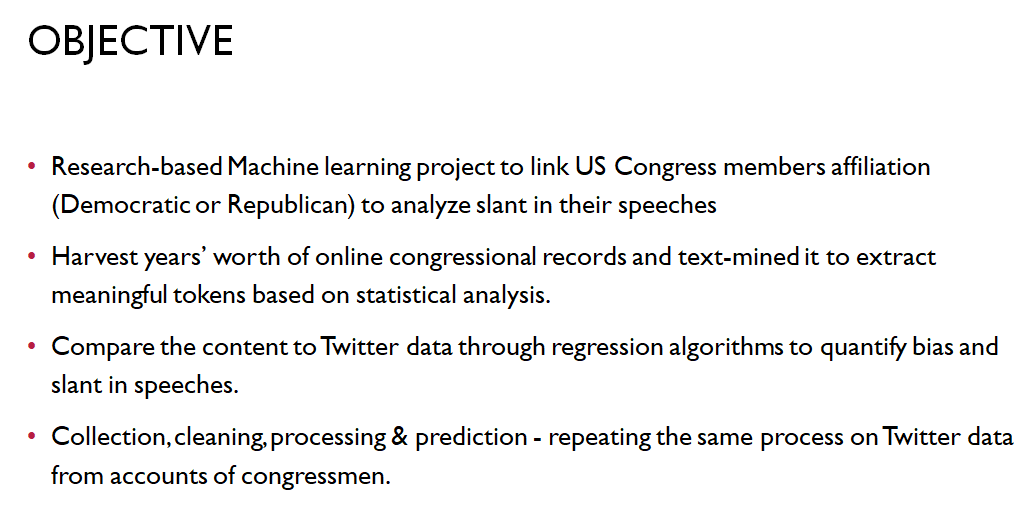
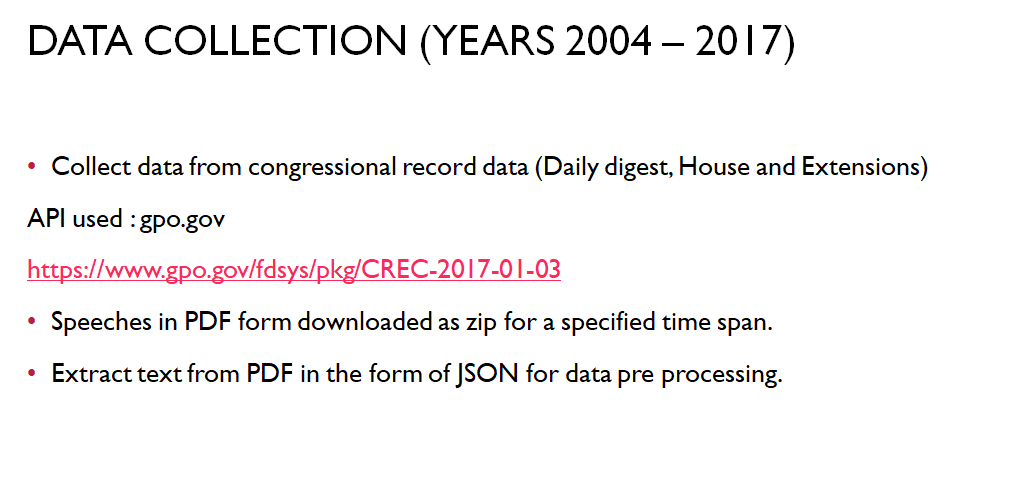
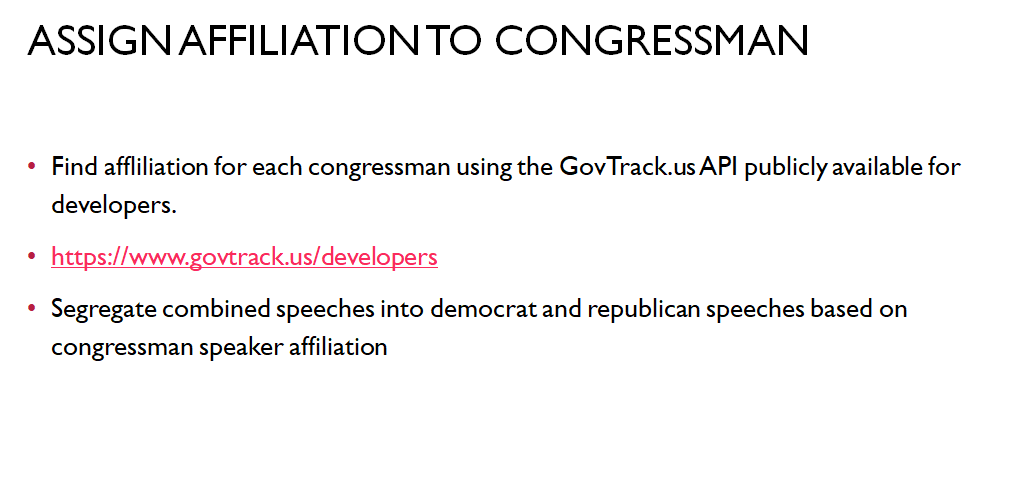


