 1   Tweet Id  276208 non-null  int64
 2   Text      276208 non-null  object
 3   Username  276208 non-null  object
 4   Company   276208 non-null  object
 5   Type      276208 non-null  object
dtypes: int64(1), object(5)
memory usage: 14.8+ MB
```

Figure 3: Dropping Duplicate Text and Viewing Column and Row Information

The figure above showed that the “Datetime” column was formatted as a string object. The next step was to convert this column to a datetime data type.

```
# Convert Datetime column to Datetime type
df_clean['Datetime']= pd.to_datetime(df_clean['Datetime'])
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 276208 entries, 0 to 285166
Data columns (total 6 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   Datetime  276208 non-null  datetime64[ns, UTC]
 1   Tweet Id  276208 non-null  int64
 2   Text      276208 non-null  object
 3   Username  276208 non-null  object
 4   Company   276208 non-null  object
 5   Type      276208 non-null  object
dtypes: datetime64[ns, UTC](1), int64(1), object(4)
memory usage: 14.8+ MB
```

Figure 4: Convert “Datetime” Column to datetime64 Data Type

To eliminate bias, rows with tweets from the official accounts of the entities were dropped from the dataframe. Upon inspection, this involved less than 200 tweets.

```
# Check for tweets from official entity Twitter accounts
print("The number of tweets from the official SpaceX account: ", len(df_clean[df_clean['Username'] == 'SpaceX']))
print("The number of tweets from the official Virgin Galactic account: ", len(df_clean[df_clean['Username'] ==
                                                                                'virgingalactic']))
print("The number of tweets from the official Blue Origin account: ", len(df_clean[df_clean['Username'] == 'blueorigin']))
print("The number of tweets from the official NASA account: ", len(df_clean[df_clean['Username'] == 'NASA']))
print("The number of tweets from the official ESA account: ", len(df_clean[df_clean['Username'] == 'esaoperations']))
print("The number of tweets from the official JPL account: ", len(df_clean[df_clean['Username'] == 'NASAJPL']))
```

```
The number of tweets from the official SpaceX account:  5
The number of tweets from the official Virgin Galactic account:  67
The number of tweets from the official Blue Origin account:  3
The number of tweets from the official NASA account:  52
The number of tweets from the official ESA account:  1
The number of tweets from the official JPL account:  16
```

```
# Get names of indexes for which column Username is one of the studied entities
indexNames = df_clean[(df_clean['Username'] == 'SpaceX') | (df_clean['Username'] == 'virgingalactic')
                      | (df_clean['Username'] == 'blueorigin') | (df_clean['Username'] == 'NASA')
                      | (df_clean['Username'] == 'esaoperations') | (df_clean['Username'] == 'NASAJPL')].index
# Delete these row indexes from dataFrame
df_clean.drop(indexNames , inplace=True)
```

Figure 5: Inspection and Elimination of Tweets from Official Entity Accounts

The text for the tweets was then cleaned. Mentions (ex. @NASA), hashtags (ex. #NASA), URL links, non-alphabetic characters, and retweets were removed. This was accomplished with the use of regular expression syntax in a user-defined function. This function was then applied to the “Text” column of each row in the data frame.

```
# Cleaning the tweets
def cleanUpTweet(txt):
    # Remove mentions
    txt = re.sub(r'@[A-Za-z0-9_]+', '', txt)
    # Remove hashtags
    txt = re.sub(r'#', '', txt)
    # Remove retweets:
    txt = re.sub(r'RT : ', '', txt)
    # Remove other undesirable text
    txt = re.sub(r'b\'', '', txt)
    # Remove urls
    txt = re.sub(r'https?:\/\/[A-Za-z0-9\.\/]+', '', txt)
    return txt
```

```
# Apply cleaning method to each tweet
df_clean['Text'] = df_clean['Text'].apply(cleanUpTweet)
```

Figure 6: Defining the Text Cleaning Function and Applying to “Text” Column

After applying the data cleaning function, some of the “Text” fields were left empty. Therefore, those rows were dropped from the dataframe. After this elimination, 276,013 rows remained.

