Machine Learning for Legislative Forecasting: Predicting which bills will pass Congress

Jett Carruth, Claudia Moses

Datasci 112 Winter 2024

Abstract

We used predictive modeling to try and measure which bills presented before the 117th U.S. Congress would become law. We used a variety of methods to collect data from government databases, used merger features to gather them into a single dataframe, and conducted an exploration into the text of the data to learn more about the bills. We fit a regression model to predict which bills would become law, and had a 72% test precision and 83% test recall.

Sources

GovInfo

govtrack.us

CONGRESS.GOV

Stanford University

Research Question

We wanted to see if we could use existing legislation to predict which bill introduced on the floor of congress would become law.

Data Collection

While the information we needed was relatively easy to find online, it turned out to be difficult to collect into a dataframe. We used a lot of new methods for collection including:

1) Simulated human page scrolling to prep for scraping using Selenium

```
import os
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

os.environ['PATH'] += ':/usr/bin/chromedriver'
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome(options=chrome_options)



# Load the webpage
url = "https://www.govtrack.us/congress/bills/browse?congress=118#congress=117&current_status[]=1"
driver.get(url)

member_bills = {}
# Scroll down to the bottom of the page
scrolls = 456  # Adjust the number of scrolls as needed
for _ in range(scrolls):
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(1)  # Adjust the wait time as needed

# Extract the HTML content
html = driver.page_source

# Parse the HTML with BeautifulSoup
soup = BeautifulSoup(html, 'html.parser')
```

2) Parsing XML from zip files

```
from xml.etree import ElementTree as ET
import os

import zipfile
import pandas as pd
import io
import re

# Function to extract text from XML
def extract_text_from_xml(xml_content):
    root = ET.fromstring(xml_content)
    paragraphs = root.findall('.//text')
    return ' '.join(p.text for p in paragraphs if p.text)

# Function to extract bill name from filename
def extract_bill_name_hr(filename):
    hr_number = re.search(r'hr(\d+)', filename)
    if hr_number:
        return "H.R." + hr_number.group(1)

#Function to read zip files
def fill_hr_df(zip_file_path, sessionNumber):
  data = []
  with zipfile.ZipFile(zip_file_path) as zip_file:
      for file_name in zip_file.namelist():
          if file_name.endswith('.xml'):
              with zip_file.open(file_name) as xml_file:
                  xml_content = xml_file.read()
                  text = extract_text_from_xml(xml_content)
                  bill_name = extract_bill_name_hr(file_name)
                  data.append({'filename': file_name, 'bill_name': bill_name,
                               'session': sessionNumber, 'text': text})

  df = pd.DataFrame(data)
  return df
```

Data Exploration

Once we'd gathered all the information we needed, we were able to assemble a dataframe with all bill numbers, text, congressmen sponsors, text length, and associated political parties.

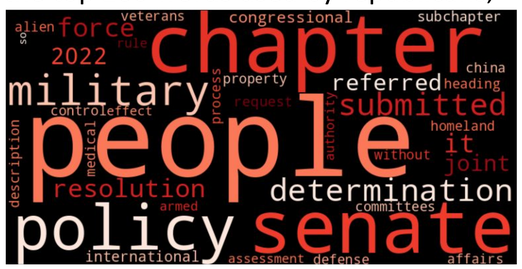
	filename	bill_name	session	text	member	Party	Passed	text_length
1	BILLS-117s11pcs.xml	S.11	Session 1	Notwithstanding the second sentence of section...	Charles "Chuck" Schumer	D	0	666.0
2	BILLS-117s13pcs.xml	S.13	Session 1	There is established within the Election Assis...	Tim Scott	R	0	6067.0
5	BILLS-117s45is.xml	S.45	Session 1	This Act may be cited as the Section 4104 of ...	Ted Cruz	R	0	2106.0
7	BILLS-117s72is.xml	S.72	Session 1	This Act may be cited as the Congress finds t...	Chris Van Hollen	D	0	8107.0
9	BILLS-117s59is.xml	S.59	Session 1	This Act may be cited as the In this Act: Exc...	Thom Tillis	R	0	4618.0
...	...	...	...	...	...	...	...	...
18120	BILLS-117hr9692ih.xml	H.R.9692	Session 2	This Act may be cited as the On the date that...	Louie Gohmert	R	0	2540.0
18121	BILLS-117hr9624ih.xml	H.R.9624	Session 2	This Act may be cited as the The table of con...	Henry C. "Hank" Johnson	D	0	123902.0
18122	BILLS-117hr9623ih.xml	H.R.9623	Session 2	This Act may be cited as the The table of con...	Pramila Jayapal	D	0	407095.0
18123	BILLS-117hr9673ih.xml	H.R.9673	Session 2	Congress finds the following: Oral history fro...	Donald Beyer	D	0	1781.0
18124	BILLS-117hr9553ih.xml	H.R.9553	Session 2	This Act may be cited as the by repealing sec...	Pramila Jayapal	D	0	203.0

