TLAP

October 8, 2025

[1]:
```
# This is my Talk Like a President Project

# The goal of this project is to build a traslator that makes any text sound␣
 ↪like a selected President
```

[2]:
```
import pandas as pd
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import nltk
from nltk.stem import WordNetLemmatizer
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from collections import Counter
import random
import textstat
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
```

DATA CLEANING

Each presidential inaugural address is converted to lowercase, stripped of punctuation, and normalized to remove special characters. This ensures consistent text for tokenization and TF-IDF scoring.

[ ]:
```
# pre processing
# load my data set

df = pd.read_excel("prez_data.xlsx")

df.head() # name, inaugural address, date, and text
print(len(df))
```

[4]:
```
df['President'] = df['Name']
speeches = df.groupby('President')['Text'].apply(lambda x: ' '.join(x)).
 ↪reset_index()
```

[5]:
```
# start cleaning
def clean(t):
    t = t.lower() # lower case
    t = re.sub(r'[-]', ' ', t) # replace '-' with white spaces
    t = re.sub(r'[^\w\s.,!?;:]', '', t) # keep puncuation
 ↪#t = re.sub(r'[^a-z\s]', '', t) # no special chars
    t = re.sub(r'\s+', ' ', t) # only 1 space
    t = re.sub(r'[\u0000-\u001F\u007F-\u009F]', '', t) # weird characters i
 ↪noticed

    return t.strip()

speeches['clean'] = speeches['Text'].apply(clean) # apply it

# tokenzie
def tokenize(t):
    return nltk.word_tokenize(t)

# segment
def segment(t):
    return [ s.strip() for s in re.split(r'[.!?]', t) if s.strip()]
```

[6]:
```
# use tokenization
speeches['tokens'] = speeches['clean'].apply(tokenize)
```

TF-IDF Analysis

TF-IDF (Term Frequency–Inverse Document Frequency) highlights which words are most unique to each president's speech. Higher scores indicate words that define a president's rhetorical style rather than general political language.

[7]:
```
# building the president's vocabs TF IDF
top_w = {}

for _, row in speeches.iterrows(): # iterating over speeches
    j = ' '.join(row['tokens']) # taking the tokens and putting back  into
 ↪string
    vectz = TfidfVectorizer(stop_words='english')
    v = vectz.fit_transform([j])
    score = {}
    for w in j.split():
        if w in vectz.vocabulary_:
            score[w] = v[0, vectz.vocabulary_[w]]
    sorted_score = sorted(score.items(), key=lambda x: x[1], reverse=True)
    top_w[row['President']] = pd.DataFrame(sorted_score, columns = ['Word(s)',
 ↪'TF-IDF Score'])

    #print(row['President'])
```

```
    #print("\ttop TF-IDF terms:", sorted_score[:10])
```

Lemmatization

Words are reduced to their base forms (e.g., running → run) to group similar words and reduce noise before analysis.

[8]:
```
# lemmatize
lemmatizer = WordNetLemmatizer()
def lemmatize(t):
    return[lemmatizer.lemmatize(w) for w in t]

# apply it
speeches['tokens'] = speeches['tokens'].apply(lambda t: [lemmatizer.
 ↪lemmatize(w) for w in t])
```

[9]:
```
# my rewriter using TF IDF
def rewrite(Input, president):
    if president not in top_w:
        return "Pick a different President!" # quick check! it's case sensitive␣
 ↪so it will display this message if not written the exact way

    tokens = tokenize(clean(Input)) # the user input must also be cleaned and␣
 ↪tokenized
    vocab = top_w[president]
    top_words = vocab['Word(s)'].tolist() # based on the president, pick their␣
 ↪top words

    # add back punc
    rew = []
    for w in tokens:
        if w.isalpha() and len(w) > 3 and random.random() < 0.65: # 65% of the␣
 ↪time words will be replaced
            rew.append(random.choice(top_words))
        else:
            rew.append(w)  # keep punctuation
    return " ".join(rew).replace(" ,", ",").replace(" .", ".").replace(" !", "!
 ↪").replace(" ?", "?")
```

[10]:
```
def rewrite_pos(Input, president):
    if president not in top_w:
        return "Pick a different President!" # quick check again

    tokens = nltk.word_tokenize(clean(Input)) # clean & tokenize
    tagged = nltk.pos_tag(tokens, tagset='universal') # tag each word based on␣
 ↪POS

    vocab = top_w[president]
```

```
    top_words = vocab['Word(s)'].tolist()
    prez_w = [w for w in top_words if w.isalpha()]
    prez_tagged = nltk.pos_tag(prez_w, tagset='universal')

    pos_dict = {}
    for w, tag in prez_tagged:
        pos_dict.setdefault(tag, []).append(w) # group them

    rew = [] # rew is for my rewritten sentances
    content_tags = {"NOUN", "VERB", "ADJ", "ADV"}
    for w, tag in tagged:
        if (
            w.isalpha() and
            tag in content_tags and tag in pos_dict and len(w) > 3 and random.
 ↪random() < 0.5):
            rew.append(random.choice(pos_dict[tag]))
        else:
            rew.append(w)

    return " ".join(rew).replace(" ,", ",").replace(" .", ".").replace(" !", "!
 ↪").replace(" ?", "?")
```

[11]:
```
# analysis
def Lmetrics(tokens):
    total = len(tokens) # total words
    unique = len(set(tokens)) # unique words

    return pd.Series({
        'totalWords': total,
        'uniqueWords': unique,
        'ttr': unique / total if total else 0,
    })
```

[12]:
```
mets = speeches['tokens'].apply(Lmetrics) # apply metrics
mets['President'] = speeches['President']
```

[13]:
```
def avgSentLen(t): # this is for my future plotting
    sentences = re.split(r'[.!?]', t) # putting common puncutation
    sentences = [s for s in sentences if s.strip()] # no empty spaces
    wc = [len(tokenize(s)) for s in sentences] # using tokenize to sount the␣
 ↪words in each sentance
    return sum(wc) / len(wc) if sentences else 0 # calcuating the average

speeches['avg_sentence_length'] = speeches['clean'].apply(avgSentLen) # new␣
 ↪column!
```

[14]:
```
# recalc the metrics
L = speeches['tokens'].apply(Lmetrics)
L['President'] = speeches['President']

# i want the avg sentance lengths
slen = speeches[['President', 'avg_sentence_length']]

# merge on president
ttrAndLen = pd.merge(L, slen, on='President')
```

[15]:
```
ttrAndLen = ttrAndLen.sort_values(by='ttr', ascending=True)
ttrAndLen
```

[15]:

	totalWords	uniqueWords	ttr	President
20	8571.0	1574.0	0.183643	James Monroe
35	5956.0	1257.0	0.211048	William Henry Harrison
28	4322.0	948.0	0.219343	Richard Milhous Nixon
36	5827.0	1291.0	0.221555	William Howard Taft
37	6767.0	1523.0	0.225063	William McKinley
8	6101.0	1415.0	0.231929	Franklin D. Roosevelt
29	5609.0	1302.0	0.232127	Ronald Reagan
18	5153.0	1200.0	0.232874	James Knox Polk
0	4748.0	1118.0	0.235468	Abraham Lincoln
6	4959.0	1178.0	0.237548	Donald J. Trump
15	4071.0	970.0	0.238271	Herbert Hoover
4	4200.0	1026.0	0.244286	Bill Clinton
11	4110.0	1008.0	0.245255	George W. Bush
5	4439.0	1100.0	0.247804	Calvin Coolidge
22	2970.0	736.0	0.247811	Joe Biden
7	4549.0	1138.0	0.250165	Dwight D. Eisenhower
38	3534.0	903.0	0.255518	Woodrow Wilson
2	4972.0	1293.0	0.260056	Barack Obama
3	4723.0	1237.0	0.261910	Benjamin Harrison
32	4300.0	1146.0	0.266512	Thomas Jefferson
10	2651.0	715.0	0.269710	George Bush
17	3065.0	861.0	0.280914	James Buchanan
13	3936.0	1110.0	0.282012	Grover Cleveland
14	2492.0	711.0	0.285313	Harry S. Truman
30	2694.0	769.0	0.285449	Rutherford B. Hayes
34	3729.0	1068.0	0.286404	Warren G. Harding
25	3144.0	910.0	0.289440	John Quincy Adams
16	3207.0	929.0	0.289679	James A. Garfield
33	2686.0	781.0	0.290767	Ulysses S. Grant
23	2577.0	759.0	0.294529	John Adams
9	3625.0	1068.0	0.294621	Franklin Pierce
27	4209.0	1262.0	0.299834	Martin Van Buren
26	1670.0	509.0	0.304790	Lyndon Baines Johnson

1	2473.0	781.0	0.315811	Andrew Jackson
19	2566.0	833.0	0.324630	James Madison
31	1085.0	378.0	0.348387	Theodore Roosevelt
24	1485.0	529.0	0.356229	John F. Kennedy
21	1347.0	482.0	0.357832	Jimmy Carter
12	1675.0	610.0	0.364179	George Washington
39	1175.0	473.0	0.402553	Zachary Taylor

	avg_sentence_length
20	32.488281
35	40.685315
28	23.451977
36	35.425000
37	28.421739
8	20.800000
29	20.755814
18	32.679739
0	28.490683
6	17.120438
15	23.823171
4	19.507317
11	20.005102
5	21.647959
22	14.893048
7	20.176744
38	26.826772
2	24.792746
3	29.082803
32	49.011628
10	17.074830
17	33.438202
13	37.588235
14	20.135593
30	44.661017
34	24.195946
25	40.368421
16	27.891892
33	31.361446
23	68.648649
9	33.855769
27	41.969388
26	16.956989
1	43.963636
19	46.518519
31	31.878788
24	26.036364
21	24.903846

```
12              61.111111
39              52.409091
```