```
# Ensure there are no zero value columns after cleaning
df_clean[df_clean['Text'] == 0]
```

	Datetime	Tweet Id	Text	Username	Company	Type

```
# Ensure there are no empty value columns after cleaning
df_clean[df_clean['Text'] == '']
```

268022	2019-12-27 16:45:06+00:00	1210602514485334016	ricardo_jpl	JPL	Government
268037	2019-12-27 16:15:45+00:00	1210595130660532224	rocaroja_jpl	JPL	Government
268998	2019-12-24 19:32:26+00:00	1209557462472364033	sol_acad_fc_JPL	JPL	Government
269476	2019-11-28 14:56:37+00:00	1200065967302283274	ricardo_jpl	JPL	Government
270487	2019-11-24 16:31:23+00:00	1198640263234756608	JPL_Reader	JPL	Government
270572	2019-11-24 13:48:46+00:00	1198599340245303296	Arch_JPL	JPL	Government
271014	2019-11-23 02:37:38+00:00	1198068056200634369	JPL_Richard	JPL	Government
275403	2019-08-30 05:38:53+00:00	1167310701926338561	sol_acad_fc_JPL	JPL	Government
277079	2019-08-24 20:30:46+00:00	1165360823868375040	dr_jpl	JPL	Government
277292	2019-07-30 14:59:03+00:00	1156217646259548162	sol_acad_fc_JPL	JPL	Government
280276	2019-06-26 08:44:53+00:00	1143802299086757889	ricardo_jpl	JPL	Government
281515	2019-05-29 14:18:54+00:00	1133739496221040640	dr_jpl	JPL	Government
282101	2019-05-27 10:19:26+00:00	1132954457128558597	Cinderford_JPL	JPL	Government

```
# Drop rows with empty text column
df_clean = df_clean.drop(df_clean[df_clean['Text'] == ''].index)
```

Figure 7: Dropping rows with empty "Text" field

The final step to prepare the data for analysis was to add the sentiment labels. This was accomplished using TextBlob, which is a library that can be used for NLP tasks including calculating the polarity of text (TextBlob, n.d.). First, TextBlob was called in a user-defined function to assign a polarity score to the "Polarity" column of each tweet. This score has a range of negative one to one (-1 to 1). Then, another user-defined function was created to assign each score to a text label. Negative polarity values were assigned a "negative" label, positive values were given the label "positive", and a score of zero was assigned the label "neutral".

```
# Calculate polarity of each tweet
from textblob import TextBlob

def getTextPolarity(txt):
    return TextBlob(txt).sentiment.polarity
```

```
# Add the polarity values to each tweet
df_clean['Polarity'] = df_clean['Text'].apply(getTextPolarity)
```

```
# Analyze text for negative, neutral, positive sentiment
def getTextAnalysis(a):
    if a < 0:
        return "Negative"
    elif a == 0:
        return "Neutral"
    else:
        return "Positive"
```

```
# Apply sentiment label to each text
df_clean['Score'] = df_clean['Polarity'].apply(getTextAnalysis)
```

