Trump Twitter Analysis

Group 14: MDS 522 : Members: Quan Hoang, Mailys Guegon, Joel Peterson, Li Pu

Analyzing the **realDonaldTrump_in_office.csv** from
https://github.com/MarkHershey/CompleteTrumpTweetsArchive/blob/master/data/realDo

For the initial EDA and data processing we ended up having to drop around 1/2 of the rows to stop unconventional characters at the end of the tweet string from tripping up pandas. After a visual review we noticed that these problem rows were quite evenly distributed (every 2-3 rows) and that this wouldn't be too much of an issue to get started.

Some somewhat common-sense assumptions we made were that the frequency of tweets would be highest during the daytime and night-time. These assumptions were proven wrong as the numbers show that the most frequent time of day for tweets was the overnight period.

Question 1: Does the time of day/period of the year affect the frequency of the tweets?

- The initial data filtering by time of day visualized in the bar chart shows that the highest frequency of tweets occurs during the overnight period between 12:01am and 8:00am (as mentioned above) with a total of 3706 analyzed tweets. The second most frequent tweet period occurred during the nighttime period between 4:01pm-12:00am with a total of 3653 analyzed tweets, with the time of day resulting in the least frequent amount of tweets actually being the daytime period between 8:01am-4:00pm with a total of 3325 analyzed tweets. Perhaps this is because Trump is most busy during the day (golfing?) and cannot tend to tweets.

- The seasonal filtering of the tweets returned the results of the most frequent amount of tweets being in the summer with 3148 analyzed tweets, with the autumn being the second busiets season for tweets resulting in 3067 analysed tweets, followed by the spring recording 2520 analyzed tweets, and lastly the winter being the least busy season for tweets recording a number of 1949 tweets. This falls inline with inutitive assumptions about mood and activity as the summer is usually the busiest with longer days (daylight) and nicer weather, then the fall slowly tapers off into the winter season being the shortest days for daylight and the most restricting weather.

In [1]:
```
import pandas as pd
import altair as alt


url = "https://raw.githubusercontent.com/MarkHershey/CompleteTrumpTweetsArch

tweets = pd.read_csv(
    url,
    encoding="utf-8-sig",                         # removes a BOM "Byte Order
    on_bad_lines="skip"                                   # skip the lines enc
)
tweets.columns = tweets.columns.str.strip()                         # strip
#print(tweets.columns)
tweets["Date & Time"] = pd.to_datetime(tweets["Time"], errors="coerce")
tweets = tweets.drop(columns=["ID", "Tweet URL", "Time"])
tweets = tweets.set_index("Date & Time")
tweets.head(10)
```

Out[1]:

Date & Time	Tweet Text
2017-01-20 06:31:00	"It all begins today! I will see you at 11:00...
2017-01-20 11:54:00	"We will bring back our jobs. We will bring b...
2017-01-20 11:55:00	"We will follow two simple rules: BUY AMERICA...
2017-01-20 11:58:00	"It is time to remember that...https://www.fa...
2017-01-20 12:13:00	"TO ALL AMERICANS https://www.facebook.com/Do...
2017-01-21 05:53:00	"A fantastic day and evening in Washington D....
2017-01-22 06:47:00	"Watched protests yesterday but was under the...
2017-01-23 05:38:00	"Busy week planned with a heavy focus on jobs...
2017-01-24 05:11:00	"Will be meeting at 9:00 with top automobile ...
2017-01-24 10:58:00	"A photo delivered yesterday that will be dis...

In [2]:
```
daytime = tweets.between_time("08:01", "16:00")      # 8:01 am - 4:00 pm
evening = tweets.between_time("16:01", "00:00")        # 4:01 pm - 12:00 am
overnight = tweets.between_time("00:01", "08:00")        # 12:01 am - 8:00 a

print(f"The number of tweets in the 'Daytime' category is: {len(daytime)}")
print(f"The number of tweets in the 'Evening' category is: {len(evening)}")
print(f"The number of tweets in the 'Overnight' category is: {len(overnight)

print("Just by the distribution of tweets it is surprisingly evenly spread;
The number of tweets in the 'Daytime' category is: 3325
The number of tweets in the 'Evening' category is: 3653
The number of tweets in the 'Overnight' category is: 3706
Just by the distribution of tweets it is surprisingly evenly spread; with th
e most tweets happening overnight.
```