[ ]:

Lexical Metrics

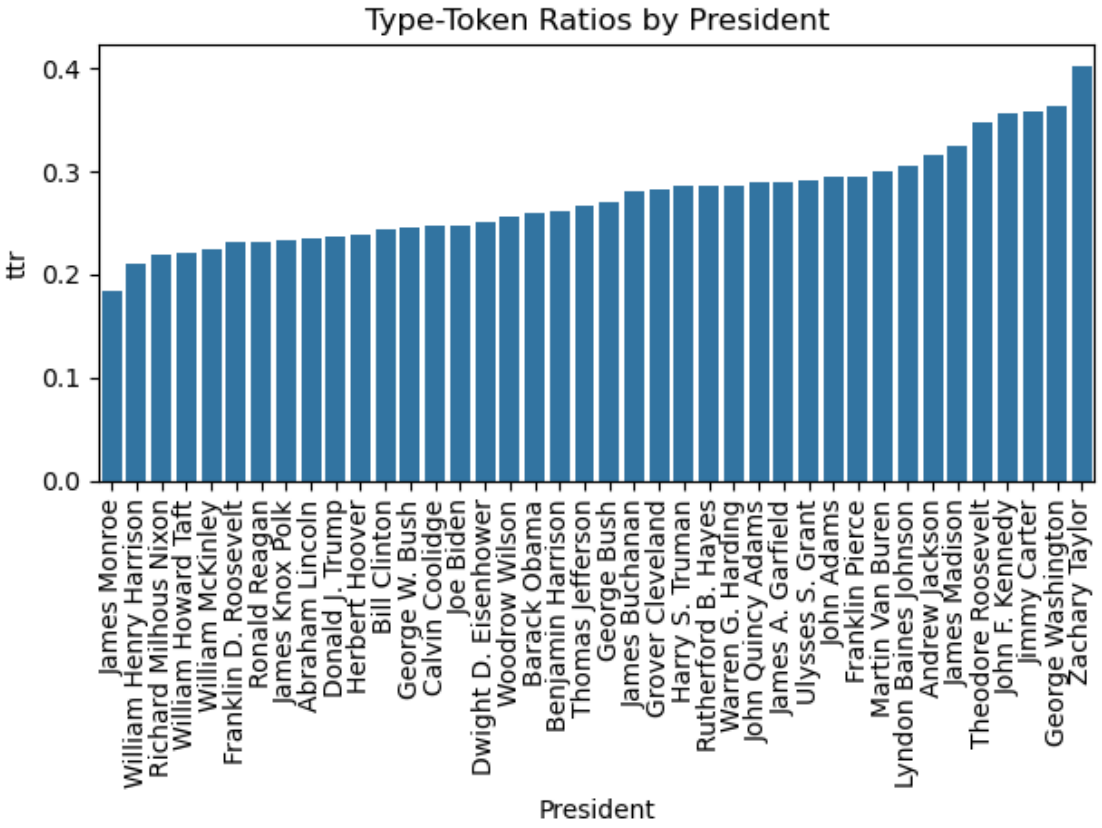
The Type-Token Ratio (TTR) measures vocabulary richness:

Higher TTR → greater word variety

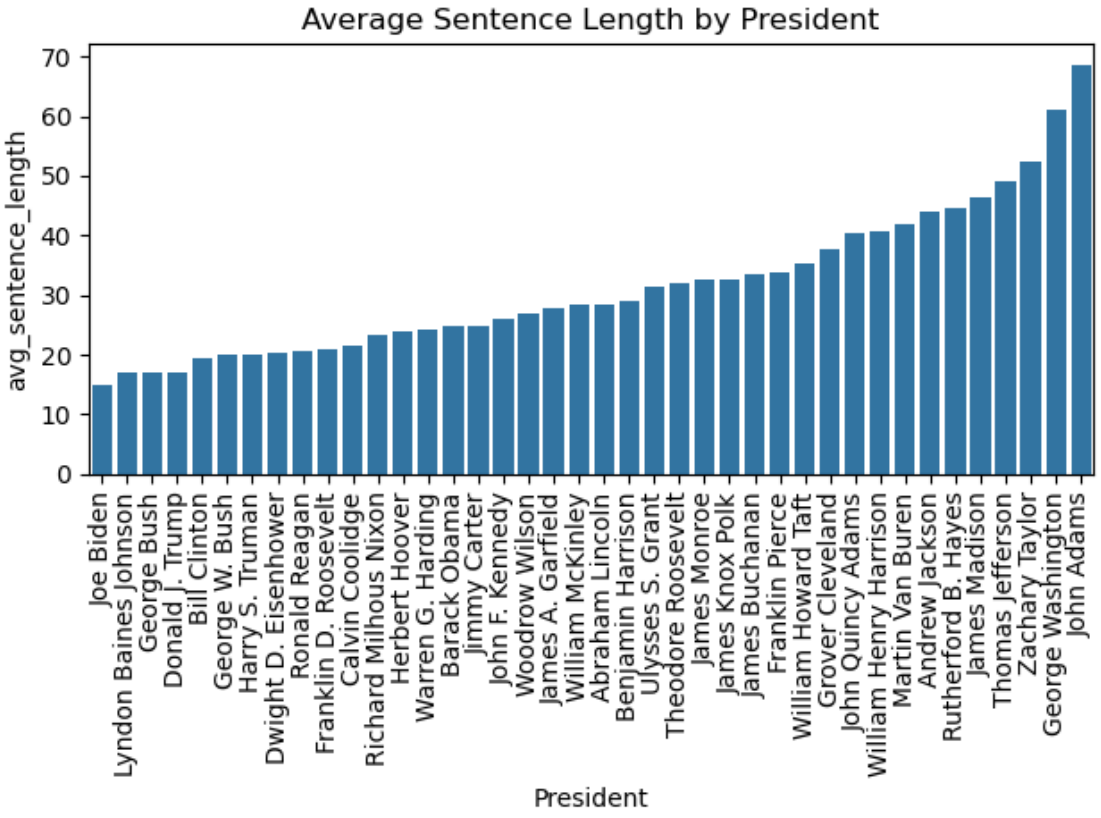
Lower TTR → more repetition

The Average Sentence Length shows complexity of syntax and rhetorical density.

[16]:
```
sns.barplot(data=ttrAndLen.sort_values('ttr'), x='President', y='ttr')
plt.title('Type-Token Ratios by President')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



[17]:
```
sns.barplot(data=ttrAndLen.sort_values('avg_sentence_length'), x='President',␣
↪y='avg_sentence_length')
plt.title('Average Sentence Length by President')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



[18]:
```
print(ttrAndLen.sort_values('ttr'))
```

```
    totalWords  uniqueWords       ttr              President  \
20      8571.0       1574.0  0.183643          James Monroe
35      5956.0       1257.0  0.211048  William Henry Harrison
28      4322.0        948.0  0.219343  Richard Milhous Nixon
36      5827.0       1291.0  0.221555    William Howard Taft
37      6767.0       1523.0  0.225063       William McKinley
8       6101.0       1415.0  0.231929  Franklin D. Roosevelt
29      5609.0       1302.0  0.232127         Ronald Reagan
18      5153.0       1200.0  0.232874       James Knox Polk
0       4748.0       1118.0  0.235468       Abraham Lincoln
6       4959.0       1178.0  0.237548       Donald J. Trump
15      4071.0        970.0  0.238271        Herbert Hoover
```