```
df_clean.head(3)
```

	Datetime	Tweet Id	Text	Username	Company	Type	Polarity	Score
0	2021-03-30 23:59:49+00:00	1377047947259576322	SpaceX Starship SN11 Rocket Fails to Land Safe...	toolsness	SpaceX	Commercial	0.0	Neutral
1	2021-03-30 23:59:47+00:00	1377047938938060803	Frak.	rookisaacman	SpaceX	Commercial	0.0	Neutral
2	2021-03-30 23:59:38+00:00	1377047902044872706	So grateful for your contribution to helping ...	BlueJ11274903	SpaceX	Commercial	0.0	Neutral

Figure 8: Calculating the Polarity Score and Assigning to Text Labels

An advantage of using TextBlob to assign the polarity is that it provides a simple interface to the Natural Language Toolkit (NLTK), which is the standard library used for NLP tasks (Jain, 2020). More than two hundred thousand texts were able to be analyzed for sentiment using only a few lines of Python code. A disadvantage is that TextBlob tends to be slower than other NLP libraries at processing tasks (Jain, 2020). The processing time for this data set took several minutes to complete.

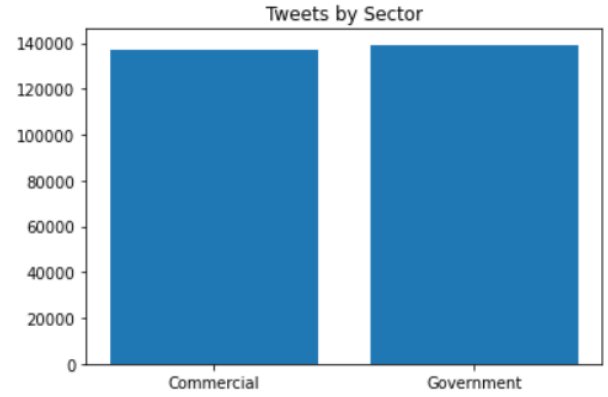
D. Analysis

Visualizing Sentiment Distribution

After cleaning and preparing the data, the number of tweets per sector is fairly even. There are approximately 2,500 more tweets in the data labeled as “Government”, but since the

total dataset contains nearly 300,000 tweets, this is less than a 1% difference, therefore it should not affect the results of the analysis.

```
# Plot overall count of tweets by sector
labels = df_clean.groupby('Type').count().index.values
values = df_clean.groupby('Type').size().values
plt.bar(labels, values)
plt.title('Tweets by Sector');
```



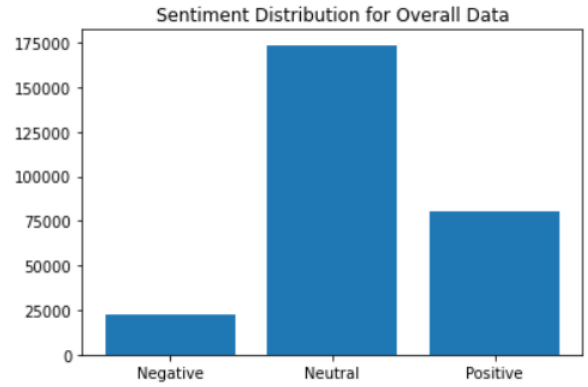

```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Commercial 136812
Government 139201
```

Figure 9: Tweets by Sector

In viewing the overall sentiment on a bar chart, it seems that the majority of the text in the tweets is classified as neutral sentiment. In terms of polarized sentiment, positive sentiment occurs much more often than negative sentiment.

```
# Plot overall sentiment values
labels = df_clean.groupby('Score').count().index.values
values = df_clean.groupby('Score').size().values
plt.bar(labels, values)
plt.title('Sentiment Distribution for Overall Data');
```



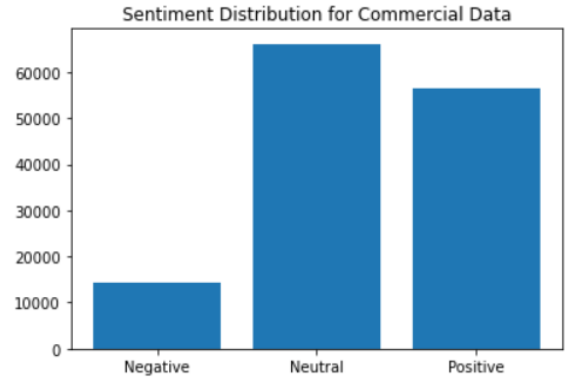

```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Negative 22295
Neutral 173725
Positive 79993
```

Figure 10: Sentiment Distribution for Overall Data

When looking at each sector individually, there are more instances of neutral sentiment than other categories. In the commercial sector, however, positive sentiment is nearly equal to neutral sentiment and much greater than negative sentiment. In the government sector, the neutral category is far greater than the other two labels.

