Patterns and Influential Factors in Twitter Interactions among U.S. Congress Members

Data to Paper

December 31, 2023

Abstract

Understanding the dynamics of social interactions among members of Congress is crucial for analyzing political relationships and decision-making processes. However, there is limited knowledge about the patterns and factors influencing Twitter interactions among Congress members. This study presents a comprehensive analysis of Twitter interactions among members of the 117th US Congress, examining the role of party affiliation, chamber membership, and state representation. Our dataset, derived from a 4-month period, captures the Twitter activity of a diverse set of Congress members. Through regression analysis, we investigate the relationship between members' attributes and their incoming and outgoing interactions on Twitter. The results reveal intriguing insights into the influence of factors such as party affiliation and state representation on congressional Twitter interactions. While party affiliation shows some relationship with interaction patterns, the influence of state representation is less pronounced. These findings shed light on the social dynamics among members of Congress and underscore the potential of Twitter as a platform for political discourse and networking. However, it is important to note that this analysis focuses solely on Twitter interactions and does not account for offline or other online communication channels. Additionally, the dataset limitations and the inherently networked nature of Twitter should be considered when interpreting the findings. Overall, this research contributes to our understanding of social relationships within Congress and offers valuable insights for policymakers and political scientists studying online political behavior.

Results

To understand whether different factors influence Twitter interactions within members of the 117th US Congress, we performed a regression analysis on the key attributes. Specifically, we investigated the role of party affiliation, chamber membership, and state representation in predicting the number of incoming and outgoing interactions on Twitter.

Initially, we assessed the dependence of incoming Twitter interactions on political party affiliation, chamber of service, and the number of representatives from the same state. The results of this analysis are summarized in Table 1. The regression model shows an intercept value of 29.5 with a p-value smaller than 10^{-6}, indicating a baseline level of interactions. Surprisingly, the model does not provide any substantial evidence to support that party affiliation or the number of representatives from the same state significantly influence the number of incoming Twitter interactions. In contrast, chamber membership shows a considerable negative influence on incoming interactions (p-value = 0.00455), with senators receiving fewer interactions than House members.

Table 1: Regression results for variables predicting incoming interactions

	Beta	p-value
Intercept	29.5	$<10^{-6}$
Republican Party	0.848	0.678
Senate	-7.5	0.00455
State Rep. Count	-0.027	0.728

Republican Party: Membership in Republican Party, 1: Yes, 0: No
Senate: Membership in Senate, 1: Yes, 0: No
State Rep. Count: Number of Representatives from the Same State
Beta: Regression Coefficient

Building upon the analysis of incoming interactions, we next explored whether the same factors influence the outgoing interactions on Twitter. We used a similar regression model, and the results are presented in Table 2. The intercept of this model is 24.2 with a p-value smaller than 10^{-6}, suggesting a base level of outgoing interactions. The membership in the Republican party, and chamber did not statistically significantly influence the number of outgoing interactions. However, state representation (measured by the count of representatives from the same state) positively affects the number of outgoing interactions, with a substantial p-value of 0.0113.

In summary, these regression analyses suggest that while party affiliation is not a significant predictor of Twitter interactions among Congressional members, chamber membership influences incoming interactions and state representation impacts outgoing interactions. Nevertheless, the explana-

Table 2: Regression results for variables predicting outgoing interactions

	Beta	p-value
Intercept	24.2	$<10^{-6}$
Republican Party	0.824	0.629
Senate	3.2	0.146
State Rep. Count	0.165	0.0113

Republican Party: Membership in Republican Party, 1: Yes, 0: No
Senate: Membership in Senate, 1: Yes, 0: No
State Rep. Count: Number of Representatives from the Same State
Beta: Regression Coefficient

tory power of these models is relatively minor, with an R-squared value of 0.01781 for incoming interactions and 0.01491 for outgoing interactions. These modest values suggest the existence of additional elements that were not accounted for in the current model but determine Twitter interactions among Congress members.