In [3]:
```
tw = pd.Series(tweets.index.strftime('%m-%d'), index=tweets.index)     # Cha

spring = tweets.loc[tw.between('04-01', '06-30')]                    # April
summer = tweets.loc[tw.between('07-01', '09-30')]                   # July 1
autumn = tweets.loc[tw.between('10-01', '12-31')]                  # October
winter = tweets.loc[tw.between('01-01', '03-31')]                 # January

print(f"The number of tweets in the 'Spring' category is: {len(spring)}")
print(f"The number of tweets in the 'Summer' category is: {len(summer)}")
print(f"The number of tweets in the 'Autumn' category is: {len(autumn)}")
print(f"The number of tweets in the 'Winter' category is: {len(winter)}")
print("")
print("Judging by the distribution of tweets throughout the seasons, the fre
```

```
The number of tweets in the 'Spring' category is: 2520
The number of tweets in the 'Summer' category is: 3148
The number of tweets in the 'Autumn' category is: 3067
The number of tweets in the 'Winter' category is: 1949

Judging by the distribution of tweets throughout the seasons, the frequency
decays in the winter and spring and peaks in the summer
```

In [4]:
```
tweet_time = pd.DataFrame({
    "Time": ["daytime", "evening", "overnight"],
    "Count": [len(daytime), len(evening), len(overnight)]
})

tweet_time
```

Out[4]:

	Time	Count
0	daytime	3325
1	evening	3653
2	overnight	3706

In [5]:
```
tweet_season = pd.DataFrame({
    "Season": ["Spring", "Summer", "Autumn", "Winter"],
    "Count": [len(spring), len(summer), len(autumn), len(winter)]
})

tweet_season
```

Out[5]:

	Season	Count
0	Spring	2520
1	Summer	3148
2	Autumn	3067
3	Winter	1949

In [6]:
```
time_bars = alt.Chart(tweet_time).mark_bar(color="#f00808").encode(
    y = "Time:N",
    x = "Count:Q",
    ).properties(
    title="Trump's Tweet Frequency by Time of Day",
    width=500,
    height=350)

#time_bars
```

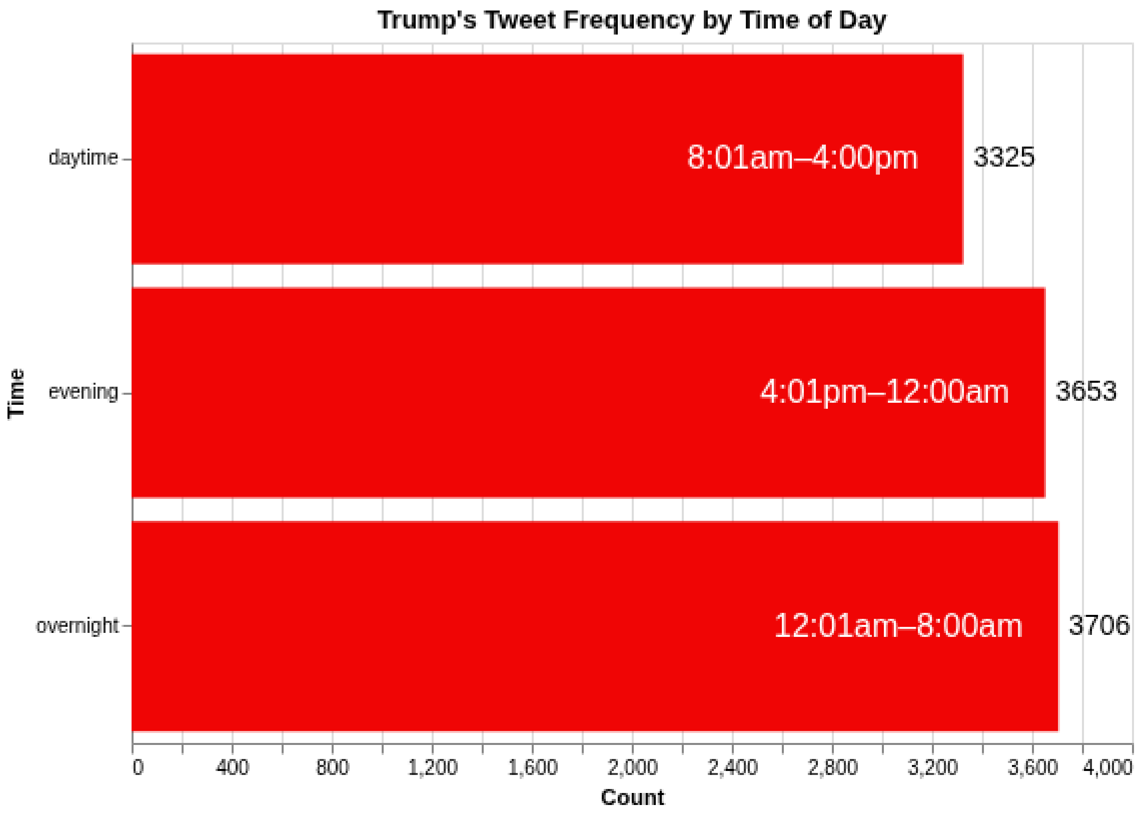
In [7]:
```
label_df = pd.DataFrame({
    "Time": ["daytime", "evening", "overnight"],
    "range": ["8:01am–4:00pm", "4:01pm–12:00am", "12:01am–8:00am"],
    "Count": tweet_time["Count"].values
})
```