```
# Plot overall sentiment values for commercial sector
labels = df_com.groupby('Score').count().index.values
values = df_com.groupby('Score').size().values
plt.bar(labels, values)
plt.title('Sentiment Distribution for Commercial Data');
```

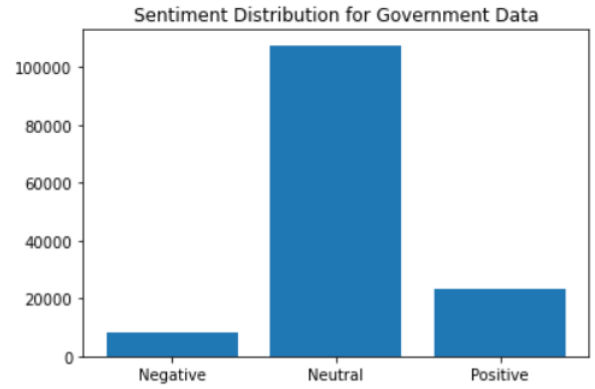



```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Negative 14151
Neutral 66135
Positive 56526
```

Figure 11: Sentiment Distribution for Commercial Data

```
# Plot overall sentiment values for government sector
labels = df_gov.groupby('Score').count().index.values
values = df_gov.groupby('Score').size().values
plt.bar(labels, values)
plt.title('Sentiment Distribution for Government Data');
```




```
# Print value counts for each sentiment label
for i in range(len(labels)):
    print(labels[i], values[i])
```

```
Negative 8144
Neutral 107590
Positive 23467
```

Figure 12: Sentiment Distribution for Government Data

Each entity in the commercial sector seems pretty evenly matched, with positive sentiment far exceeding negative sentiment. In each case, the positive sentiment is very close in number to the tweets labeled neutral.

```
# Plot Sentiment Values By Commercial Entity
sns.countplot(x="Company", hue="Score", data=df_com).set_title("Sentiment of Commercial Entities");
```

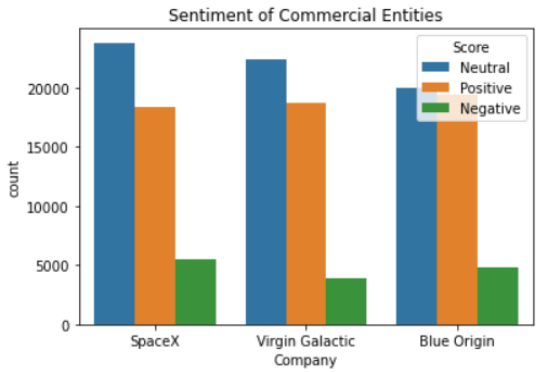


Figure 13: Sentiment Distribution for Each Entity in Commercial Sector

The government sector appears to be much different from the commercial sector in its sentiment distribution. Overall, there is much less polarized sentiment than neutral sentiment for each entity.

```
# Plot Sentiment Values By Government Entity
sns.countplot(x="Company", hue="Score", data=df_gov).set_title("Sentiment of Government Entities");
```

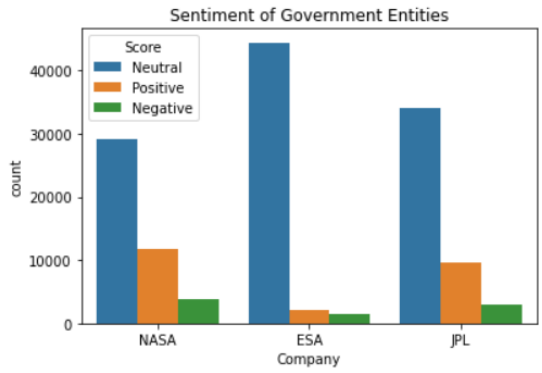


Figure 14: Sentiment Distribution for Each Entity in Government Sector

Visualizing Sentiment over Time

The polarity score can also be viewed over time to get an idea if there are any trends in changing sentiment. To accomplish this, the data is grouped by day and then the mean is calculated. Since the 2,000 tweet cutoff may limit the range of the data, some days may have no data associated with it and therefore any null values for the mean are dropped. For the overall tweet dataframe, there do not appear to be any trends, and the distribution appears random. The overall daily scores seem to stay above zero.