Congressional speech analysis







# **Environment set up instructions**

# 1. Install Python on Ubuntu

sudo apt-get install python

2. Install this library to use the buildins library of python

sudo apt install python-future

3. Use this command to goto the root user level for installing pyopenssl

sudo -i

Install openssl for python using this command

sudo pip install pyopenssl

4. Use this command to install the elastic search library for python

pip install pyelasticsearch

5. Install the Unicode csv library

pip install unicodecsv

6. Execute this command to open the tool’s build in help menu:

python -m congressionalrecord.cli -h

7. This command was used to extract data from 1st Jan to 20th Jan 2017 in JSON Format.

python -m congressionalrecord.cli 2017-01-01 2017-01-20 'json'

Gov website url for files :

<https://www.gpo.gov/fdsys/pkg/CREC-2017-01-03>

Data size downloaded : **7.91 GB (~ 8 GB)**

Average program run time : **~ 26 hours +**

Data size after JSON extraction from PDF downloaded, text stemming and data cleaning : 82.6 MB (~ 83 MB)

Number of comparisons :

~ 40 MB [Individual congressmen speeches] for every 2 grams x ~ 40 MB [ Consolidated congressmen speeches] for every 1000 2 grams

# 

# **NEXT STEPS**

* Pre-processing:
  + Exclude speech by officers (e.g. the Clerk, Speaker of the House, President of the Senate).
  + Exclude block quotations, text that is inserted into the Record from other sources such as reports or letters, and nonspeech items like records of roll-call votes.
* Pre-processing:
  + Removing stop words using
    - the Fox 1990 paper – list of words on p. 27-34 of the attached paper.
    - days of the week, Hart Senate Office Building, and Dirksen Senate Office Building
    - names of major newspapers (this page has the top 100 in 2015 - <http://247wallst.com/media/2017/01/24/americas-100-largest-newspapers/>; this page has the top 100 in 2007 - <https://www.infoplease.com/arts-entertainment/newspapers-and-magazines/top-100-newspapers-united-states>)
  + Stemming using Porter stemmer
* The above program will download the speeches on a per year per day basis in the output folder in JSON format
* in **congressionalrecord - > fsds -> crparser.py**
* remove\_stop\_words function removes the custom stop words from the list stop\_words\_list.txt
* Custom code extracts individual speeches based on affiliation of the congressmen

for speech in self.crdoc['content']:

#print("Code running")

if speech['kind'] == 'speech':

#print(speech['text'])

if speech['speaker\_bioguide']:

keybioguideid = speech['speaker\_bioguide']

outpath = os.path.join('','json',keybioguideid+'.json')

with open(outpath) as json\_data:

d = json.load(json\_data)

if d['party']=='D':

print("D")

with open('democratic/'+speech['speaker']+'-'+str(keybioguideid)+'.txt','a+') as out\_json:

out\_json.write(str(self.remove\_stop\_words(speech['text']))+'\n')

os.chmod('democratic/'+speech['speaker']+'-'+str(keybioguideid)+'.txt', 0o777)

with open('democratic\_speeches.txt','a+') as out\_json:

out\_json.write(str(self.remove\_stop\_words(speech['text']))+'\n')

elif d['party'] =='R':

print("R")

with open('republican/'+speech['speaker']+'-'+str(keybioguideid)+'.txt','a+') as out\_json:

out\_json.write(str(self.remove\_stop\_words(speech['text']))+'\n')

os.chmod('republican/'+speech['speaker']+'-'+str(keybioguideid)+'.txt', 0o777)

with open('republican\_speeches.txt','a+') as out\_json:

out\_json.write(str(self.remove\_stop\_words(speech['text']))+'\n')

* This will extract individual speeches into the democratic and republican folder

# **DERIVING UNIQUE 2 GRAMS - 3 GRAMS**

the next step is to derive the following:

- Unique 2-grams and the frequency of each for republicans and democrats separately. For example, tax cut might be repeated 40 times with republicans and 3 times with democrats; this yields f jr = 40 and f jd = 3. Note that if these counts are mostly 1 then we have a problem that we need to address.