In [8]:
```
range_text = (
    alt.Chart(label_df)
    .mark_text(
        align="center",
        baseline="middle",
        color="white",
        fontSize=16,
        dx=-80
    )
    .encode(
        y="Time:N",
        x="Count:Q",
        text="range:N"
    )
)

#time_bars + range_text
```

In [9]:
```
count_text = alt.Chart(label_df).mark_text(
    align="left",
    baseline="middle",
    dx=5,
    color="black",
    fontSize=14
).encode(
    y="Time:N",
    x="Count:Q",
    text="Count:Q"
)

tweet_times = time_bars + range_text + count_text
#tweet_times
```

In [10]:
```
final_time_of_day_chart = (time_bars + range_text + count_text).properties(p

final_time_of_day_chart
```

Out[10]:



In [11]:
```
season_bars = alt.Chart(tweet_season).mark_bar(color="#f00808").encode(
    x="Count:Q",
    y=alt.Y("Season:N",sort=["Spring", "Summer", "Autumn", "Winter"])
    ).properties(
    title="Trump's Tweet Frequency by Season",
    width=500,
    height=350)

#season_bars
```

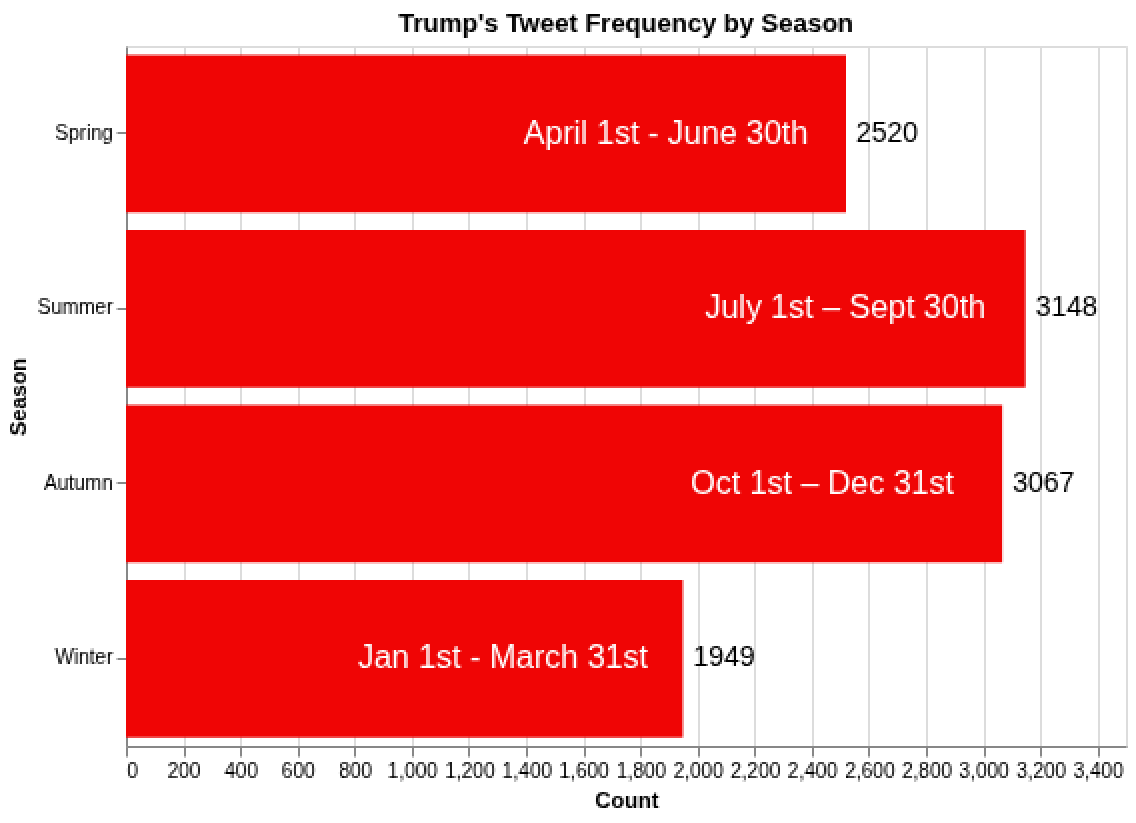
In [12]:
```
season_label_df = pd.DataFrame({
    "Season": ["Spring", "Summer", "Autumn", "Winter"],
    "range": ["April 1st - June 30th", "July 1st – Sept 30th", "Oct 1st – De
    "Count": tweet_season["Count"].values
})
#season_label_df
```