```
# Calculate daily average polarity for all tweets
df_time = df_clean.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
```

```
# Plot total polarity over time
df_time.plot(y='Polarity')
plt.title('Total Polarity Over Time');
```

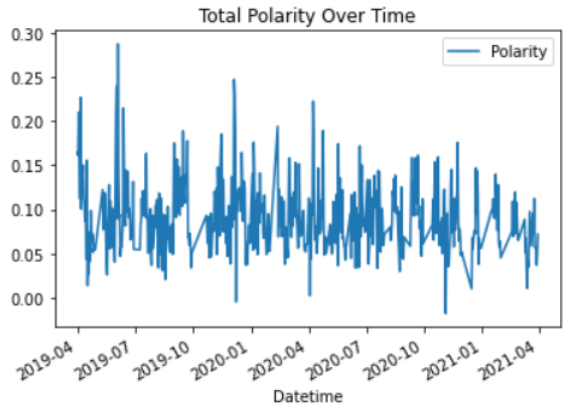


Figure 15: Total Polarity over Time

The average daily sentiment is then calculated for each sector. There appear to be no trends when looking at the commercial sector polarity scores over time as the distribution appears random and above zero. The government distribution appears to be less noisy, but this could be due to having a smaller range for polarity score than the commercial sector.

```
# Calculate daily average polarity for each sector
df_com_time = df_com.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_time = df_gov.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
```

```
# Plot commercial polarity over time
df_com_time.plot(y='Polarity')
plt.title('Commercial Polarity Over Time');
```

```
# Plot government polarity over time
df_gov_time.plot(y='Polarity')
plt.title('Government Polarity Over Time');
```

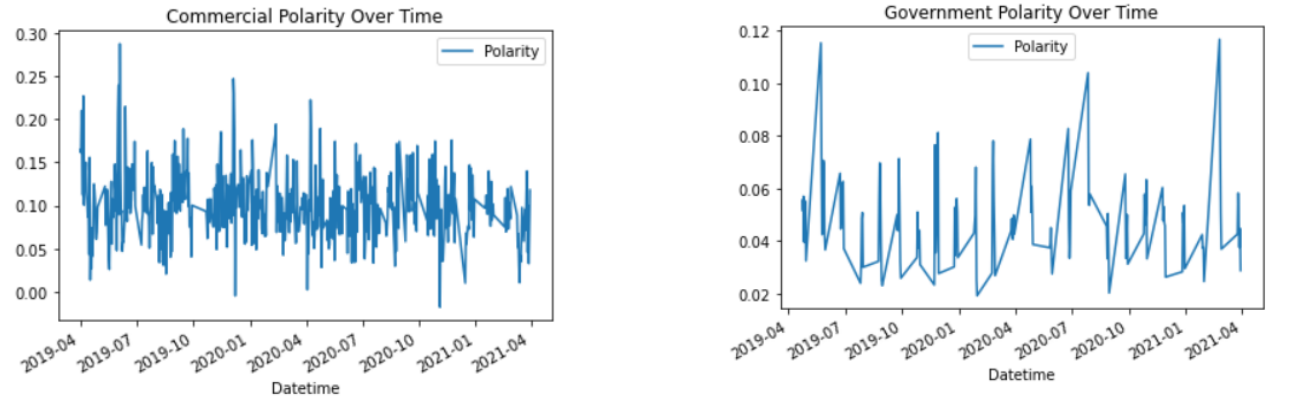


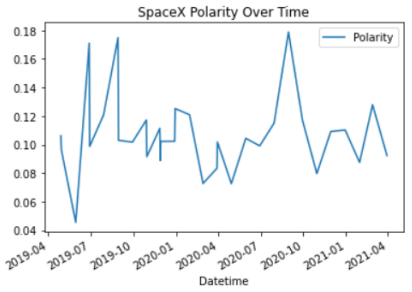
Figure 16: Polarity over Time by Sector

Next, the sentiment over time for each entity was calculated and plotted. This was done by first breaking the data into separate data frames by each entity. Then the average polarity by day was calculated and any null values dropped. The SpaceX, NASA, and ESA plots are less noisy than Virgin Galactic, Blue Origin, and JPL. This means that those entities with less noisy plots may experience trends in polarity. ESA stands out as having more negative average daily sentiment than any of the other entities, all of which have mostly positive daily sentiment.

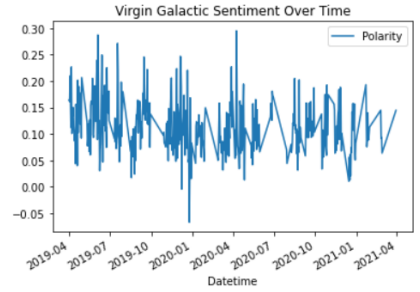
```
# Separate data by entity
df_com_spa = df_com[df_com['Company'] == 'SpaceX']
df_com_vir = df_com[df_com['Company'] == 'Virgin Galactic']
df_com_blu = df_com[df_com['Company'] == 'Blue Origin']
df_gov_nas = df_gov[df_gov['Company'] == 'NASA']
df_gov_esa = df_gov[df_gov['Company'] == 'ESA']
df_gov_jpl = df_gov[df_gov['Company'] == 'JPL']
```