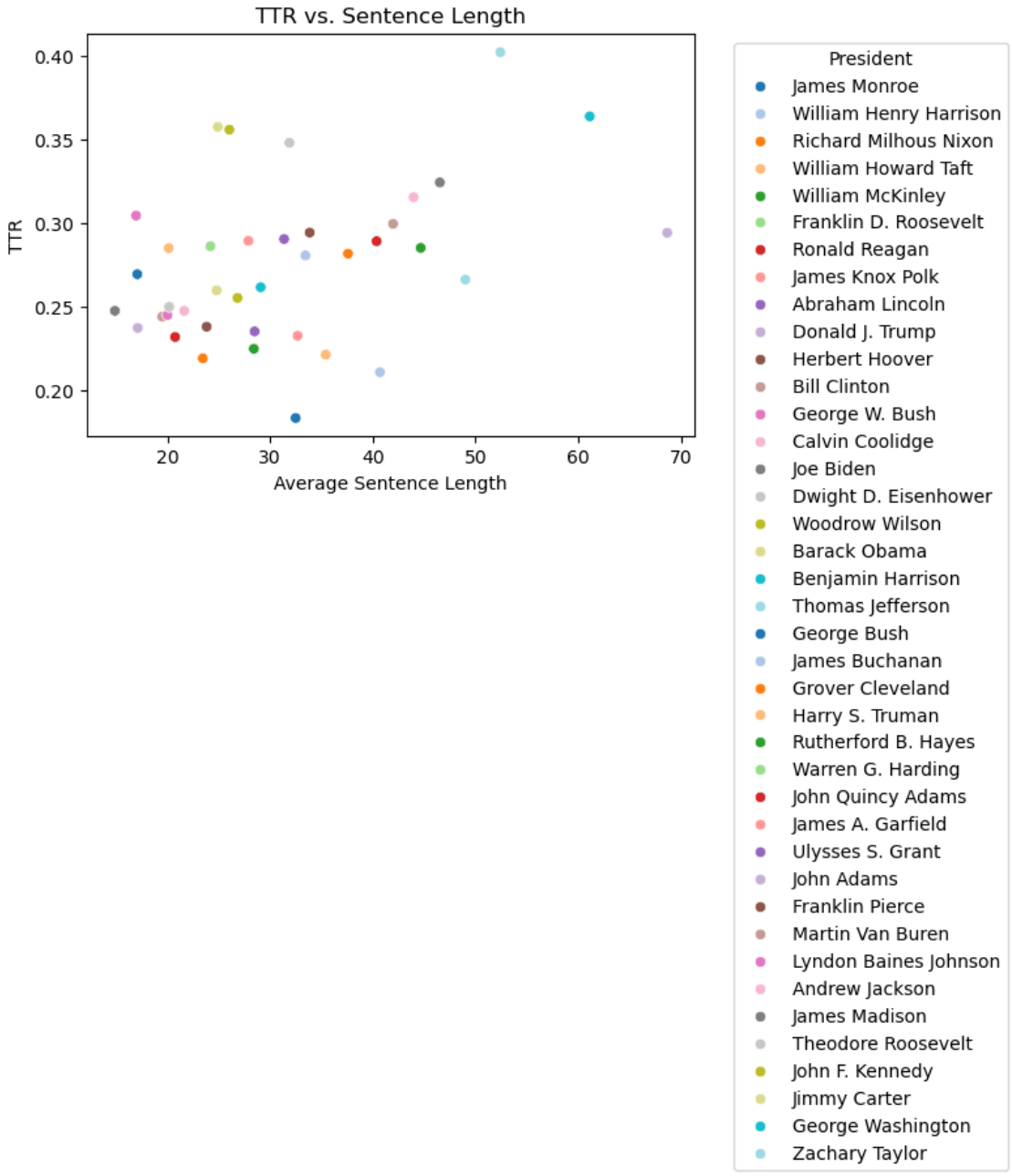
4	4200.0	1026.0	0.244286	Bill Clinton
11	4110.0	1008.0	0.245255	George W. Bush
5	4439.0	1100.0	0.247804	Calvin Coolidge
22	2970.0	736.0	0.247811	Joe Biden
7	4549.0	1138.0	0.250165	Dwight D. Eisenhower
38	3534.0	903.0	0.255518	Woodrow Wilson
2	4972.0	1293.0	0.260056	Barack Obama
3	4723.0	1237.0	0.261910	Benjamin Harrison
32	4300.0	1146.0	0.266512	Thomas Jefferson
10	2651.0	715.0	0.269710	George Bush
17	3065.0	861.0	0.280914	James Buchanan
13	3936.0	1110.0	0.282012	Grover Cleveland
14	2492.0	711.0	0.285313	Harry S. Truman
30	2694.0	769.0	0.285449	Rutherford B. Hayes
34	3729.0	1068.0	0.286404	Warren G. Harding
25	3144.0	910.0	0.289440	John Quincy Adams
16	3207.0	929.0	0.289679	James A. Garfield
33	2686.0	781.0	0.290767	Ulysses S. Grant
23	2577.0	759.0	0.294529	John Adams
9	3625.0	1068.0	0.294621	Franklin Pierce
27	4209.0	1262.0	0.299834	Martin Van Buren
26	1670.0	509.0	0.304790	Lyndon Baines Johnson
1	2473.0	781.0	0.315811	Andrew Jackson
19	2566.0	833.0	0.324630	James Madison
31	1085.0	378.0	0.348387	Theodore Roosevelt
24	1485.0	529.0	0.356229	John F. Kennedy
21	1347.0	482.0	0.357832	Jimmy Carter
12	1675.0	610.0	0.364179	George Washington
39	1175.0	473.0	0.402553	Zachary Taylor

	avg_sentence_length
20	32.488281
35	40.685315
28	23.451977
36	35.425000
37	28.421739
8	20.800000
29	20.755814
18	32.679739
0	28.490683
6	17.120438
15	23.823171
4	19.507317
11	20.005102
5	21.647959
22	14.893048
7	20.176744
38	26.826772

```
2              24.792746
3              29.082803
32             49.011628
10             17.074830
17             33.438202
13             37.588235
14             20.135593
30             44.661017
34             24.195946
25             40.368421
16             27.891892
33             31.361446
23             68.648649
9              33.855769
27             41.969388
26             16.956989
1              43.963636
19             46.518519
31             31.878788
24             26.036364
21             24.903846
12             61.111111
39             52.409091
```

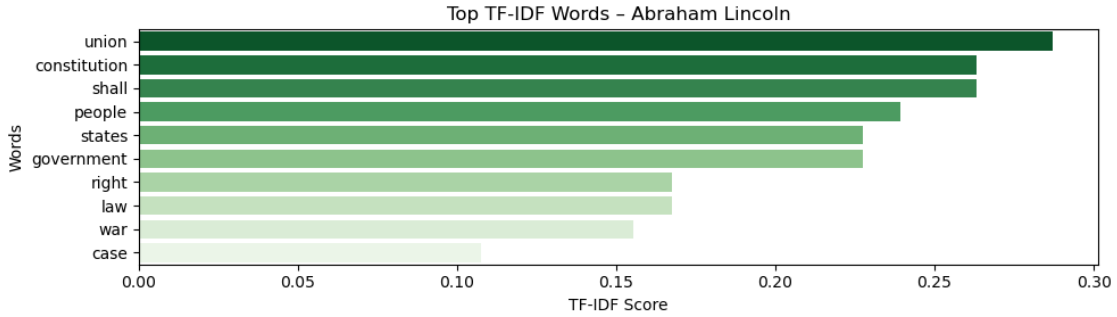
The following barplots and scatterplots compare presidents based on lexical diversity and sentence structure. The TF-IDF barplots display their most characteristic words.