In [13]:
```
season_range_text = alt.Chart(season_label_df).mark_text(
        align="center",
        baseline="middle",
        color="white",
        fontSize=16,
        dx=-90
    ).encode(
        y=alt.Y("Season:N",sort=["Spring", "Summer", "Autumn", "Winter"]),
        x="Count:Q",
        text="range:N"
    )

#season_bars + season_range_text
```

In [14]:
```
season_count_text = alt.Chart(season_label_df).mark_text(
    align="left",
    baseline="middle",
    dx=5,
    color="black",
    fontSize=14
).encode(
    y=alt.Y("Season:N",sort=["Spring", "Summer", "Autumn", "Winter"]),
    x="Count:Q",
    text="Count:Q"
)

tweet_seasons = season_bars + season_range_text + season_count_text
#tweet_seasons
```

In [15]:
```
final_season_chart = (season_bars + season_range_text + season_count_text).p

final_season_chart
```

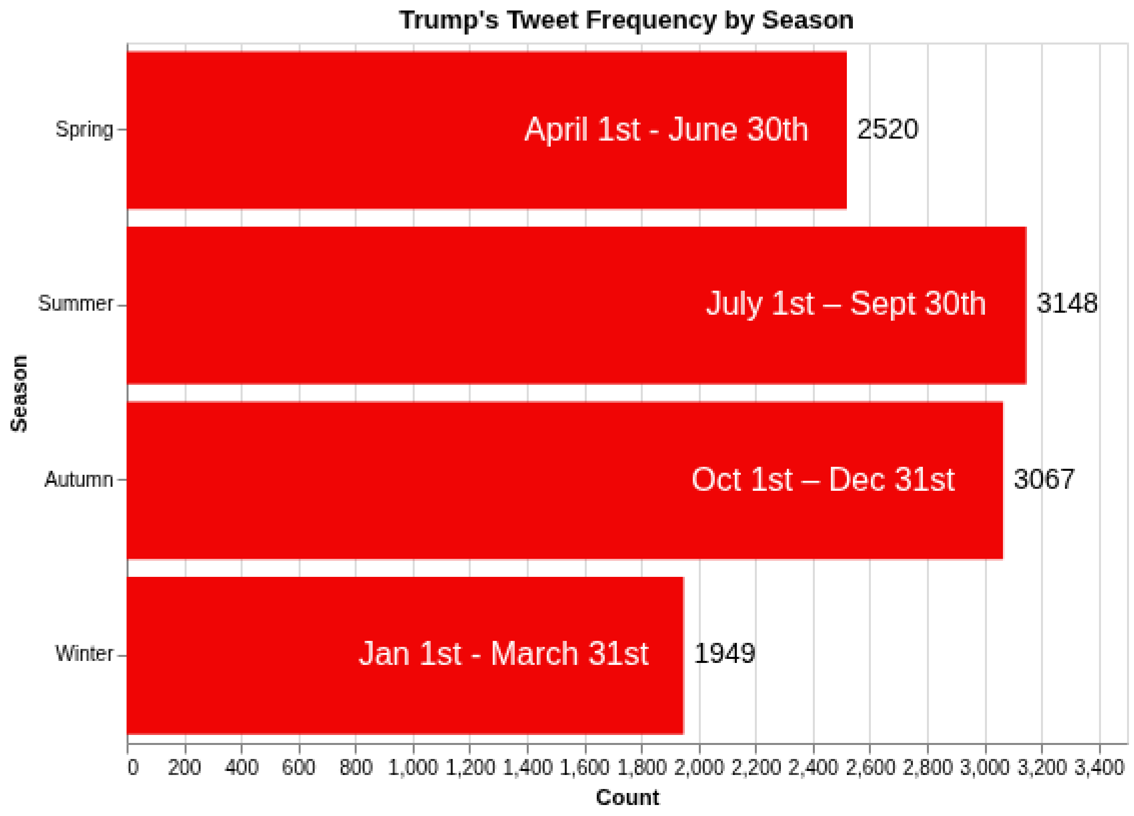
Out[15]:



In [16]:
```
#tweet_seasons | tweet_times
```

In [17]:
```
tweet_charts = (tweet_seasons | tweet_times).properties(padding={"right": 50
tweet_charts
```

Out[17]:



Question 2: How many tweets are positive vs negative? (2 classification methods)

We use a simple sentiment analysis model (VADER) to classify each tweet as positive, negative, or neutral, and then compare the counts, this methodological inspiration is from ChatGPT.

Method 1: Using the VADER sentiment analyzer the tweets were classified as wither positive, neutral, or negative. Some assumptions made were that the tweets were likely more negative than positive, but in fact the tweets were classified as far more positive than negative, almost doubling the negatively classified tweets. The tweets classified as 'neutral' fall almost directly in between the tweets classified as positive and negative. The actual context and literal meaning of the tweets was not analyzed, nor the accuracy of the classification at this point, so there is certainly going to be a margin of error and some 'false positives' etc as the positivity is quite often a self directed compliment of sorts.

```
In [18]: #import nltk
         from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
         sia = SentimentIntensityAnalyzer()
```

```
In [19]: tweets_q2 = tweets.copy()
         tweets_q2["Tweet Text"] = tweets_q2["Tweet Text"].fillna("").astype(str)

         tweets_q2["sentiment_score"] = tweets_q2["Tweet Text"].apply(
             lambda t: sia.polarity_scores(t)["compound"]
         )

         def score_to_label(score, pos_threshold=0.05, neg_threshold=-0.05):
             if score >= pos_threshold:
                 return "positive"
             elif score <= neg_threshold:
                 return "negative"
             else:
                 return "neutral"

         tweets_q2["sentiment_label"] = tweets_q2["sentiment_score"].apply(score_to_l

         tweets_q2[["Tweet Text", "sentiment_score", "sentiment_label"]].head(10)
```

Out[19]:

Date & Time	Tweet Text	sentiment_score	sentiment_label
2017-01-20 06:31:00	"It all begins today! I will see you at 11:00...	0.0000	neutral
2017-01-20 11:54:00	"We will bring back our jobs. We will bring b...	0.7345	positive
2017-01-20 11:55:00	"We will follow two simple rules: BUY AMERICA...	0.0000	neutral
2017-01-20 11:58:00	"It is time to remember that...https://www.fa...	0.0000	neutral
2017-01-20 12:13:00	"TO ALL AMERICANS https://www.facebook.com/Do...	0.0000	neutral
2017-01-21 05:53:00	"A fantastic day and evening in Washington D....	0.8666	positive
2017-01-22 06:47:00	"Watched protests yesterday but was under the...	-0.8459	negative
2017-01-23 05:38:00	"Busy week planned with a heavy focus on jobs...	0.4939	positive
2017-01-24 05:11:00	"Will be meeting at 9:00 with top automobile ...	0.3382	positive
2017-01-24 10:58:00	"A photo delivered yesterday that will be dis...	0.4199	positive

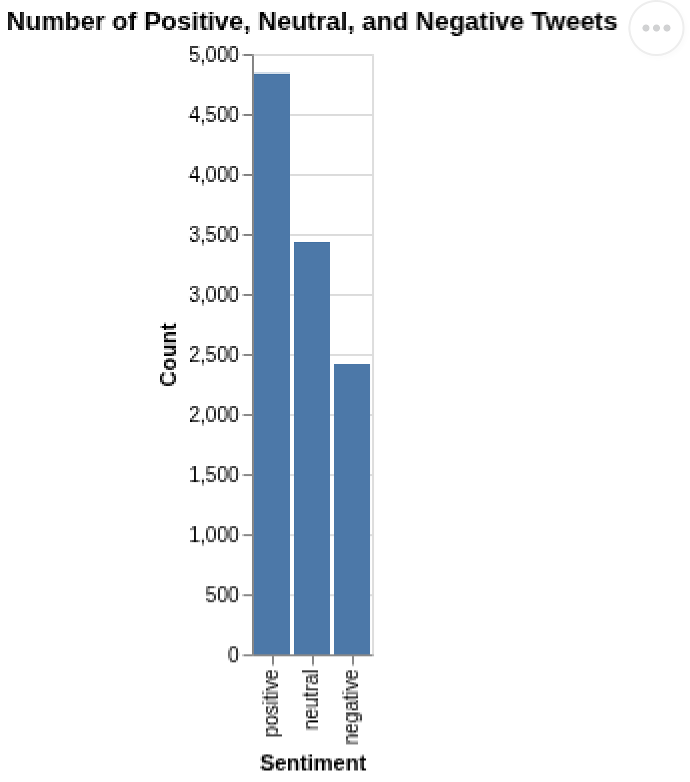
In [20]:
```
sentiment_counts = (
    tweets_q2
      .groupby("sentiment_label")
      .size()
      .reset_index(name="Count")
      .rename(columns={"sentiment_label": "Sentiment"})
)

sentiment_counts
```