```
# Calculate daily average polarity for each entity
df_com_spa_time = df_com_spa.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_com_vir_time = df_com_vir.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_com_blu_time = df_com_blu.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_nas_time = df_gov_nas.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_esa_time = df_gov_esa.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
df_gov_jpl_time = df_gov_jpl.groupby(pd.Grouper(key='Datetime', freq="D")).mean().dropna()
```

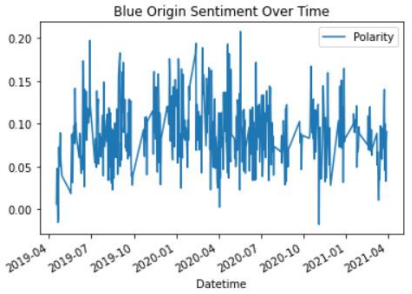
```
# Plot SpaceX polarity over time
df_com_spa_time.plot(y='Polarity')
plt.title('SpaceX Polarity Over Time');
```




```
# Plot Virgin Galactic polarity over time
df_com_vir_time.plot(y='Polarity')
plt.title('Virgin Galactic Sentiment Over Time');
```

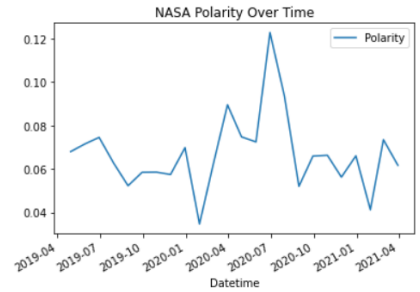



```
# Plot Blue Origin polarity over time
df_com_blu_time.plot(y='Polarity')
plt.title('Blue Origin Sentiment Over Time');
```

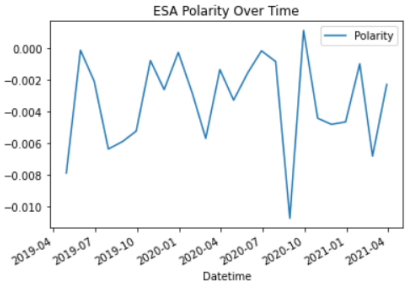



```
# Plot NASA polarity over time
df_gov_nas_time.plot(y='Polarity')
plt.title('NASA Polarity Over Time');
```

```
<Figure size 1440x1440 with 0 Axes>
```




```
# Plot ESA polarity over time
df_gov_esa_time.plot(y='Polarity')
plt.title('ESA Polarity Over Time');
```