A Data Description

Here is the data description, as provided by the user:

```
* Rationale:
The dataset maps US Congress's Twitter interactions into a directed graph with
    social interactions (edges) among Congress members (nodes). Each member (node)
    is further characterized by three attributes: Represented State, Political
    Party, and Chamber, allowing analysis of the adjacency matrix structure, graph
    metrics and likelihood of interactions across these attributes.

* Data Collection and Network Construction:
Twitter data of members of the 117th US Congress, from both the House and the
    Senate, were harvested for a 4-month period, February 9 to June 9, 2022 (using
    the Twitter API). Members with fewer than 100 tweets were excluded from the
    network.

- `Nodes`. Nodes represent Congress members. Each node is designated an integer
    node ID (0, 1, 2, ...) which corresponds to a row in `congress_members.csv`,
    providing the member's Represented State, Political Party, and Chamber.

- `Edges`. A directed edge from node i to node j indicates that member i engaged
    with member j on Twitter at least once during the 4-month data-collection
    period. An engagement is defined as a tweet by member i that mentions member j's
    handle, or as retweets, quote tweets, or replies of i to a tweet by member j.


* Data analysis guidelines:
- Your analysis code should NOT create tables that include names of Congress
    members, or their Twitter handles.
- Your analysis code should NOT create tables that include names of States, or
    their two-letter abbreviations. The code may of course do statistical analysis
    of *properties* related to States, but should not single out specific states.


2 data files:
File #1: "congress_members.csv"
A csv file of members of the 117th Congress, including their Twitter handles,
    Represented State, Party, and Chamber.
Data source: `https://pressgallery.house.gov/member-data/members-official-
```

twitter-handles`.
Rows are ordered according to the node ID, starting at 0.

Fields:

`Handle`: Twitter handle (without `@`)
`State`: Categorical; Two-letter state abbreviation; including also: "DC", "PR", "VI", "AS", "GU", "MP".
`Party`: Categorical; Party affiliation ("D", "R", or "I")
`Chamber`: Categorical; The member's chamber ("House", "Senate")

Here are the first few lines of the file:
```output
Handle,State,Party,Chamber
SenatorBaldwin,WI,D,Senate
SenJohnBarrasso,WY,R,Senate
SenatorBennet,CO,D,Senate

```

File #2: "congress_edges.dat"
This file provides the interaction network between members of the 115th US Congress on Twitter.
Download and adapted from: `https://snap.stanford.edu/data/congress-twitter`

Each line contains two integers (i, j), indicating a directed edge from node ID i to node ID j, compatible with nx.read_edgelist('congress_edges.dat', create_using=nx.DiGraph()). An i->j edge indicates that Congress member i had at least one tweet engaging with Congress member j during the 4-month collection period.

B Data Exploration

B.1 Code

The Data Exploration was carried out using the following custom code:

```python

import pandas as pd
import numpy as np
import scipy
import networkx as nx

# Load data
members_df = pd.read_csv("congress_members.csv")
edges_df = pd.read_table("congress_edges.dat", delimiter=" ", names=("Source", "Target"))

# Open output txt file
file = open("data_exploration.txt", "w")

# Data size
file.write("# Data Size\n")
file.write(f"Rows in congress_members: {len(members_df)}\n")
file.write(f"Columns in congress_members: {len(members_df.columns)}\n")
file.write(f"Rows in congress_edges: {len(edges_df)}\n")
file.write(f"Columns in congress_edges: {len(edges_df.columns)}\n")

# Summary statistics - for now we can only do this for the 'edges' data
file.write("# Summary Statistics\n")
description = edges_df.describe()
file.write(f"{description}\n")

# Categorical Variables - assume 'State', 'Party', 'Chamber' are the only categorical variables
file.write("# Categorical Variables\n")
for column in ['State', 'Party', 'Chamber']:
    file.write(f"{column} values and counts: {members_df[column].value_counts().to_string()}\n")
```

```python

# Checking for Missing Values
file.write("# Missing Values\n")
file.write(f"Missing values in congress_members: {members_df.isnull().sum().to_string()}\n")
file.write(f"Missing values in congress_edges: {edges_df.isnull().sum().to_string()}\n")

# Closing the file
file.close()

```

B.2 Code Description

The provided code performs data exploration on the original dataset consisting of members of the 117th US Congress and their Twitter interactions.

First, the code loads the data from two files: "congress_members.csv" containing information about the members of Congress, and "congress_edges.dat" specifying the directed edges representing Twitter interactions between the members.