Out[20]:

	Sentiment	Count
0	negative	2415
1	neutral	3433
2	positive	4836

In [21]:
```
sentiment_chart = (
    alt.Chart(sentiment_counts)
    .mark_bar()
    .encode(
        x=alt.X("Sentiment:N", sort=["positive", "neutral", "negative"]),
        y=alt.Y("Count:Q"),
        tooltip=["Sentiment", "Count"]
    )
    .properties(
        title="Number of Positive, Neutral, and Negative Tweets"
    )
)

sentiment_chart
```

Out[21]:



Question 2.2: How many tweets are positive vs negative? (Another method) +

Question 3: What are the most frequent words in the positive and negative tweets

Using a CountVectorizer and Logistic Regression, combining with a Wordcloud visualization. Syntax and bug fixing credited to ChatGPT 5.1

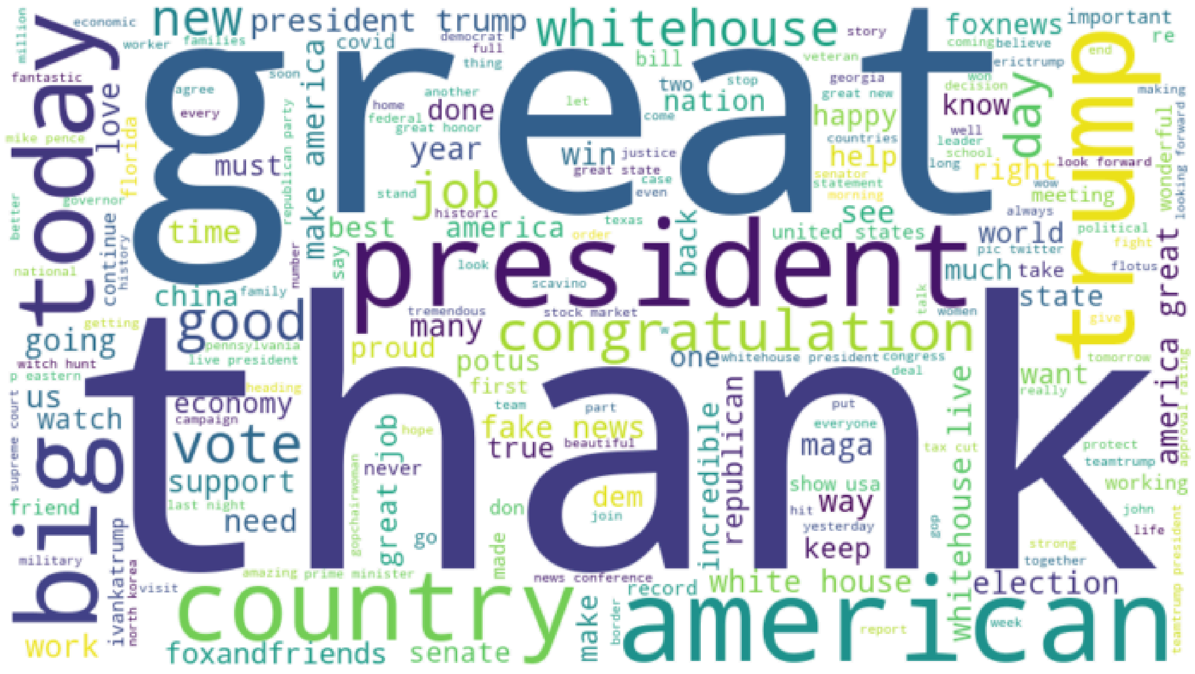
Q2 Method 2 & Q3: The second classifier method used to analyze the tweets was utilizing the CountVectorizer() to classify sets as positive words, negative words and some stop words unique to this dataset. Some The overall positive VS negative tweet counts are skewed in the opposite direction as the 'neutral' category here is somewhat added to the postively classified category. The results of the most frequent words are not very surprising to anyone who has followed American news at all in the last 10 years with the positive tweets containing words of praise and common phrases used by Trump, as well as his own name. The most frequent words appearing in the negative tweets could also be somewhat assumed as they contain words of the opposing political parties and leaders as well as other common phrases and words that one would hear often in a news report of Trump lashing out over twitter or a speech. A somewhat interesting note is that the word 'Trump' appears frequently in both positive and negative classifications, but could be rationalized by the fact that Trump often speaks about himself in the third person. A great visual indicator of these word frequencies is the word clouds for the respective most frequent positive and negatively classified words.