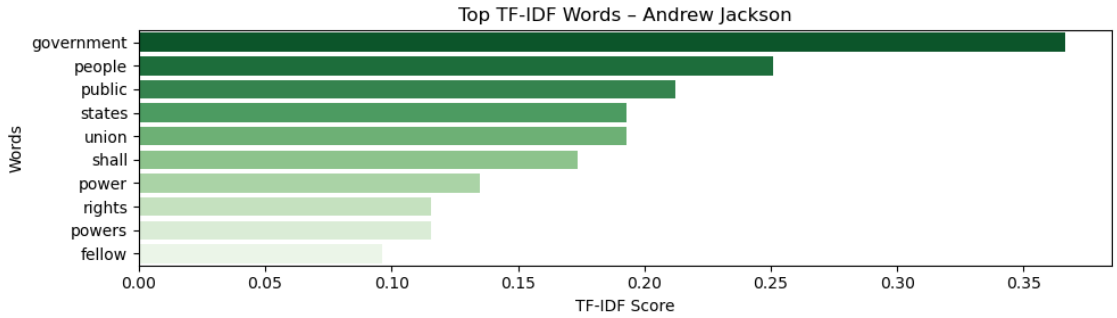
[19]:
```
plt.figure(figsize=(6, 4))
sns.set_palette("tab20", n_colors=len(ttrAndLen['President'].unique())) # more␣
 ↪different colors
sns.scatterplot(data=ttrAndLen, x='avg_sentence_length', y='ttr',␣
 ↪hue='President', legend=True)
plt.title("TTR vs. Sentence Length")
plt.xlabel("Average Sentence Length")
plt.ylabel("TTR")
plt.legend(title='President', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

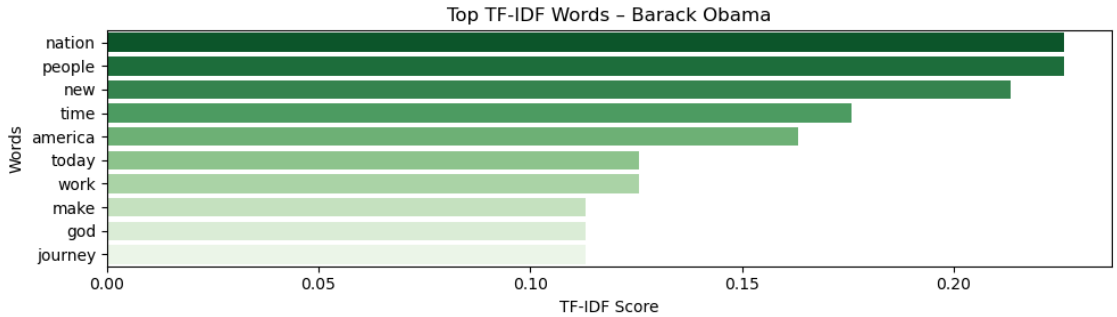


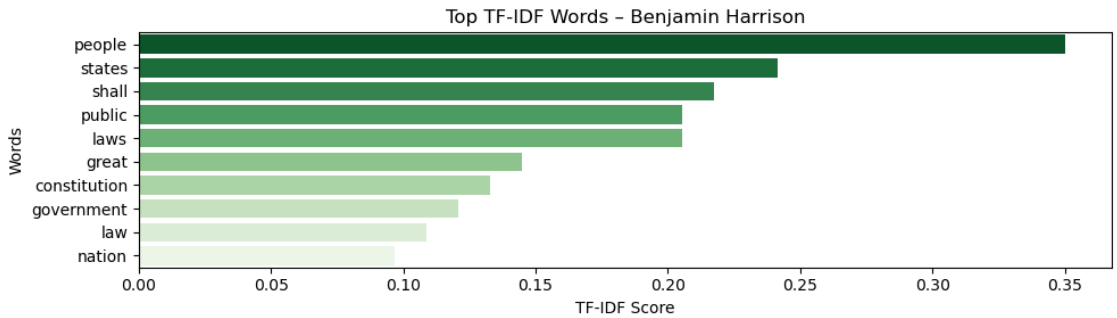
[20]:
```
for president, df in top_w.items():
    plt.figure(figsize=(10, 3))
    sns.barplot(data=df.head(10), x='TF-IDF Score', y='Word(s)',␣
↪hue='Word(s)', palette='Greens_r', legend=False)
    plt.title(f"Top TF-IDF Words - {president}")
    plt.xlabel("TF-IDF Score")
    plt.ylabel("Words")
    plt.tight_layout()
```

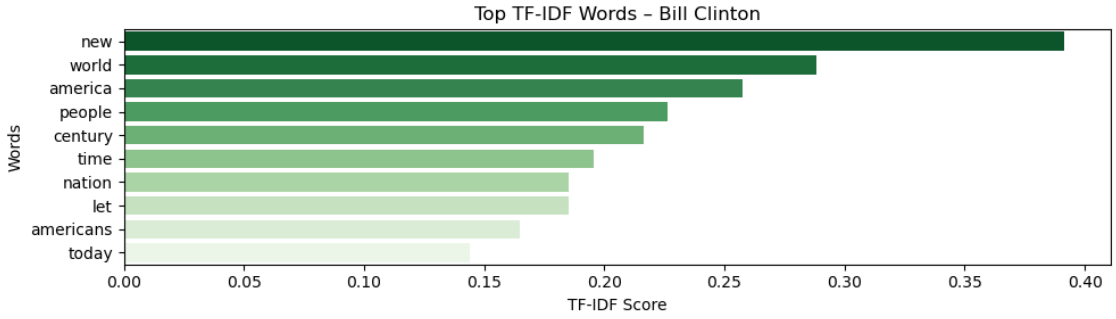
```
plt.show()
```

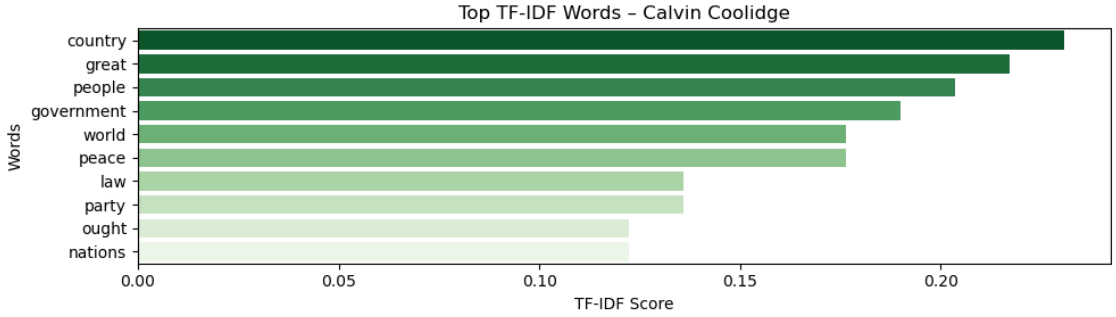

Top TF-IDF Words – Abraham Lincoln
union
constitution
shall
people
states
government
right
law
war
case
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

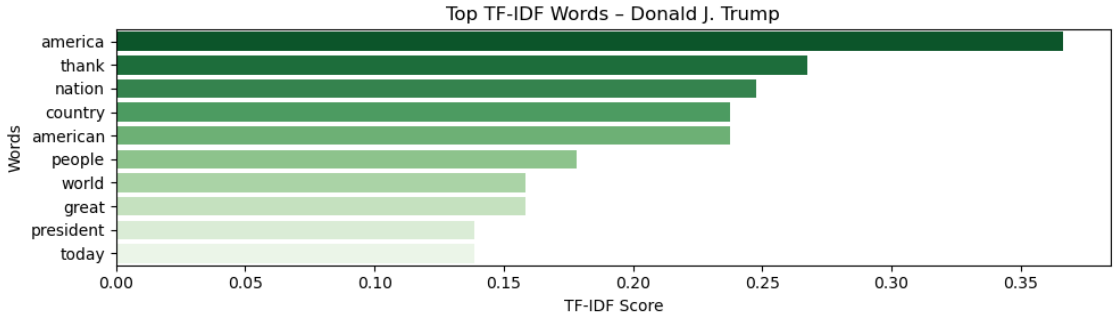

Top TF-IDF Words – Andrew Jackson
government
people
public
states
union
shall
power
rights
powers
fellow
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

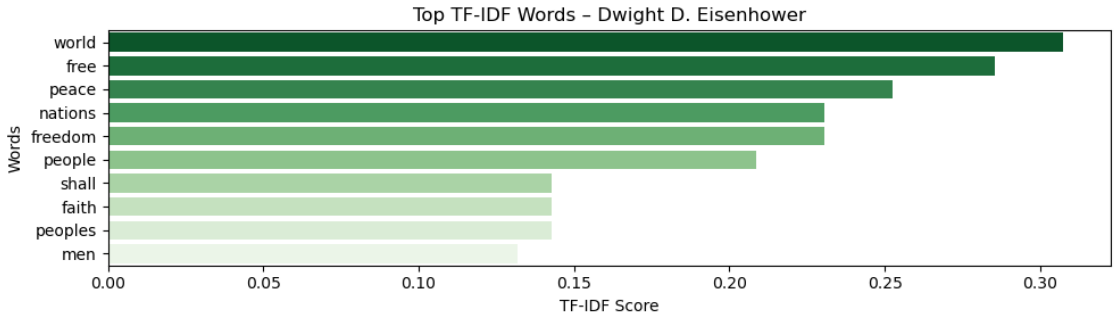

Top TF-IDF Words – Barack Obama
nation
people
new
time
america
today
work
make
god
journey
Words
0.00
0.05
0.10
0.15
0.20
TF-IDF Score

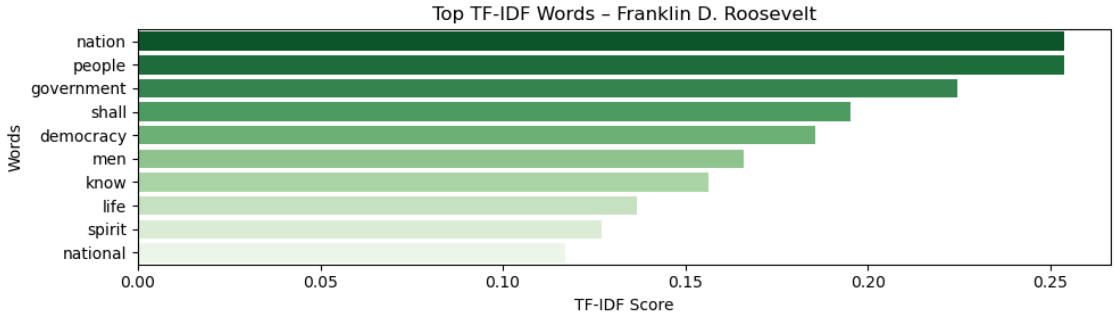

Top TF-IDF Words – Benjamin Harrison
people
states
shall
public
laws
great
constitution
government
law
nation
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

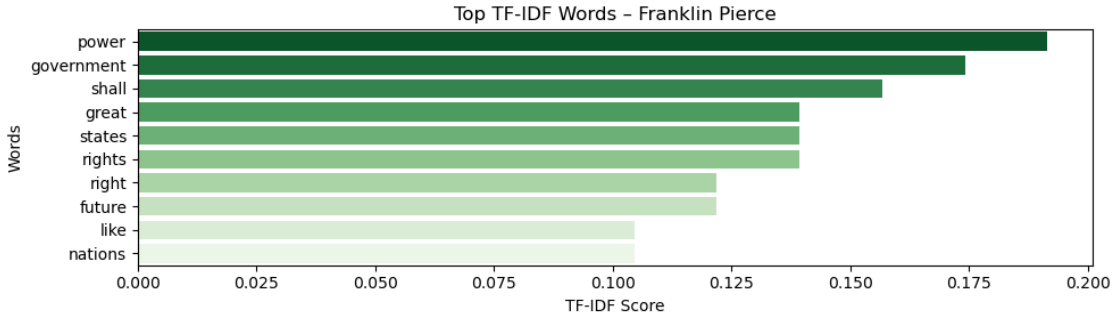

Top TF-IDF Words – Bill Clinton
new
world
america