Next, the code opens an output file named "data_exploration.txt" to write the results of the data exploration.

The code starts by reporting the data size, including the number of rows and columns in the "congress_members" and "congress_edges" dataframes.

Then, the code computes summary statistics for the "congress_edges" dataframe using the describe() function and writes the results to the output file.

Next, the code analyzes categorical variables, which in this case are the "State", "Party", and "Chamber" attributes of the members. It computes the occurrence count of each unique value for these variables using the value_counts() function and writes the results to the output file.

The code also checks for missing values in both the "congress_members" and "congress_edges" dataframes and reports the number of missing values for each column.

Finally, the output file is closed, completing the data exploration process.

In summary, the code performs various analyses on the dataset, including data size calculations, summary statistics, analysis of categorical variables, and checks for missing values. The results of these analyses are written to the "data_exploration.txt" file.

B.3 Code Output

data_exploration.txt

```
# Data Size
Rows in congress_members: 475
Columns in congress_members: 4
Rows in congress_edges: 13289
Columns in congress_edges: 2
# Summary Statistics
       Source  Target
count   13289   13289
mean    237.1   241.1
std     137.8     132
min         0       0
25%       115     132
50%       241     242
75%       359     354
max       474     474
# Categorical Variables
State values and counts: State
CA    49
TX    32
NY    27
FL    26
IL    17
PA    16
GA    15
OH    15
NC    13
VA    13
MI    13
NJ    12
AZ    11
WA    11
SC    10
MN     9
WI     9
MO     9
MD     9
```

```
TN     9
MA     9
AL     8
IN     8
LA     8
CO     8
OK     7
CT     7
IA     6
WV     6
KY     6
UT     6
AR     5
NM     5
MS     5
OR     5
NV     5
KS     4
NH     4
ID     4
VT     3
ME     3
WY     3
NE     3
MT     3
RI     3
ND     3
DE     3
HI     2
AK     2
SD     2
PR     1
DC     1
VI     1
MP     1
Party values and counts: Party
D    250
R    223
I      2
Chamber values and counts: Chamber
```

```
House     383
Senate     92
# Missing Values
Missing values in congress_members: Handle     0
State      0
Party      0
Chamber    0
Missing values in congress_edges: Source    0
Target    0
```

C Data Analysis

C.1 Code

The Data Analysis was carried out using the following custom code:

```python

# IMPORT
import pandas as pd
import networkx as nx
import numpy as np
from statsmodels.formula.api import ols
import pickle

# LOAD DATA
df_members = pd.read_csv('congress_members.csv')
edges = nx.read_edgelist('congress_edges.dat', create_using=nx.DiGraph(), nodetype=int)

# DATASET PREPARATIONS
state_counts = df_members['State'].value_counts()
df_members['state_rep_count'] = df_members['State'].map(state_counts)

# Use the node of the edge as the index to match with the Congress member DataFrame
in_degree = dict(edges.in_degree())
out_degree = dict(edges.out_degree())
```

```python
df_members['in_interactions'] = df_members.index.to_series().map(in_degree)
df_members['out_interactions'] = df_members.index.to_series().map(out_degree)

# DESCRIPTIVE STATISTICS
# No descriptive statistics table is needed.

# PREPROCESSING
df_members = pd.get_dummies(df_members, columns=['Party', 'Chamber'], drop_first=True)

# ANALYSIS
formula1 = 'in_interactions ~ state_rep_count + Party_R + Chamber_Senate'
model1 = ols(formula1, data=df_members.fillna(0)).fit()
df1 = pd.DataFrame({'Beta': model1.params, 'p-value': model1.pvalues})
df1.to_pickle('table_1.pkl')

# New table based on 'out_interactions'
formula2 = 'out_interactions ~ state_rep_count + Party_R + Chamber_Senate'
model2 = ols(formula2, data=df_members.fillna(0)).fit()
df2 = pd.DataFrame({'Beta': model2.params, 'p-value': model2.pvalues})
df2.to_pickle('table_2.pkl')

# SAVE ADDITIONAL RESULTS
additional_results = {
 'Total number of observations': len(df_members),
 'R-squared of regression model (in_interactions)': model1.rsquared,
 'R-squared of regression model (out_interactions)': model2.rsquared
}
with open('additional_results.pkl', 'wb') as f:
 pickle.dump(additional_results, f)
```

C.2 Code Description

The provided code performs data analysis on a dataset of the Twitter interactions among members of the 117th US Congress. The main steps of the analysis are as follows:

1. Loading the Data: - The code imports the necessary libraries, including pandas, networkx, numpy, and statsmodels. - It reads two data files: "congress_members.csv" containing information about the Congress members, and "congress_edges.dat" containing the directed edges indicating Twitter interactions.