13767 rows × 8 columns

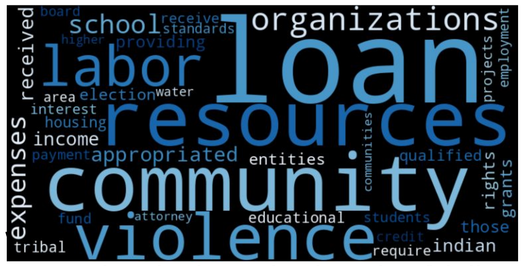
Only about 1% of the bills became law!

We wanted to visualize the differences between bills proposed by democrats and republicans.

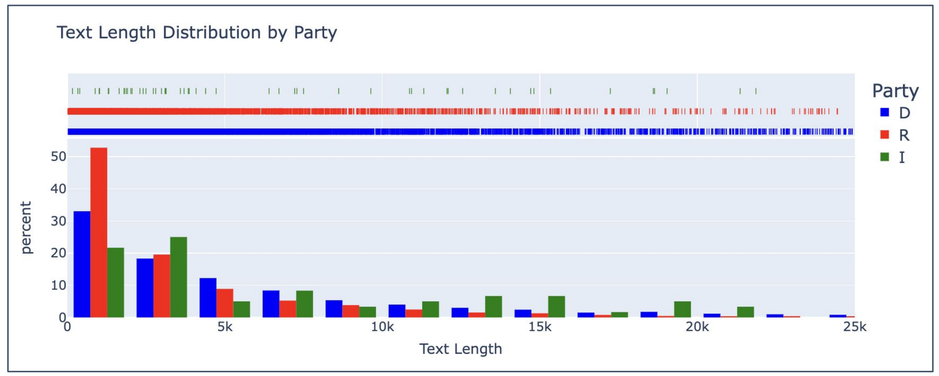
Below are the terms uniquely among the top 200 words used by republicans,



and democrats in their bills.



We took a look at the length of bills and found that on average, democrats' bills were longer than republicans'.



Data Analysis

We first tried to fit a k-neighbors regressor using a machine learning method for **imbalanced data: downsampling and upweighting**, to train our model. It turned out a classic logistic regressor was a better model:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.compose import make_column_transformer
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

warnings.filterwarnings("ignore")

pipeline = make_pipeline(
    make_column_transformer(
        (TfidfVectorizer(smooth_idf=False, norm=None), 'text'),
        (OneHotEncoder(), ['Party']),
        remainder ='passthrough'
    ),
    LogisticRegression()
)

scores = cross_val_score(
pipeline,
X=df[['text', "Party"]],
y=df["Passed"], # this is all of the training data!
scoring="neg_mean_squared_error", # higher is better for a score
cv=4)

#np.sqrt(scores.mean)
print(scores)
```

```
print( np.sqrt(-1 * scores))

[0.26515648 0.21083382 0.18198978        nan]
```

This was the error calculated on 4 folds of training data. We decided to calculate the test error by training the model on a random part of our data and testing it on the rest. We created a confusion matrix and calculated the recall and precision of our predictions.

```
conf_matrix = confusion_matrix(y_test, y_test_)
conf_matrix

array([[ 122,   25],
       [  47, 2560]])
```

We ended up with a pleasant surprise: our precision was 72%, and our recall was 83%! Our model accurately predicted 83% of the bills in the test data that were signed into law, and 72% of our predicted passed laws were actually enacted.

Conclusion

We were very pleased with the results of our model, which predicted which bills were enacted with a much higher probability of success than we initially hypothesized. This is important, as it could help us learn what types of factors are most important in deciding which bills become law- from party splits to language in the bills.

Some further considerations include:

- What factor in the pipeline held the most weight?
- How can we use this to optimize bills?