people
century
time
nation
let
americans
today
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TF-IDF Score

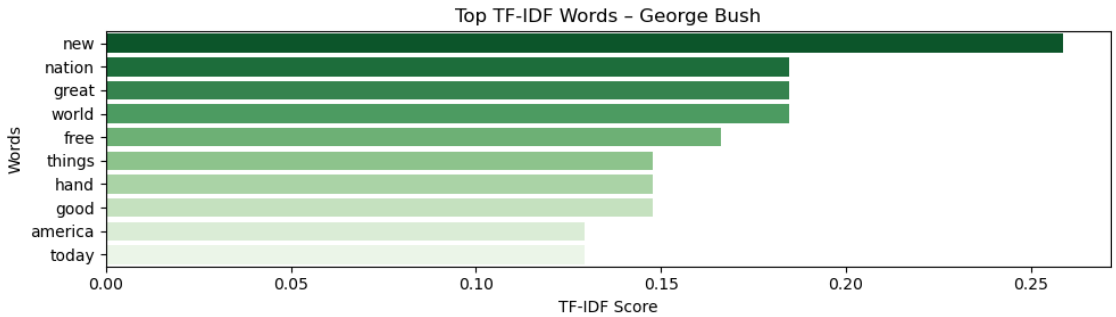

Top TF-IDF Words – Calvin Coolidge
country
great
people
government
world
peace
law
party
ought
nations
Words
0.00
0.05
0.10
0.15
0.20
TF-IDF Score

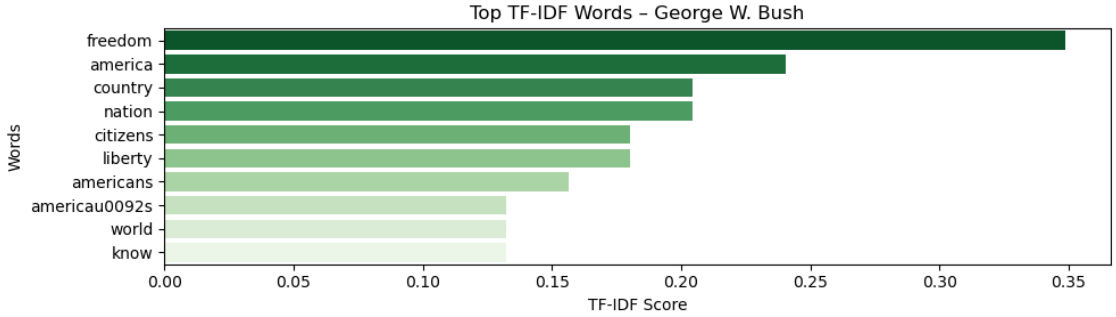

Top TF-IDF Words – Donald J. Trump
america
thank
nation
country
american
people
world
great
president
today
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

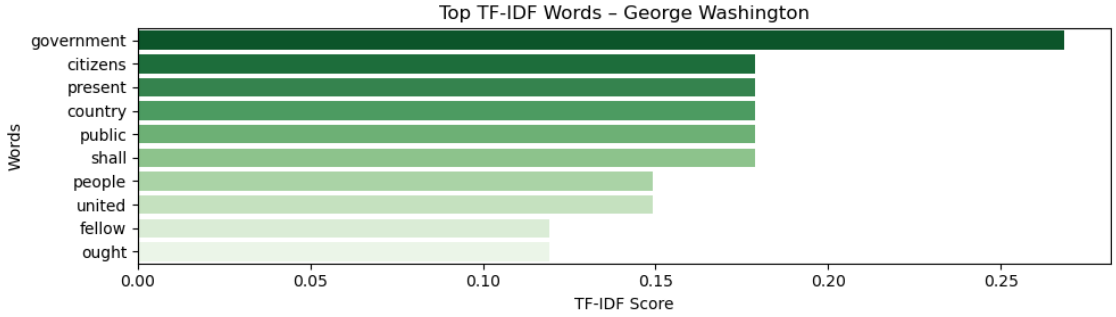

Top TF-IDF Words – Dwight D. Eisenhower
world
free
peace
nations
freedom
people
shall
faith
peoples
men
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

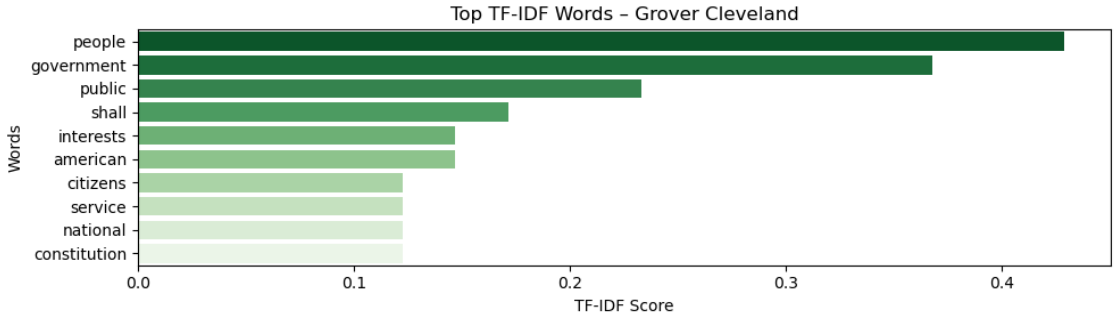

Top TF-IDF Words – Franklin D. Roosevelt
nation
people
government
shall
democracy
men
know
life
spirit
national
Words
0.00
0.05
0.10
0.15
0.20
0.25
TF-IDF Score

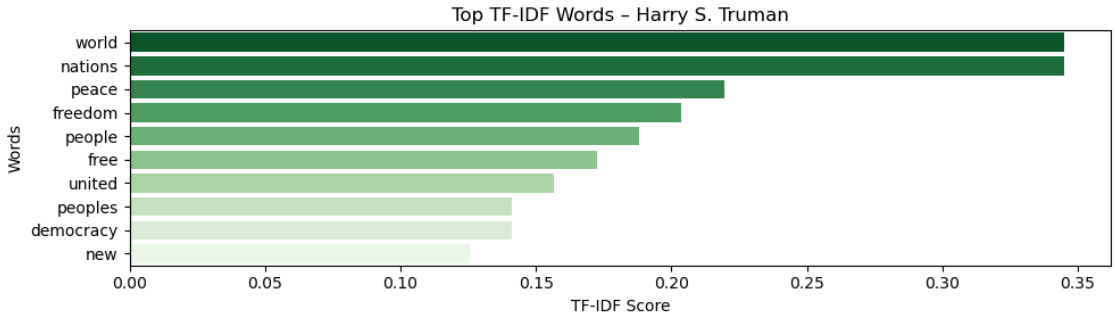

Top TF-IDF Words – Franklin Pierce
power
government
shall
great
states
rights
right
future
like
nations
Words
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
TF-IDF Score

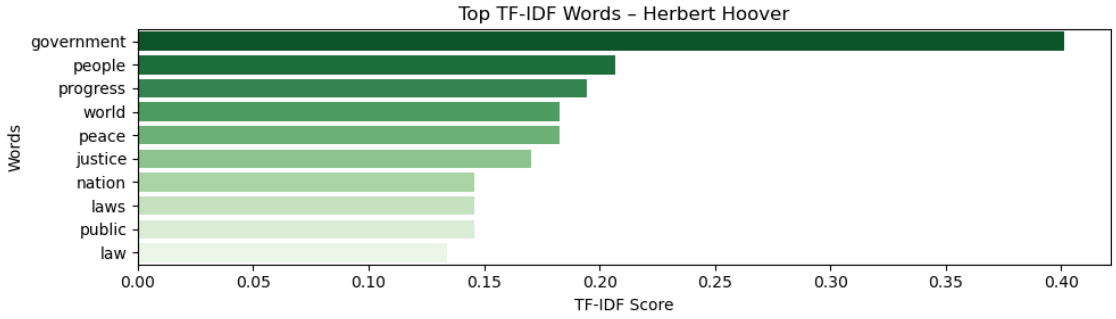

Top TF-IDF Words – George Bush
new
nation
great
world
free
things
hand
good
america
today
Words
0.00
0.05
0.10
0.15
0.20
0.25
TF-IDF Score

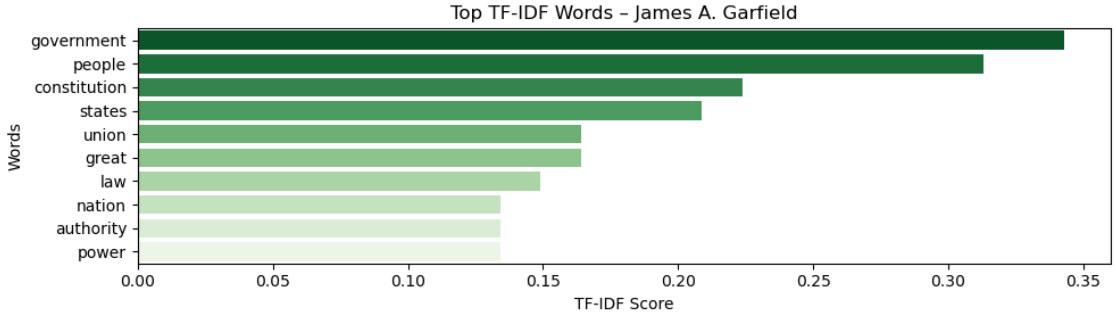

Top TF-IDF Words – George W. Bush
freedom
america
country
nation
citizens
liberty
americans
americau0092s
world
know
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

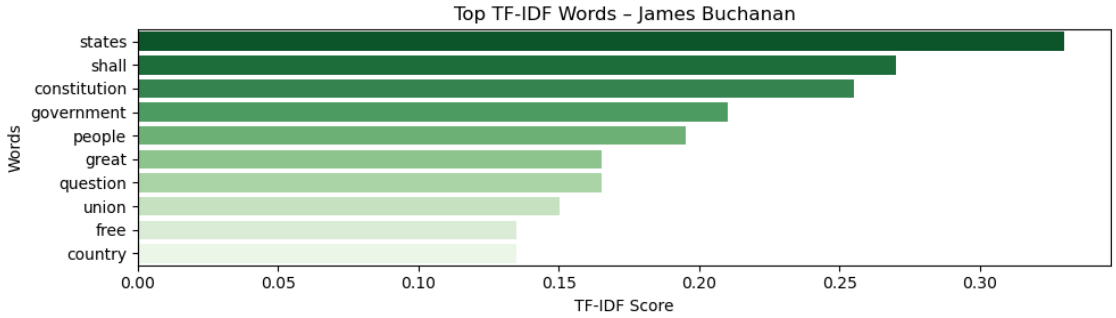

Top TF-IDF Words – George Washington
government
citizens
present
country
public
shall
people
united
fellow
ought
Words
0.00
0.05
0.10
0.15
0.20
0.25
TF-IDF Score

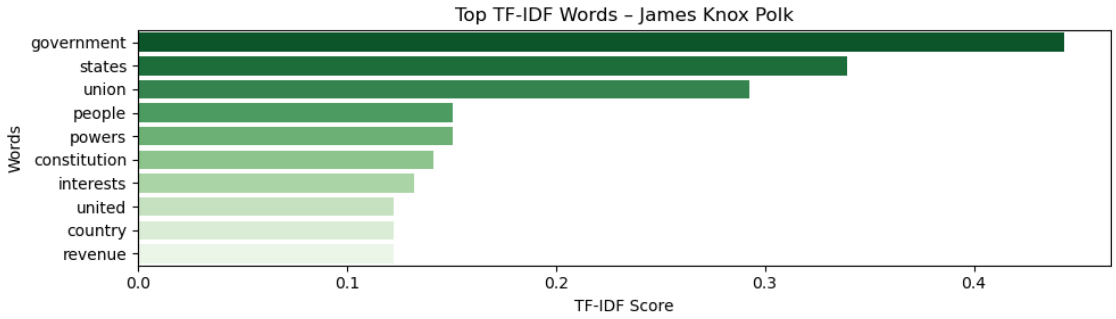

Top TF-IDF Words – Grover Cleveland