2. Dataset Preparations: - The code calculates the count of Congress members from each state and adds it as a new column called "state_rep_count" to the DataFrame. - It calculates the in-degree and out-degree of each node in the graph and adds them as new columns called "in_interactions" and "out_interactions" respectively to the DataFrame.

3. Preprocessing: - The code uses one-hot encoding to convert categorical variables, such as "Party" and "Chamber", into binary variables.

4. Analysis: - The code performs linear regression analysis to examine the relationship between the number of in-interactions of a Congress member and the number of representatives from their state, their party affiliation (Republican), and their chamber (Senate). - The results of the regression analysis, including the coefficient estimates (Beta) and p-values, are stored in a pandas DataFrame and saved as a pickle file named "table_1.pkl".

5. Additional Results: - The code performs another linear regression analysis to examine the relationship between the number of out-interactions of a Congress member and the number of representatives from their state, their party affiliation (Republican), and their chamber (Senate). - Similar to the previous step, the results are stored in a pandas DataFrame and saved as a pickle file named "table_2.pkl". - The code also calculates additional results, including the total number of observations and the R-squared values of the regression models for in-interactions and out-interactions. These results are stored in a dictionary named "additional_results" and saved as a pickle file named "additional_results.pkl".

The "additional_results.pkl" file provides a summary of the analysis by including the total number of observations and the R-squared values of the regression models. The R-squared values indicate the proportion of the variance in the dependent variable that can be explained by the independent variables in the model. These additional results can be used for further analysis or reporting.

C.3 Code Output

table_1.pkl

```
                            Beta    p-value
Intercept               29.490068  1.894e-35
Party_R[T.True]          0.848442     0.6777
Chamber_Senate[T.True]  -7.502576   0.004549
state_rep_count         -0.027006     0.7282
```

table_2.pkl

```
                            Beta    p-value
Intercept               24.170003  2.419e-34
Party_R[T.True]          0.823660     0.6293
Chamber_Senate[T.True]   3.198804     0.1465
state_rep_count          0.164986    0.01133
```

additional_results.pkl

```
{
    'Total number of observations': 475,
    'R-squared of regression model (in_interactions)': 0.01781            ,
    'R-squared of regression model (out_interactions)': 0.01491            ,
}
```

D LaTeX Table Design

D.1 Code

The LaTeX Table Design was carried out using the following custom code:

```

# IMPORT
import pandas as pd
from typing import Dict, Any, Tuple, Optional
from my_utils import to_latex_with_note, format_p_value,
↪  is_str_in_df, split_mapping, AbbrToNameDef

# PREPARATION FOR ALL TABLES
shared_mapping: AbbrToNameDef = {
```

```python
    'Party_R[T.True]': ('Republican Party', 'Membership in Republican Party, 1: Yes, 0: No'),
    'Chamber_Senate[T.True]': ('Senate', 'Membership in Senate, 1: Yes, 0: No'),
    'state_rep_count': ('State Rep. Count', 'Number of Representatives from the Same State'),
}

# TABLE 1:
df1 = pd.read_pickle('table_1.pkl')

# RENAME ROWS AND COLUMNS
mapping = {k: v for k, v in shared_mapping.items() if is_str_in_df(df1, k)}
mapping |= {
    'Intercept': ('Intercept', None),
    'Beta': ('Beta', 'Regression Coefficient')
}

abb_to_names, legend = split_mapping(mapping)
df1 = df1.rename(columns=abb_to_names, index=abb_to_names)

# FORMAT P-VALUES
df1['p-value'] = df1['p-value'].apply(format_p_value)

# Save as a LaTeX table:
to_latex_with_note(
    df1, 'table_1.tex',
    caption="Regression results for variables predicting incoming interactions",
    label='table:incoming_interactions',
    note=None,
    legend=legend)

# TABLE 2:
df2 = pd.read_pickle('table_2.pkl')

# RENAME ROWS AND COLUMNS
mapping = {k: v for k, v in shared_mapping.items() if is_str_in_df(df2, k)}
```

```python
mapping |= {
    'Intercept': ('Intercept', None),
    'Beta': ('Beta', 'Regression Coefficient'),
}

abb_to_names, legend = split_mapping(mapping)
df2 = df2.rename(columns=abb_to_names, index=abb_to_names)

# FORMAT P-VALUES
df2['p-value'] = df2['p-value'].apply(format_p_value)

# Save as Latex:
to_latex_with_note(
    df2, 'table_2.tex',
    caption="Regression results for variables predicting outgoing interactions",
    label='table:outgoing_interactions',
    note=None,
    legend=legend)