- For any given 2-gram, the frequency of occurrence of all the other 2-gram except the one we are currently handling. Say with the republicans, we are interested in the 2-gram "tax cut" and there are only 3 other 2-grams (of course there will be more) with the first one occurring with frequency 20, the second with frequency 10 and the third with frequency 5, then f ~jr = 20+10+5= 35. For the democrats, they have 4 other 2-grams with f ~jd = 20.

- This way, we have all the parameters of the chi square metric.

For any given 2-gram, the frequency of occurrence of all the other 2-gram except the one we are currently handling. Say with the republicans, we are interested in the 2-gram "tax cut" and there are only 3 other 2-grams (of course there will be more) with the first one occurring with frequency 20, the second with frequency 10 and the third with frequency 5, then f ~jr = 20+10+5= 35. For the democrats, they have 4 other 2-grams with f ~jd = 20.

**CODE:**

**For Congress : combined-metrics.py**

**For Twitter : combined-metrics-twitter.py**

def ngrams(inputtext, n):

inputtext = inputtext.split()

output = {}

for i in range(len(inputtext)-n+1):

#print(inputtext[i:i+n])

g = ' '.join(inputtext[i:i+n]).lower()

#print(g)

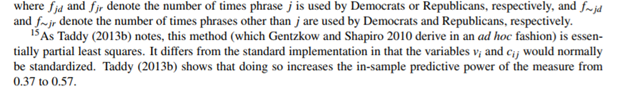
output.setdefault(g, 0)

output[g] += 1

return output

# **CHI SQUARE METRICS**





The main focus of the study is characterizing the incentives that drive newspapers’ choice of slant. With the estimated ˆvi in hand, the authors estimate a model of consumer demand in which a consumer’s utility from reading newspaper i depends on the distance between i’s slant vi and an ideal slant v ∗ which is greater the more conservative is the consumer’s ideology.

**CODE:**

**For Congress : combined-metrics.py**

**For Twitter : combined-metrics-twitter.py**

**chi\_square = float((fjr\*f\_jd - fjd\*f\_jr)/((fjr+fjd)\*(fjr+f\_jd)\*(f\_jr+fjd)\*(f\_jr+f\_jd)))**

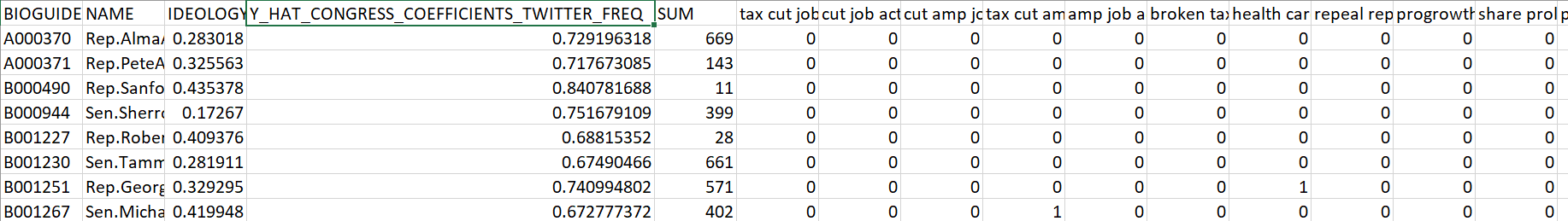
# **CALCULATING FREQUENCIES**

**Congress:**

**congressman-frequency.py**

**Twitter:**

**congressman-tweet-frequency.py**



# **MATCHING CONGRESSMEN NAME TO IDEOLOGY**

Python code:

**congressman-ideology.py**

Ideology list for senators and congressmen:

<https://www.govtrack.us/congress/members/report-cards/2017/house/ideology>

[https://www.govtrack.us/congress/members/report-cards/2017/senate/ideology](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.govtrack.us%2Fcongress%2Fmembers%2Freport-cards%2F2017%2Fsenate%2Fideology&data=02%7C01%7Cwjabr%40gsu.edu%7C15d364e8cee14dc18b3108d57d36a419%7C515ad73d8d5e4169895c9789dc742a70%7C0%7C0%7C636552595995817608&sdata=3S09nWrWkQyXcJDXa5cwLUIa1qpUKaRRICSMeZD3qcA%3D&reserved=0).

