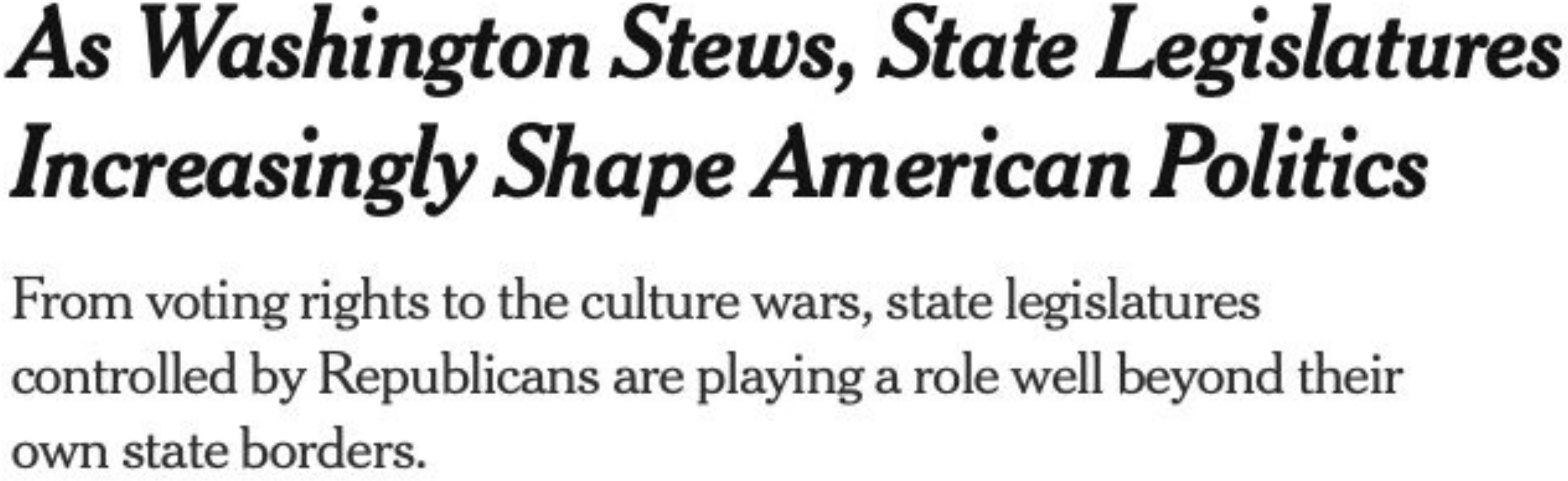
State Legiscraper

Katherine Chang, Shujie Chen, Gan Gao, Cynthia Wu

CSE 583, Fall 2021: December 15, 2021

The New York Times



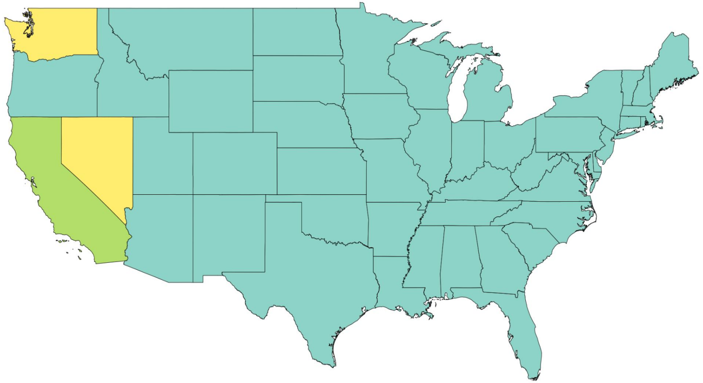
As Washington Stews, State Legislatures Increasingly Shape American Politics

From voting rights to the culture wars, state legislatures controlled by Republicans are playing a role well beyond their own state borders.