```

D.2 Provided Code

The code above is using the following provided functions:

```python
def to_latex_with_note(df, filename: str, caption: str, label: str, note: str = None, legend: Dict[str, str] = None, **kwargs):
    """
    Converts a DataFrame to a LaTeX table with optional note and legend added below the table.

    Parameters:
    - df, filename, caption, label: as in `df.to_latex`.
    - note (optional): Additional note below the table.
    - legend (optional): Dictionary mapping abbreviations to full names.
    - **kwargs: Additional arguments for `df.to_latex`.

    Returns:
```

```python
    - None: Outputs LaTeX file.
    """

def format_p_value(x):
    returns "{:.3g}".format(x) if x >= 1e-06 else "<1e-06"

def is_str_in_df(df: pd.DataFrame, s: str):
    return any(s in level for level in getattr(df.index, 'levels', [df.index]) + getattr(df.columns, 'levels', [df.columns]))

AbbrToNameDef = Dict[Any, Tuple[Optional[str], Optional[str]]]

def split_mapping(abbrs_to_names_and_definitions: AbbrToNameDef):
    abbrs_to_names = {abbr: name for abbr, (name, definition) in abbrs_to_names_and_definitions.items() if name is not None}
    names_to_definitions = {name or abbr: definition for abbr, (name, definition) in abbrs_to_names_and_definitions.items() if definition is not None}
    return abbrs_to_names, names_to_definitions

```

D.3 Code Output

table_1.tex

```latex
\begin{table}[h]
\caption{Regression results for variables predicting incoming interactions}
\label{table:incoming_interactions}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrl}
\toprule
 & Beta & p-value \\
\midrule
\textbf{Intercept} & 29.5 & $<$1e-06 \\
```

```
\textbf{Republican Party} & 0.848 & 0.678 \\
\textbf{Senate} & -7.5 & 0.00455 \\
\textbf{State Rep. Count} & -0.027 & 0.728 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Republican Party}: Membership in Republican Party, 1: Yes, 0: No
\item \textbf{Senate}: Membership in Senate, 1: Yes, 0: No
\item \textbf{State Rep. Count}: Number of Representatives from the Same State
\item \textbf{Beta}: Regression Coefficient
\end{tablenotes}
\end{threeparttable}
\end{table}
```

table_2.tex

```
\begin{table}[h]
\caption{Regression results for variables predicting outgoing interactions}
\label{table:outgoing_interactions}
\begin{threeparttable}
\renewcommand{\TPTminimum}{\linewidth}
\makebox[\linewidth]{%
\begin{tabular}{lrl}
\toprule
 & Beta & p-value \\
\midrule
\textbf{Intercept} & 24.2 & $<$1e-06 \\
\textbf{Republican Party} & 0.824 & 0.629 \\
\textbf{Senate} & 3.2 & 0.146 \\
\textbf{State Rep. Count} & 0.165 & 0.0113 \\
\bottomrule
\end{tabular}}
\begin{tablenotes}
\footnotesize
\item \textbf{Republican Party}: Membership in Republican Party, 1: Yes, 0: No
\item \textbf{Senate}: Membership in Senate, 1: Yes, 0: No
\item \textbf{State Rep. Count}: Number of Representatives from the Same State
\item \textbf{Beta}: Regression Coefficient
```

```
\end{tablenotes}
\end{threeparttable}
\end{table}
```