# **RUNNING THE REGRESSION**

**Python code:**

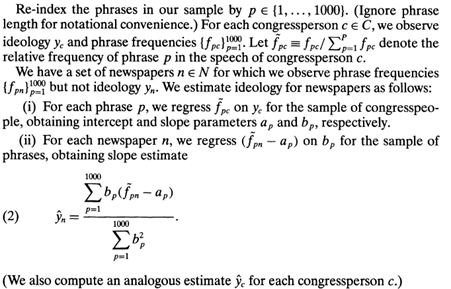
**Congress:**

**regression.py**

**Twitter:**

**regression-tweets.py**

**Mapping Phrases to Ideology**



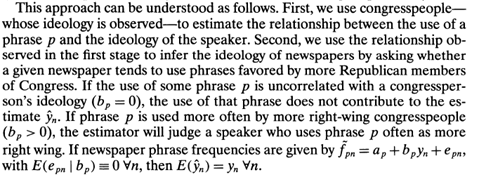
For each phrase, identify

a) fpc = frequency of phrase p (in top 1000) in congressperson c

b) ~fpc = fpc / Σp fpc

c) For each phrase p, ~fpc = ap + bp xc + єp ; where ap and bp obtained by OLS & xc congressperson affiliation

[https://www.datarobot.com/blog/ordinary-least-squares-in-python/ ]



# 

**Flow- diagram for congress speech -**

**1 -** Download required year’s speech\_data

**2-** Generate Chi-square metrics

|  |  |  |
| --- | --- | --- |
| **INPUT** | **PROCESS** | **OUTPUT** |
| dem\_speech.txt | CHI-SQUARE | chi\_square\_metrics |
| rep\_speech.txt |

dem\_speech.txt & rep\_speech.txt generated in step1

**3-** Select required tokens -> congress\_grams\_csv

**4-** Generate frequency metrics for members separately-

|  |  |  |
| --- | --- | --- |
| txt files of members speech | frequency | frequency\_metrics |
| congress\_grams |

Input files are from step 1 & step 3

**5-** Download ideology score.csv file for required year

**6-** Run the regression

|  |  |  |
| --- | --- | --- |
| ideology\_csv | regression | regression\_output |
| frequency\_csv |

Input files are from step 4 & step 5

token, no of observations, coefX, coef Const, err X, err Const, t-valX, t-valC, pvalX, pvalC

X - ideology

Y - fpc = fpc / Σp fpc

p1 - D:\Slant\Test\_Run\Test\_Run\_2005\_latest

Following steps are used to generate the output -

**Step1**-Selected 100 tokens from files (2gram=50 tokens & 3gram=50 tokens) placed at path - D:\Slant\Test\_Run\Chi\_square\_2005\Top20K\_combine

Output File at p1- congress-grams.csv

**Step2-**Generate frequency matrix for these 1000 tokens, separate for democrats & republican

Output File at p1 - democratic\_congress\_frequency\_individual.csv & republican\_congress\_frequency\_individual

**Step3**- Download sponsorshipanalysis\_h & sponsorshipanalysis\_s files for respective congress number (e.g. 109th congress is for year 2005 & 2006) from

<https://www.govtrack.us/data/us/109/stats/> - These file contains govtrack ID for members, need tom app to with bio-guide ID.

Download legislators-current.csv & legislators-historical.csv from <https://github.com/unitedstates/congress-legislators/>

Generate ideology.csv file by mapping govtrack\_id & Bioguide\_id using code in ideology\_BIOGUIDEID\_Mapp.py file (placed at path p1)

**Step4**- Run the regression

Output file - congress-grams-regression.csv

Note:

import sys, json, re

import us

import datetime # implicitly used in eval()'ing dates inside major\_actions

import django

from django.conf import settings

# Explicitly set DEBUG to False to avoid memory leak

settings.DEBUG = False

django.setup()