In [22]:
```python
import re
from collections import Counter
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

positive_words = set("""good great amazing fantastic tremendous strong win w
happy proud respect love best positive incredible honored grateful huge strc

negative_words = set("""bad terrible horrible weak fail failure disaster sac
worst negative unfair hate disgrace stupid dishonest democrat biden obama de

stopwords = set("""the a an and of to in is it this that for on with be as b
you your my our their they we i he she his her him them rt s all t just now

# ------------------------------------------------------------------
# 1. Helper functions
# ------------------------------------------------------------------

def simple_tokenize(text: str):
    """Lowercase everything, remove URLs and non letters, then split."""
    text = text.lower()
    text = re.sub(r"http\S+", "", text)          # remove URLs
    text = re.sub(r"[^a-z\s]", " ", text)        # keep letters and spaces
    return text.split()

def weak_label(text: str):
    """Use positive and negative lexicons to create weak labels for training
    if not isinstance(text, str):
        return None
    tokens = simple_tokenize(text)
    pos_hits = sum(1 for w in tokens if w in positive_words)
    neg_hits = sum(1 for w in tokens if w in negative_words)
    if pos_hits > neg_hits:
        return "positive"
    elif neg_hits > pos_hits:
        return "negative"
    else:
        return None  # ambiguous, skip from training

# ---------------------------------------------------------------------
# 2. Create weak labels for a subset of tweets
# ---------------------------------------------------------------------

tweets["weak_label"] = tweets["Tweet Text"].apply(weak_label)
train_df = tweets.dropna(subset=["weak_label"]).copy()

print("Number of weak labeled tweets:", len(train_df))
print(train_df["weak_label"].value_counts())

# ---------------------------------------------------------------------
# 3. Train a simple ML model using CountVectorizer
# ---------------------------------------------------------------------

X_train, X_test, y_train, y_test = train_test_split(
    train_df["Tweet Text"],
    train_df["weak_label"],
    test_size=0.2,
    random_state=42,
    stratify=train_df["weak_label"]
)

vectorizer = CountVectorizer(stop_words=list(stopwords), min_df=3)
X_train_vec = vectorizer.fit_transform(X_train.fillna(""))
X_test_vec = vectorizer.transform(X_test.fillna(""))

clf = LogisticRegression(max_iter=2000)
clf.fit(X_train_vec, y_train)

print("Validation accuracy (weak labels):", clf.score(X_test_vec, y_test))

# ---------------------------------------------------------------------
# 4. Predict sentiment for all tweets (positive vs negative)
# ---------------------------------------------------------------------

all_X = vectorizer.transform(tweets["Tweet Text"].fillna(""))
tweets["Sentiment"] = clf.predict(all_X)

print(tweets["Sentiment"].value_counts())

# ---------------------------------------------------------------------
# 5. Answer the question "What are the most frequent words in the positive a
# ---------------------------------------------------------------------

pos_text = " ".join(tweets[tweets["Sentiment"] == "positive"]["Tweet Text"].
neg_text = " ".join(tweets[tweets["Sentiment"] == "negative"]["Tweet Text"].

pos_tokens = [w for w in simple_tokenize(pos_text) if w not in stopwords]
neg_tokens = [w for w in simple_tokenize(neg_text) if w not in stopwords]

top_pos = Counter(pos_tokens).most_common(20)
top_neg = Counter(neg_tokens).most_common(20)

top_pos_df = pd.DataFrame(top_pos, columns=["word", "count"])
```