```
# Plot JPL polarity over time
df_gov_jpl_time.plot(y='Polarity')
plt.title('JPL Polarity Over Time');
```

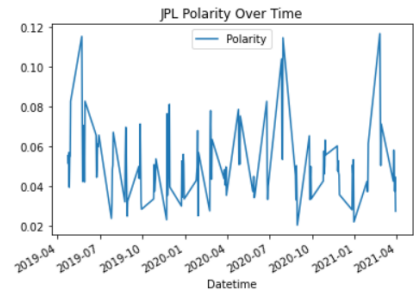


Figure 17: Polarity over Time by Entity

Chi-square Analysis

Hypothesis testing will be done using the Chi-square test for independence. This test compares two categorical variables to determine if the distributions vary from each other (Glen, 2020). The test statistic is calculated using the observed and expected values from a frequency table. This statistic is a single value that describes the amount of difference between the observed frequency and the frequency that is expected if there were no relationship. The test statistic can be used to calculate a p-value to determine the significance of the relationship between the variables. One advantage of using the Chi-square test it can be used on categorical data as opposed to strictly numeric values. A disadvantage is that each instance of the data must fit into a single category, for example, an entity cannot be both in the Commercial and Government sectors (Namuth-Covert, Merk, & Haines, 2021).

The first step in the Chi-square analysis is to create a contingency table, which is the frequency of each sentiment label broken out by sector. This is done by using the pandas.crosstab() function.

```
# Perform Chi-square analysis
from scipy.stats import chi2_contingency

# Create contingency table
contigency= pd.crosstab(df_clean['Type'], df_clean['Score'])
contigency
```

Score	Negative	Neutral	Positive
Type			
Commercial	14151	66135	56526
Government	8144	107590	23467

Figure 18: Contingency Table for Chi-square Analysis

This can also be viewed in terms of a percentage of sentiment for each sector. This is done by normalizing the data in the table created using the pandas.crosstab() function.

```
# Calculate sentiment percentage by sector
contigency_pct = pd.crosstab(df_clean['Type'], df_clean['Score'], normalize='index')
round(contigency_pct, 2)
```

Score	Negative	Neutral	Positive
Type			
Commercial	0.10	0.48	0.41
Government	0.06	0.77	0.17

Figure 19: Contingency Table in Percentage Form

Next, the chi-square test for independence was performed. This step created variables for the test statistic, p-value, degrees of freedom, and the expected values array. The p-value was then printed, as this value is used to determine the significance of the difference between the data for both sectors. This value is the determining factor on whether the null hypothesis is accepted or rejected. The p-value of 0.0 is statistically significant, therefore the null hypothesis can be rejected.

```
# Chi-square test of independence
c, p, dof, expected = chi2_contingency(contigency)
# Print the p-value
print("The p-value of the Chi-square analysis is:", p)
```

```
The p-value of the Chi-square analysis is: 0.0
```

Figure 20: Chi-square Test for Independence and P-value

Finally, the expected values table and heatmap are displayed as a comparison to the contingency table values.

```
expected_tbl = pd.DataFrame(data=np.round(expected, decimals=0), index=['Commercial', 'Government'],
                            columns=['Negative', 'Neutral', 'Positive'])
expected_tbl.astype(int)
```

	Negative	Neutral	Positive
Commercial	11051	86111	39650
Government	11244	87614	40343

```
# View expected table on a heatmap
plt.figure(figsize=(7,3))
sns.heatmap(expected_tbl, annot=True, cmap='YlGnBu', fmt='g').set_title("Expected Sentiment by Sector Heatmap");
```

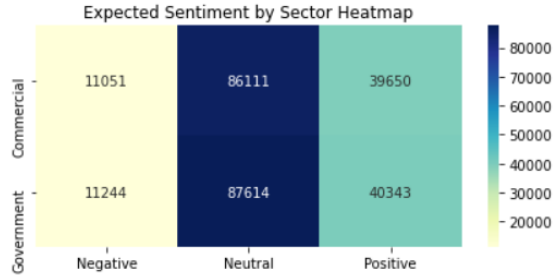


Figure 21: Expected Values Table and Heatmap