people
government
public
shall
interests
american
citizens
service
national
constitution
Words
0.0
0.1
0.2
0.3
0.4
TF-IDF Score

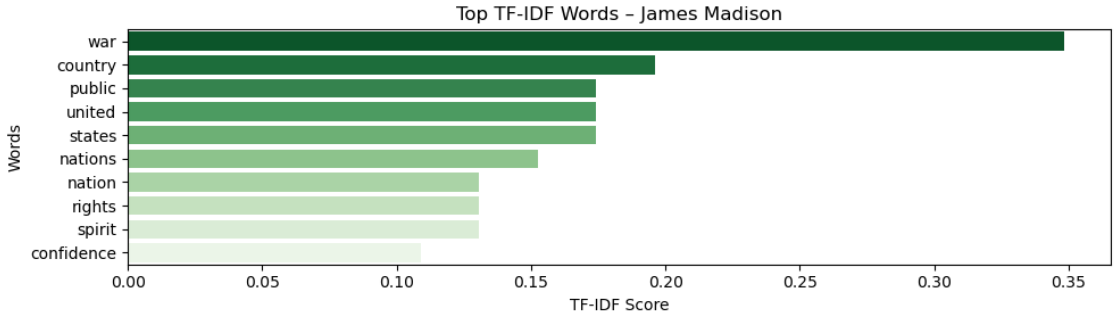

Top TF-IDF Words – Harry S. Truman
world
nations
peace
freedom
people
free
united
peoples
democracy
new
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

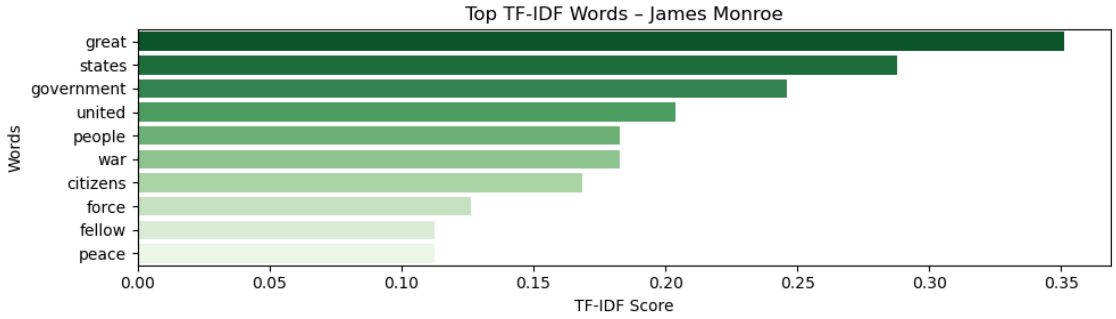

Top TF-IDF Words – Herbert Hoover
government
people
progress
world
peace
justice
nation
laws
public
law
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TF-IDF Score

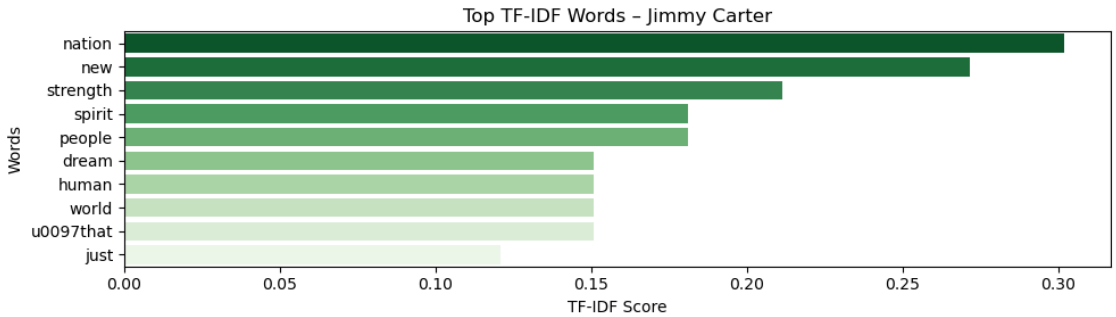

Top TF-IDF Words – James A. Garfield
government
people
constitution
states
union
great
law
nation
authority
power
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

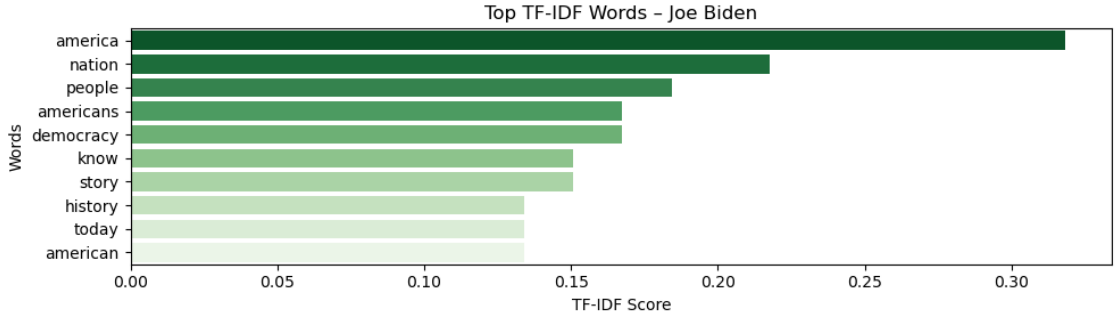

Top TF-IDF Words – James Buchanan
states
shall
constitution
government
people
great
question
union
free
country
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

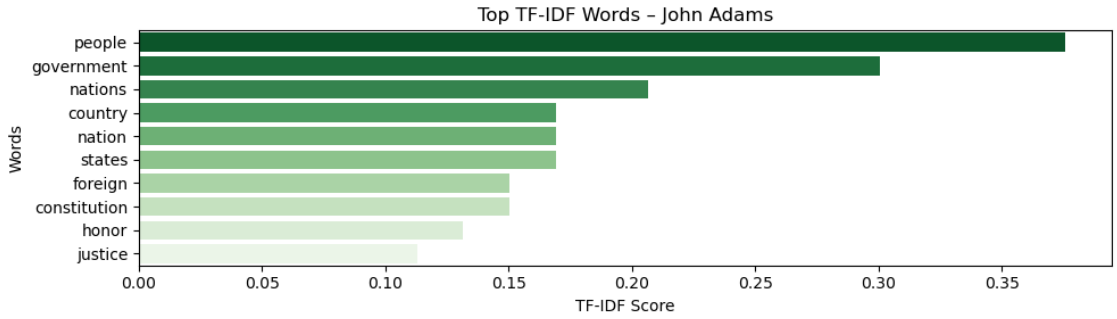

Top TF-IDF Words – James Knox Polk
government
states
union
people
powers
constitution
interests
united
country
revenue
Words
0.0
0.1
0.2
0.3
0.4
TF-IDF Score

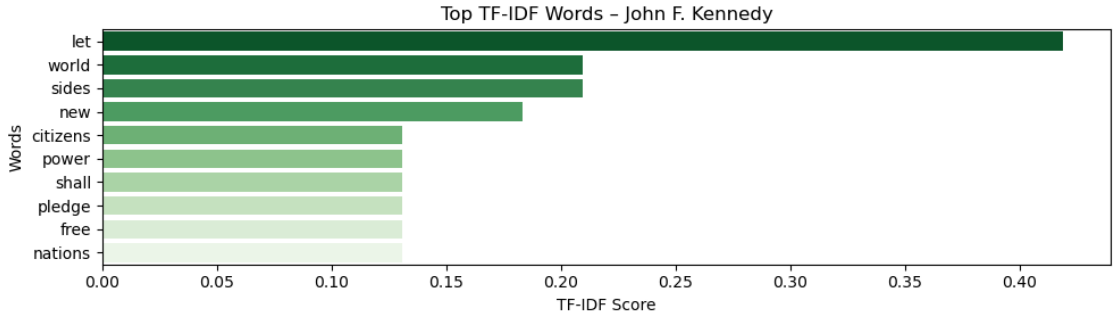

Top TF-IDF Words – James Madison
war
country
public
united
states
nations
nation
rights
spirit
confidence
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

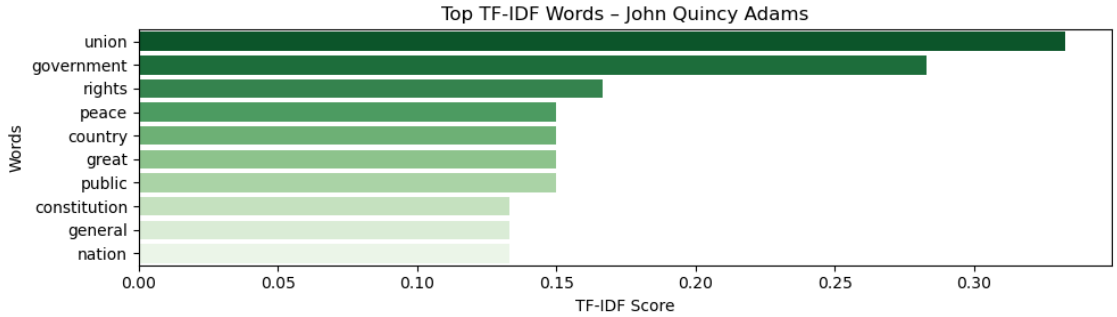

Top TF-IDF Words – James Monroe
great
states
government
united
people
war
citizens
force
fellow
peace
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

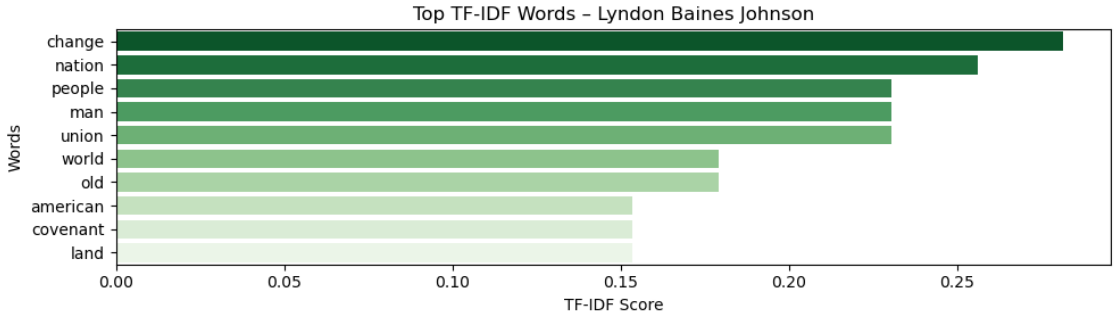

Top TF-IDF Words – Jimmy Carter
nation
new
strength
spirit
people
dream
human
world
u0097that
just
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

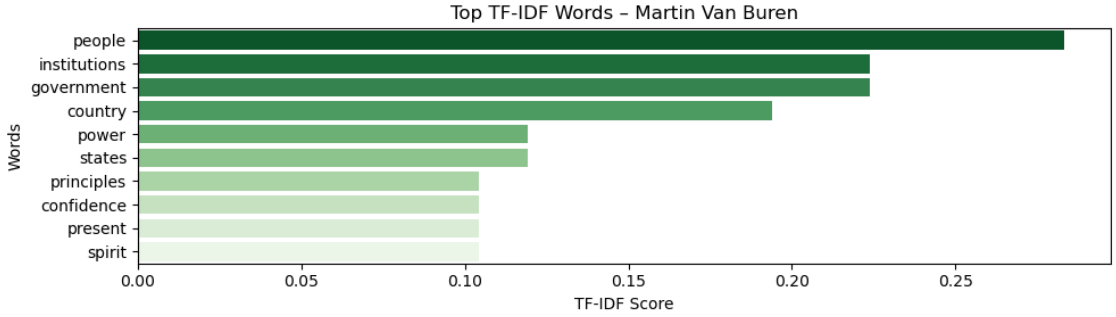