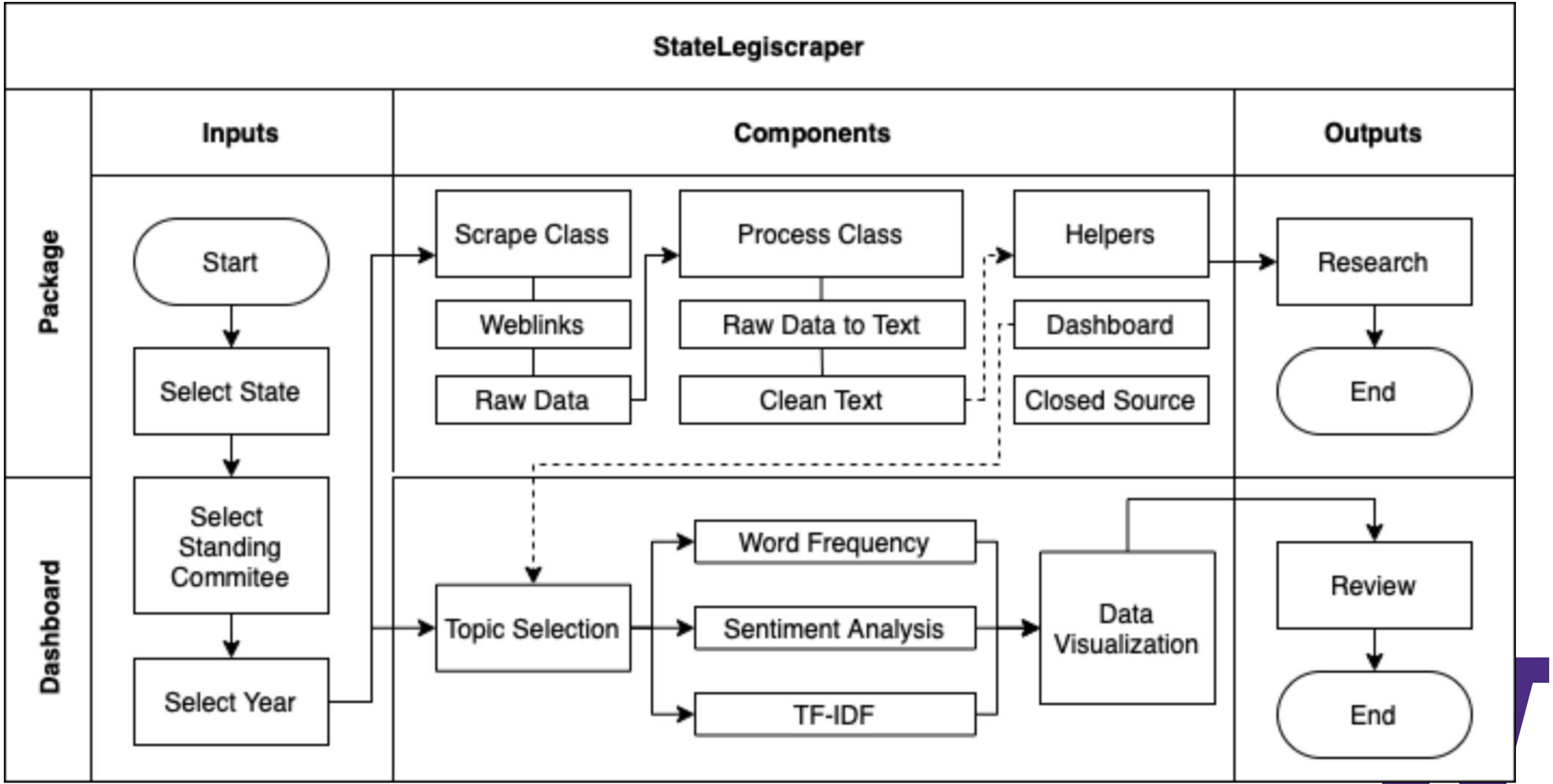
Source: https://www.nytimes.com/2021/08/29/us/state-legislatures-voting-gridlock.html

StateLegiscraper Mission

To make accessible text corpora of political, social, and scholarly significance that can build greater public transparency and academic knowledge about public policymaking and state-level politics.



Design: Schematics of Components



StateLegiscraper

Package

Dashboard

Inputs

Components

Outputs

Start

Select State

Select Standing Commitee

Select Year

Scrape Class

Weblinks

Raw Data

Process Class

Raw Data to Text

Clean Text

Helpers

Dashboard

Closed Source

Research

End

Topic Selection

Word Frequency

Sentiment Analysis

TF-IDF

Data Visualization

Review

End

Design: Scrape Class

Scrape:

- Functions that scrape websites for standing committee hearing PDF / audio / video transcript links
- Returns: Export raw data to user’s local drive or a mounted cloud drive.