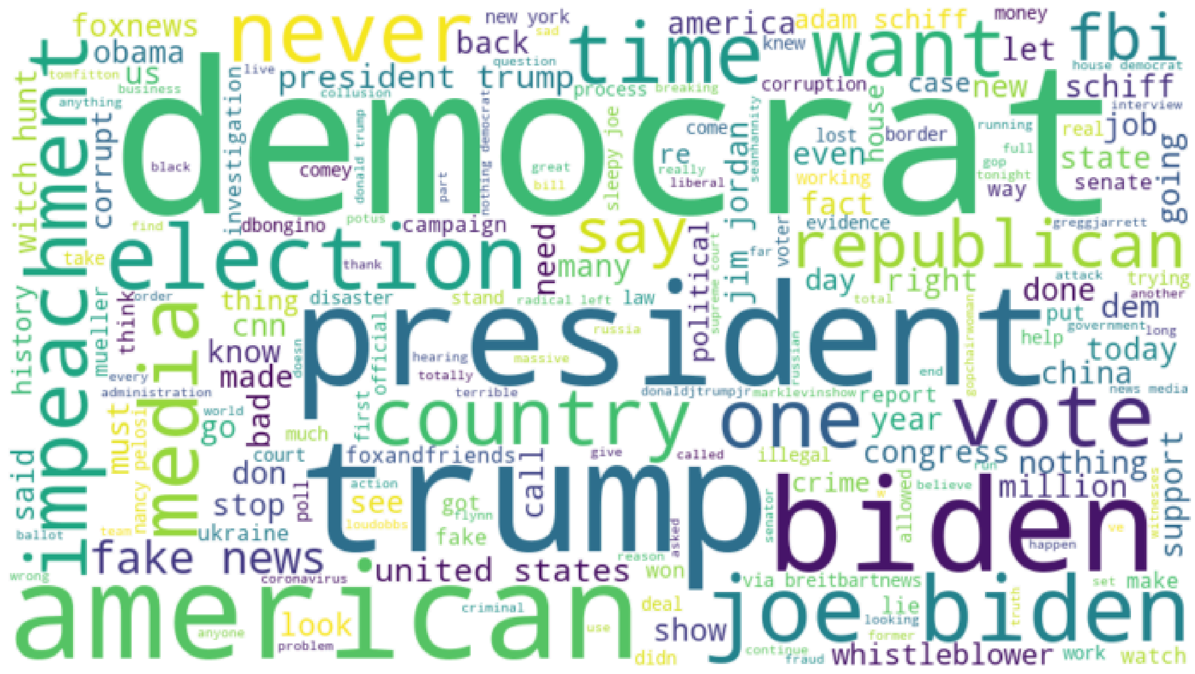
```
top_neg_df = pd.DataFrame(top_neg, columns=["word", "count"])

print("\nMost frequent words in positive tweets:")
print(top_pos_df)

print("\nMost frequent words in negative tweets:")
print(top_neg_df)

# --------------------------------------------------------------------
# 6. Wordcloud
# --------------------------------------------------------------------
pos_wc = WordCloud(
    width=900,
    height=500,
    background_color="white"
).generate(" ".join(pos_tokens))

plt.figure(figsize=(10, 5))
plt.imshow(pos_wc, interpolation="bilinear")
plt.axis("off")
plt.title("Positive Tweets Word Cloud")
plt.show()

neg_wc = WordCloud(
    width=900,
    height=500,
    background_color="white"
).generate(" ".join(neg_tokens))

plt.figure(figsize=(10, 5))
plt.imshow(neg_wc, interpolation="bilinear")
plt.axis("off")
plt.title("Negative Tweets Word Cloud")
plt.show()
```

```
Number of weak labeled tweets: 3527
weak_label
positive    2137
negative    1390
Name: count, dtype: int64
Validation accuracy (weak labels): 0.9985835694050992
Sentiment
positive    7817
negative    2867
Name: count, dtype: int64

Most frequent words in positive tweets:
          word  count
0        great   1216
1        thank    836
2    president    760
3        trump    547
4      america    392
5   whitehouse    371
6          big    325
7         news    309
8        today    297
9     american    264
10     country    234
11         new    227
12        maga    212
13          up    211
14        good    200
15        make    199
16         get    199
17       house    195
18     foxnews    180
19         day    179

Most frequent words in negative tweets:
           word  count
0     democrats    530
1         trump    468
2         biden    375
3     president    370
4            no    309
5           joe    228
6           out    203
7         media    201
8   impeachment    198
9         obama    182
10        there    177
11     election    175
12          did    162
13          why    162
14         than    161
15         fake    158
16         news    158
17         been    157
18          can    153
19        never    152
```

Positive Tweets Word Cloud



Negative Tweets Word Cloud



```
In [ ]:
```