Top TF-IDF Words – Joe Biden
Words
america
nation
people
americans
democracy
know
story
history
today
american
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

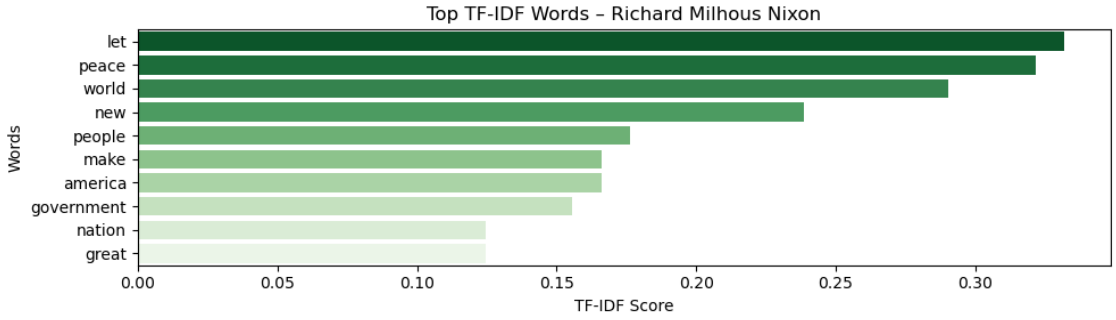

Top TF-IDF Words – John Adams
Words
people
government
nations
country
nation
states
foreign
constitution
honor
justice
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

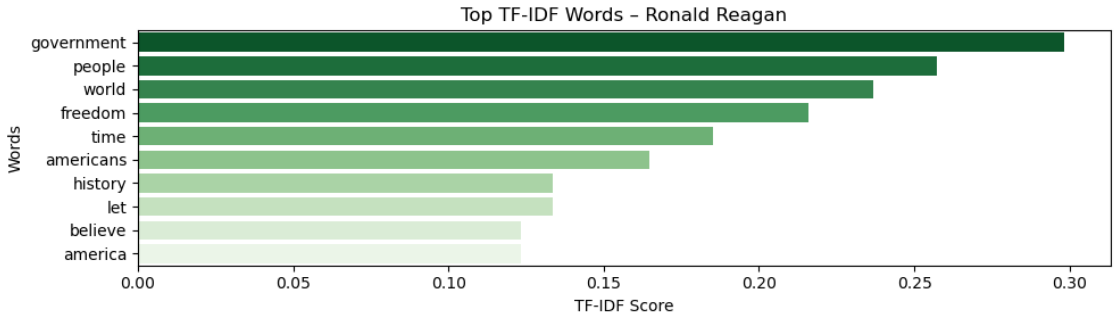

Top TF-IDF Words – John F. Kennedy
Words
let
world
sides
new
citizens
power
shall
pledge
free
nations
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TF-IDF Score

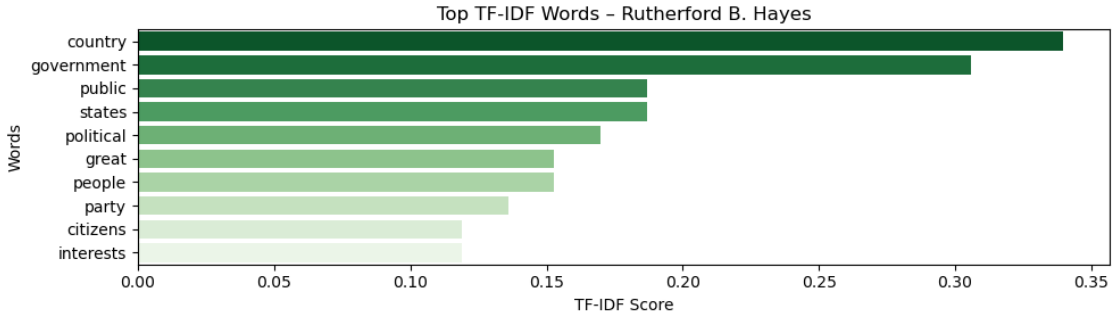

Top TF-IDF Words – John Quincy Adams
Words
union
government
rights
peace
country
great
public
constitution
general
nation
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

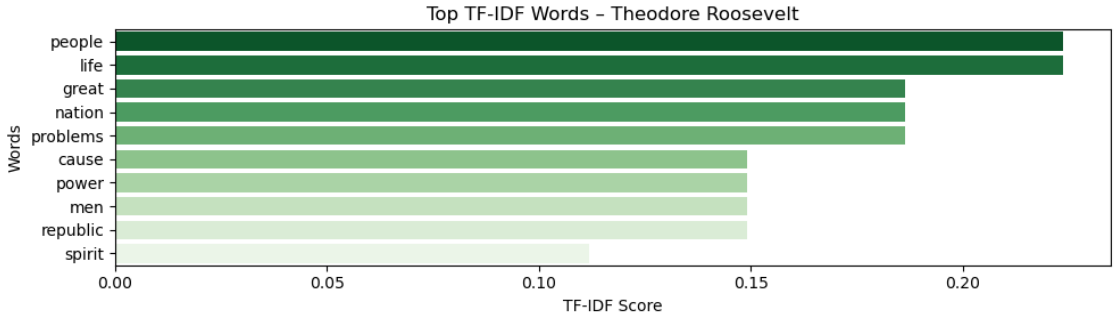

Top TF-IDF Words – Lyndon Baines Johnson
change
nation
people
man
union
world
old
american
covenant
land
Words
0.00
0.05
0.10
0.15
0.20
0.25
TF-IDF Score

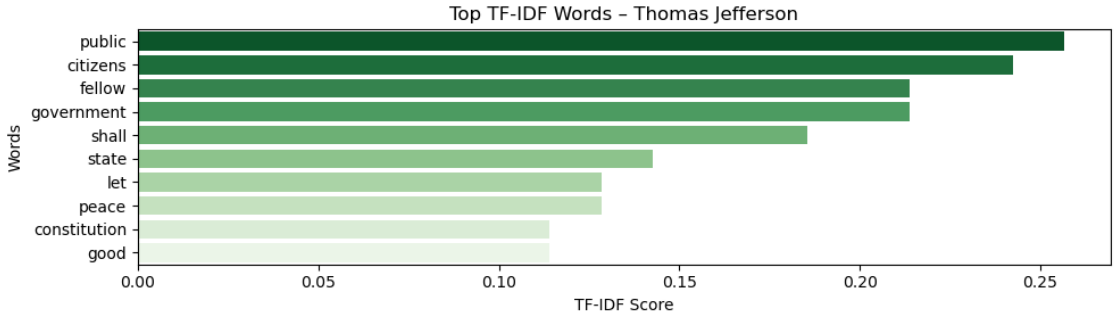

Top TF-IDF Words – Martin Van Buren
people
institutions
government
country
power
states
principles
confidence
present
spirit
Words
0.00
0.05
0.10
0.15
0.20
0.25
TF-IDF Score

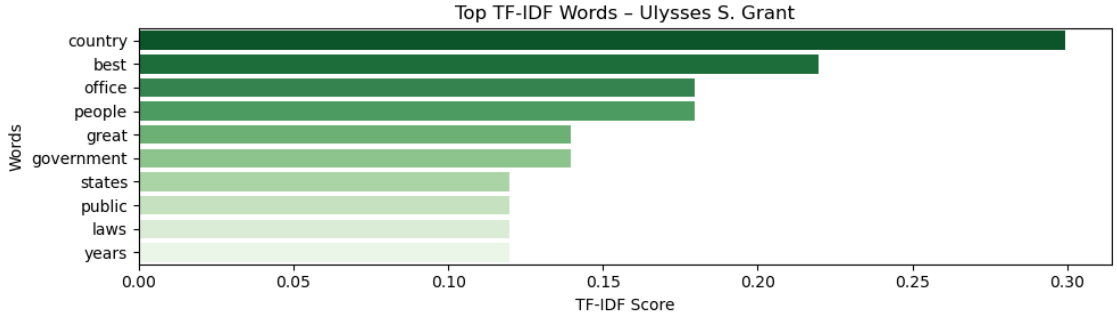

Top TF-IDF Words – Richard Milhous Nixon
let
peace
world
new
people
make
america
government
nation
great
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

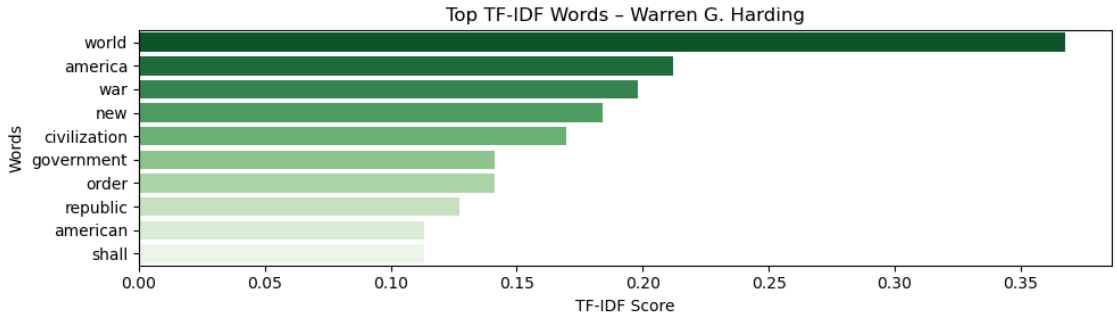

Top TF-IDF Words – Ronald Reagan
government
people
world
freedom
time
americans
history
let
believe
america
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

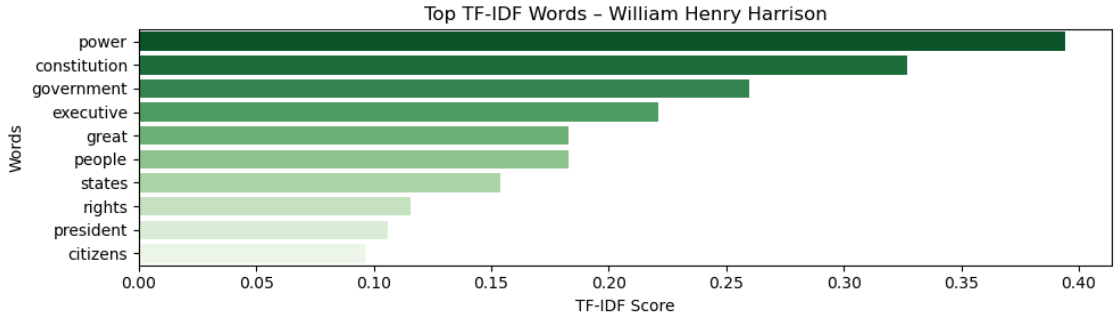

Top TF-IDF Words – Rutherford B. Hayes
country
government
public
states
political
great
people
party
citizens
interests