```
# View contingency table on a heatmap
plt.figure(figsize=(7,3))
sns.heatmap(contigency, annot=True, cmap='YlGnBu', fmt='g').set_title("Observed Sentiment by Sector Heatmap");
```

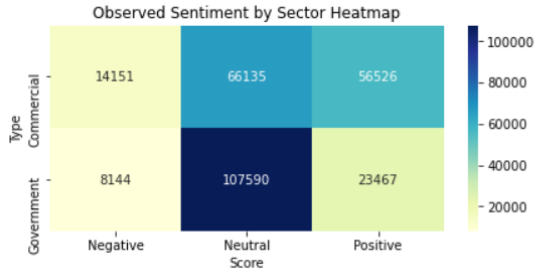


Figure 22: Contingency Table Heatmap

E. Data Summary and Implications

A chi-square p-value of less than five percent indicates that the sentiment values for the government and commercial space sectors are significantly different. The chi-square test returned a value of 0.0, which is grounds to reject the null hypothesis “There is no difference in sentiment between commercial and government entities”. As seen in figure 12, government

sentiment seems to be overwhelmingly neutral. The commercial sentiment seems to be a bit more polarized, but figure 11 shows, that sentiment tends to be more positive.

This study is limited not only due to the collection method (2,000 tweet limit) that was mentioned earlier but also in the small list of entities studied. There are many more nations, government agencies, and private sector companies involved in the exploration of space.

One recommendation for future studies would be to include more than just the six entities used in this study. This would provide a more well-rounded world view of public interest in the exploration of space. Another proposal for future studies would be to gather data not just from Twitter but to include other social media networks. Different types of people tend to have different preferences on the preferred social media platform, therefore adding more information would add to the perspective.

F. References

Advancing Academic Research with Twitter Data | Twitter Developer. (n.d.). Retrieved April 11, 2021, from https://developer.twitter.com/en/solutions/academic-research

Baker, D. (2021, March 25). Wernher von Braun's Forgotten Mission to Mars. Retrieved April 21, 2021, from https://www.skyatnightmagazine.com/space-missions/wernher-von-brauns-forgotten-mission-mars/

Bansa, N. (2020, September 30). R vs Python for Data Science: The Best Data Science Language. Retrieved April 12, 2021, from https://www.sisense.com/blog/r-vs-python-whats-the-best-language-for-natural-language-processing/

Freedman, D., Pisani, R., & Purves, R. (2014). Statistics. New York: Norton.

Glen, S. (2020, December 10). Chi-Square Statistic: How to Calculate It / Distribution. Retrieved April 20, 2021, from https://www.statisticshowto.com/probability-and-statistics/chi-square/

Grush, L. (2019, December 11). This was the Decade the Commercial Spaceflight Industry Leapt Forward. Retrieved April 21, 2021, from https://www.theverge.com/2019/12/11/20981714/spacex-commercial-spaceflight-space-industry-decade-nasa-business

Hayward, S. (n.d.). The Space Shuttle Program and the Challenger Disaster. Retrieved April 21, 2021, from https://billofrightsinstitute.org/essays/the-space-shuttle-program-and-the-challenger-disaster

Jain, S. (2020, December 23). NLP for BEGINNERS: Text classification Using TextBlob. Retrieved April 19, 2021, from https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/

Knowledge@Wharton. (2019, June 4). The Commercial Space Economy: Business is Making a 'Giant Leap'. Retrieved April 21, 2021, from https://knowledge.wharton.upenn.edu/article/commercial-space-economy/

Namuth-Covert, D., Merk, H., & Haines, C. (2021). When Chi-square is Appropriate - Strengths/Weaknesses. Retrieved April 20, 2021, from https://passel2.unl.edu/view/lesson/9beaa382bf7e/14

NASA Center for AeroSpace Information (CASI). (2008). Spinoff: 50 Years of NASA -Derived Technologies (1958-2008). Retrieved April 21, 2021, from https://spinoff.nasa.gov/Spinoff2008/pdf/spinoff2008.pdf

TextBlob. (n.d.). Simplified Text Processing¶. Retrieved April 12, 2021, from https://textblob.readthedocs.io/en/dev/