Functions:

- [state]_scrape_weblinks
- [state]_scrape_[raw format]

Design: Process Class

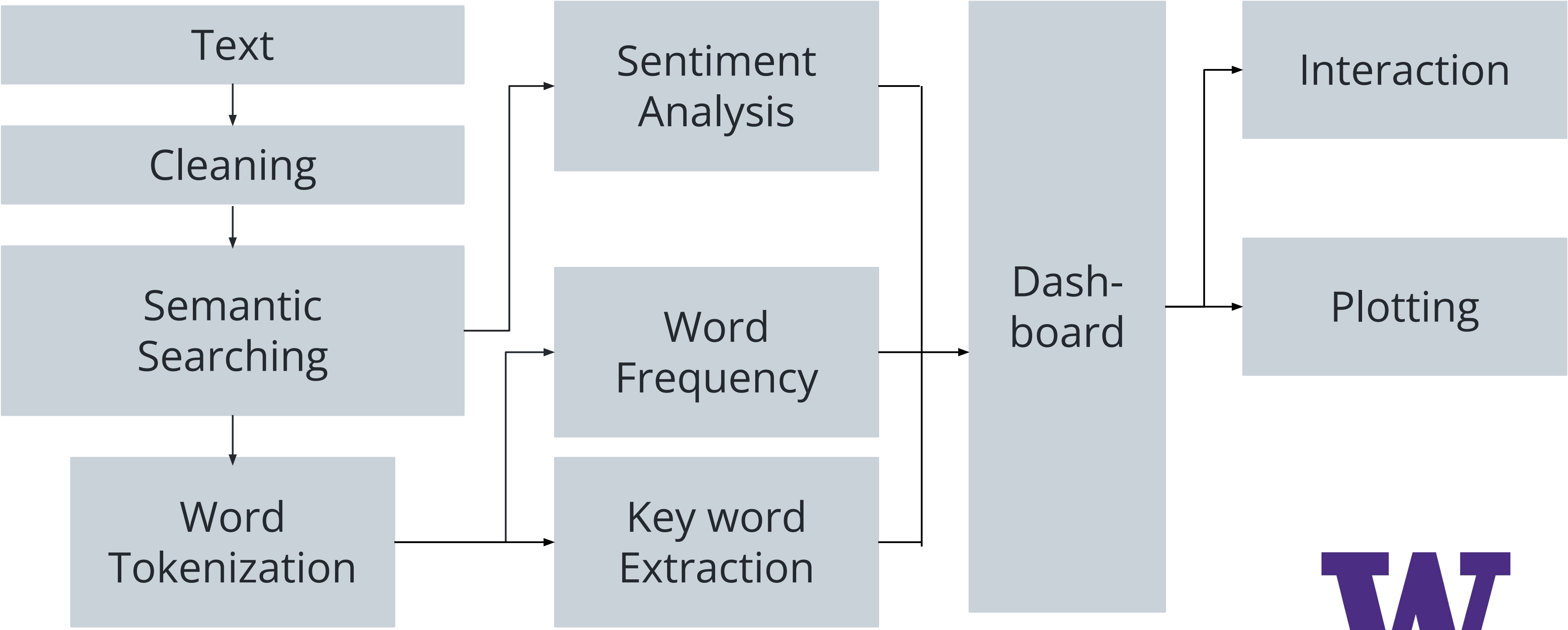
Process:

- Functions that cleans and formats the raw scraped data into Python objects appropriate to use for popular NLP packages.
- Returns: Cleaned transcripts as a dictionary object or exported JSON file.

Functions:

- [state]_[raw format]_to_text
- [state]_text_to_clean

Design: Text Analysis Workflow



Text

Cleaning

Semantic Searching

Word Tokenization

Sentiment Analysis

Word Frequency

Key word Extraction

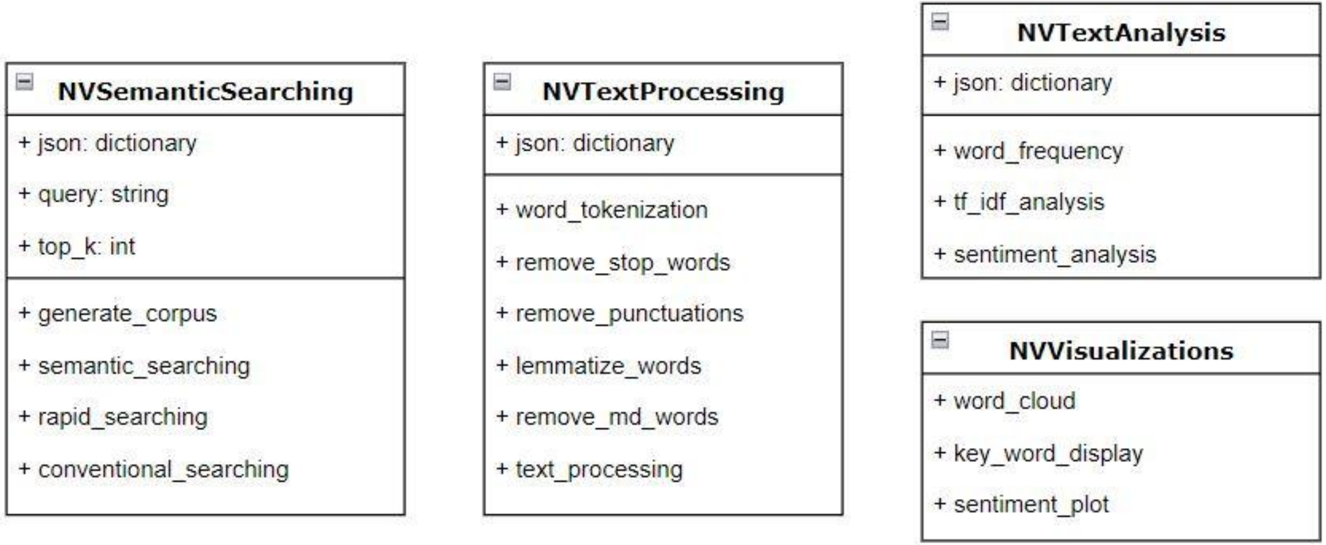
Dash-board

Interaction

Plotting

Design: Dashboard Helper

- dashboard_helper.py: text cleaning and analysis functions for the back end
  - SemanticSearching, Preprocessing, TextAnalysis, Visualization
- app.py: web visualization and interaction functions for the front end.



NVSemanticSearching

+ json: dictionary
+ query: string
+ top_k: int

+ generate_corpus
+ semantic_searching
+ rapid_searching
+ conventional_searching

NVTextProcessing

+ json: dictionary

+ word_tokenization
+ remove_stop_words
+ remove_punctuations
+ lemmatize_words
+ remove_md_words
+ text_processing

NVTextAnalysis

+ json: dictionary

+ word_frequency
+ tf_idf_analysis
+ sentiment_analysis

NVVisualizations

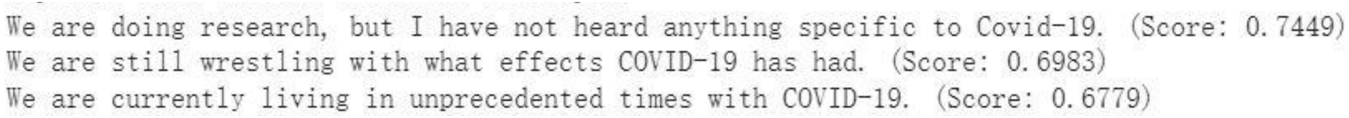
+ word_cloud
+ key_word_display
+ sentiment_plot

Technology: Semantic searching

	Open source	Accuracy	Speed
Sentence Transformer	√	√	√
Spapy	√	x	x
Openai	x	/	/

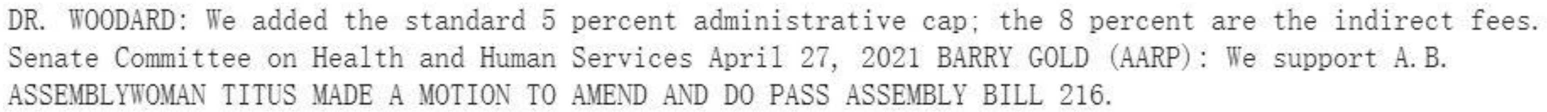
Top results for query COVID 19

Sentence Transformer:



```
We are doing research, but I have not heard anything specific to Covid-19. (Score: 0.7449)
We are still wrestling with what effects COVID-19 has had. (Score: 0.6983)
We are currently living in unprecedented times with COVID-19. (Score: 0.6779)
```

Spapy:

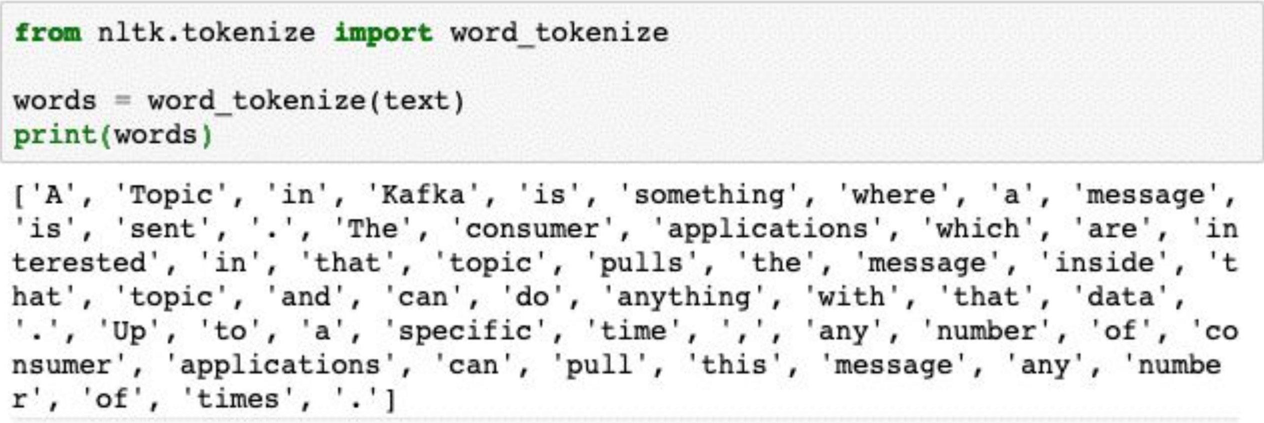


```
DR. WOODARD: We added the standard 5 percent administrative cap; the 8 percent are the indirect fees.
Senate Committee on Health and Human Services April 27, 2021 BARRY GOLD (AARP): We support A.B.
ASSEMBLYWOMAN TITUS MADE A MOTION TO AMEND AND DO PASS ASSEMBLY BILL 216.
```

Technology: nltk, textblob, wordcloud

nltk

1. Word_tokenize: split a given sentence into words



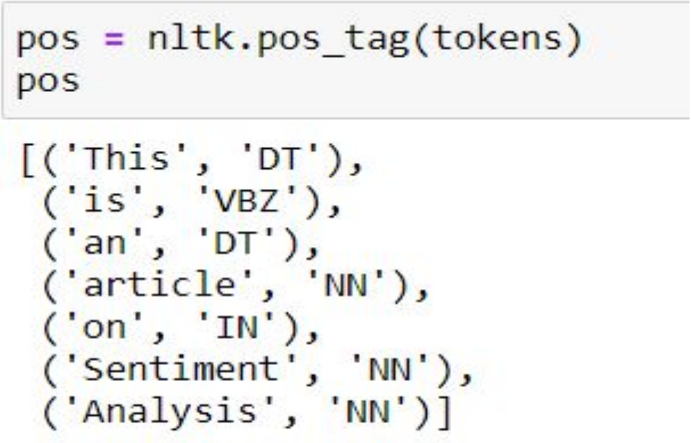
```python
from nltk.tokenize import word_tokenize

words = word_tokenize(text)
print(words)
```

```
['A', 'Topic', 'in', 'Kafka', 'is', 'something', 'where', 'a', 'message',
'is', 'sent', '.', 'The', 'consumer', 'applications', 'which', 'are', 'in
terested', 'in', 'that', 'topic', 'pulls', 'the', 'message', 'inside', 't
hat', 'topic', 'and', 'can', 'do', 'anything', 'with', 'that', 'data',
'.', 'Up', 'to', 'a', 'specific', 'time', ',', 'any', 'number', 'of', 'co
nsumer', 'applications', 'can', 'pull', 'this', 'message', 'any', 'numbe
r', 'of', 'times', '.']
```