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score

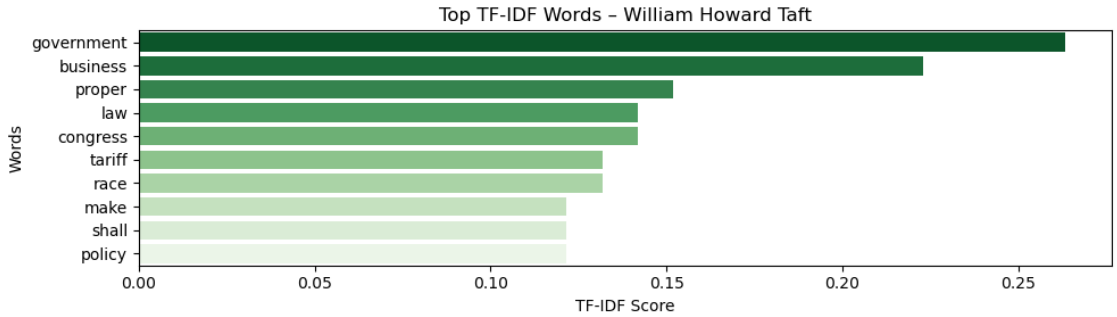

Top TF-IDF Words – Theodore Roosevelt
people
life
great
nation
problems
cause
power
men
republic
spirit
Words
0.00
0.05
0.10
0.15
0.20
TF-IDF Score

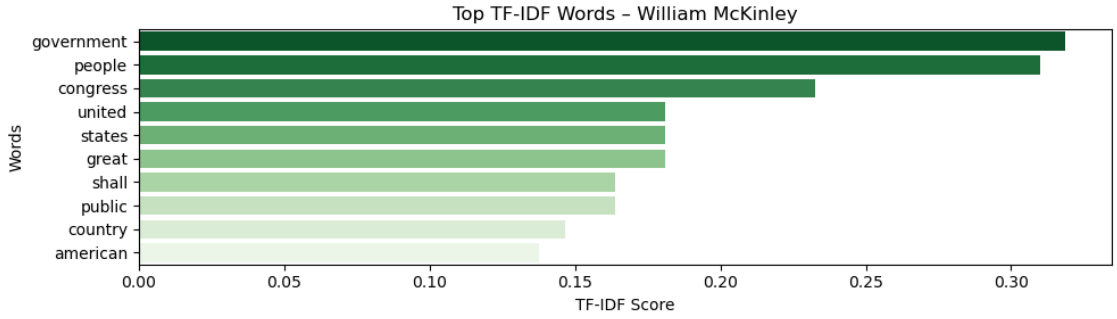

Top TF-IDF Words – Thomas Jefferson
public
citizens
fellow
government
shall
state
let
peace
constitution
good
Words
0.00
0.05
0.10
0.15
0.20
0.25
TF-IDF Score

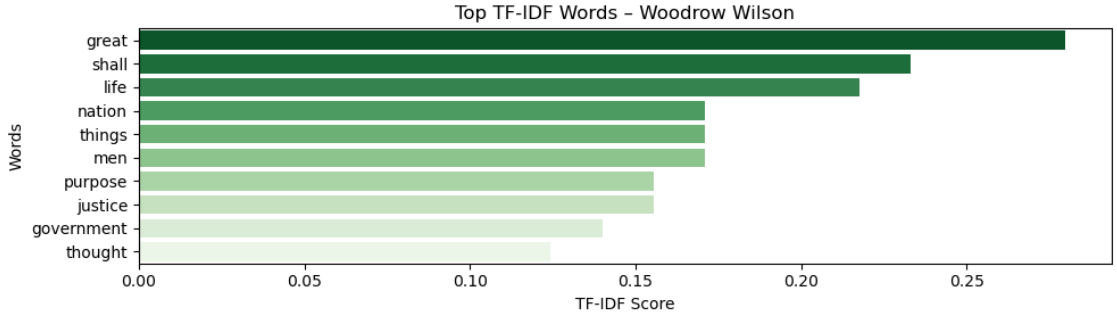

Top TF-IDF Words – Ulysses S. Grant
country
best
office
people
great
government
states
public
laws
years
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
TF-IDF Score

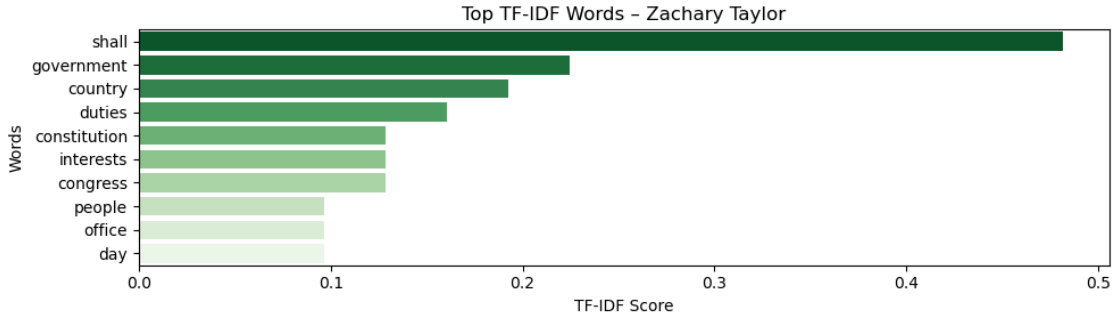

Top TF-IDF Words – Warren G. Harding
world
america
war
new
civilization
government
order
republic
american
shall
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
TF-IDF Score


Top TF-IDF Words – William Henry Harrison
power
constitution
government
executive
great
people
states
rights
president
citizens
Words
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
TF-IDF Score


Top TF-IDF Words – William Howard Taft
government
business
proper
law
congress
tariff
race
make
shall
policy
Words
0.00
0.05
0.10
0.15
0.20
0.25
TF-IDF Score







[21]:
```
# rewriters
def TalkLikeAPrez():
    print("Pick a President (case sensitive):") # the different presidents in␣
 ↪the dataset
    for name in sorted(top_w.keys()):
        print("*", name)
    prezPick = input("\nEnter a President's name: ").strip() #user input
```

```
    sentanceRew = input("Enter what you would like to be rewritten: ").strip()␣
↪# user input
    print("\n[Rewriter]") # first rewriter
    print(rewrite(sentanceRew, prezPick))
    print("\n[POS Rewriter]") # POS rewriter
    print(rewrite_pos(sentanceRew, prezPick))
```