Technology: nltk, textblob, wordcloud

nltk

2. Pos_tag: assign each word in a text corpus to a grammatical category



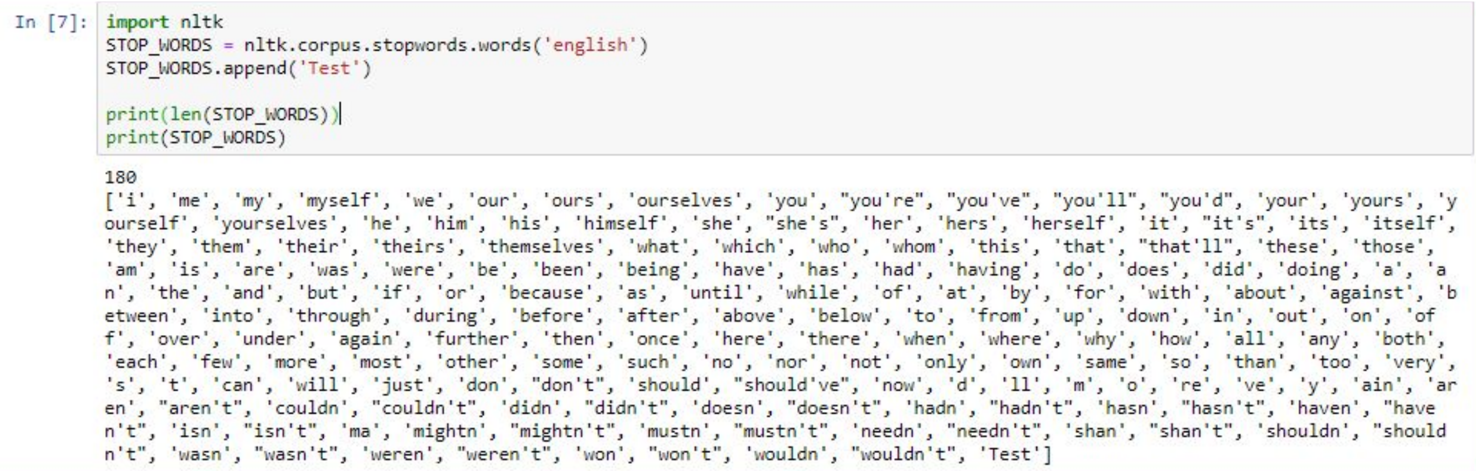
```
pos = nltk.pos_tag(tokens)
pos
```

```
[('This', 'DT'),
 ('is', 'VBZ'),
 ('an', 'DT'),
 ('article', 'NN'),
 ('on', 'IN'),
 ('Sentiment', 'NN'),
 ('Analysis', 'NN')]
```

Technology: nltk, textblob, wordcloud

nltk

3. Stopwords: a corpus, a list of words that are very common but don't provide useful information for most text analysis procedures



In [7]:

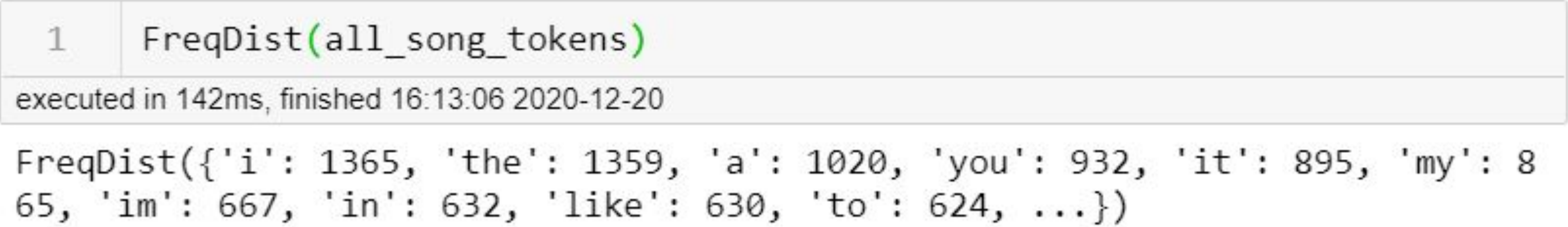
```
import nltk
STOP_WORDS = nltk.corpus.stopwords.words('english')
STOP_WORDS.append('Test')

print(len(STOP_WORDS))
print(STOP_WORDS)
```

```
180
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'y
ourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", 'Test']
```

Technology: nltk, textblob, wordcloud

nltk

4. Freqdist: a function which gives you the frequency of words within a text



```
FreqDist(all_song_tokens)
```

executed in 142ms, finished 16:13:06 2020-12-20

```
FreqDist({'i': 1365, 'the': 1359, 'a': 1020, 'you': 932, 'it': 895, 'my': 8
65, 'im': 667, 'in': 632, 'like': 630, 'to': 624, ...})
```

Technology: nltk, textblob, wordcloud

Analysis was divided into two parts:

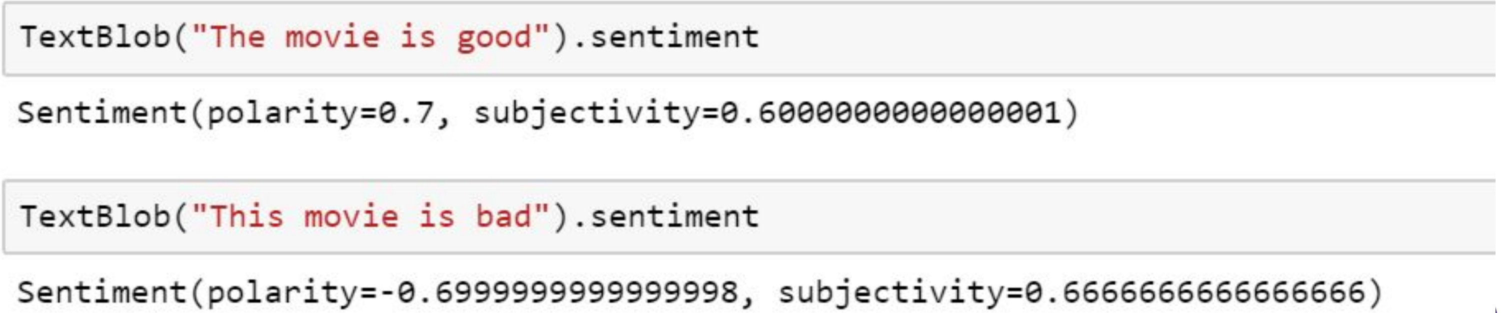
1. Cleaning: Use the Word_tokenize, pos_tag technology combined with stopwords corpus to clean stop words, punctuations, lemmatize words in the original JSON file.

2. Analysis: Use freqdist to do word frequency analysis, implement TF-IDF for keyword extraction

Technology: nltk, textblob, wordcloud

Textblob

- Use the textblob to complete sentiment analysis for the text, providing sentiment polarity of every sentence, positive or negative, ranging from [-1,1]



```
TextBlob("The movie is good").sentiment
```

```
Sentiment(polarity=0.7, subjectivity=0.6000000000000001)
```

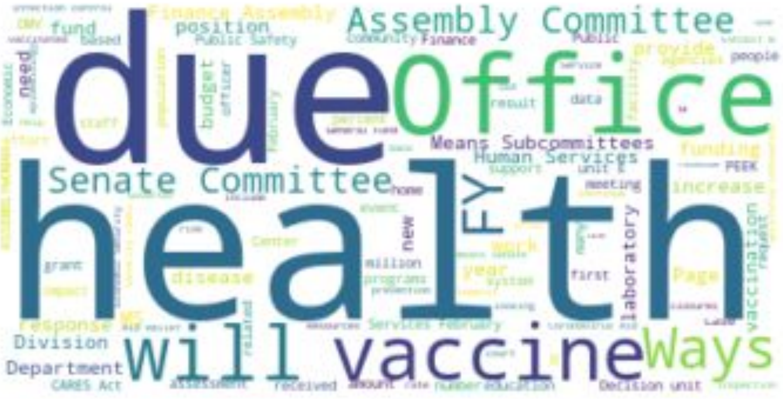
```
TextBlob("This movie is bad").sentiment
```

```
Sentiment(polarity=-0.6999999999999998, subjectivity=0.6666666666666666)
```

Technology: nltk, textblob, wordcloud

Wordcloud

> For data visualization, we use the wordcloud package , which represents data by the frequency.



Technology: Plotly Dash

1. Dash_bootstrap_components (Card): A library of Bootstrap components for use with Plotly Dash.
2. Html:html components to help us organize and display the output
3. Dcc(dropdown, input): (ConfirmDialog component) send a dialog to the browser asking the user to confirm or cancel with a custom message
4. Dash.dependencies.input, dash.dependencies.output

Dashboard Demonstration

Challenges and Lessons Learned

Challenges:

- Known bug w/ Tensorflow (DeepSpeech) and M1 Macs
- File management with large file sizes
- Subjective text processing decisions that require knowledge about state political context and policy content

Lessons learned:

- Use Github to collaborate efficiently with branches
- Cultivate good coding style and documentation
- Define the project scope early

Next Steps

- StateLegiscraper will continue active development
- First priority on adding a helpers module and building functionality for:
  - Google Cloud Speech-to-Text API
  - OpenStates API
- Secondary priority on expanding into other states

Thank you! Any questions?