```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string
from string import punctuation
import os
from os import listdir
from collections import Counter
from tensorflow.keras.preprocessing.text import Tokenizer
from nltk.tokenize import word_tokenize
from wordcloud import WordCloud
from textblob import TextBlob
from plotnine import *
from pandas.api.types import CategoricalDtype
from sklearn import decomposition
from sklearn import preprocessing
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,confusion_matrix,ConfusionMatrixDisplay,f1_scor
```

```
def file_to_string(filename):
 '''Opens the input text file and
 returns a string of all its text.'''
 file = open(filename, 'r')
 text = file.read()
 file.close()
 text = text.replace('\n', ' ')
 text = text.replace('  ', ' ')
 return text
```

```
cd ..
```

```
/Users/kamilapalys/Desktop/school/data450/capstone
```

```
pwd
```

```
'/Users/kamilapalys/Desktop/school/data450/capstone'
```

```
# example of using the function

filepath = 'data/text/cnn_trump.txt'
test_txt = file_to_string(filepath)
test_txt
```

"Donald Trump faces more than 30 counts related to business fraud in an indictment from a Ma
the first time in American history that a current or former president has faced criminal cha
which has never seen one of its ex-leaders confronted with criminal charges, let alone while
into uncharted waters. Trump released a statement in response to the indictment claiming it
united and strong - will first defeat Alvin Bragg, and then we will defeat Joe Biden, and we
or more - away. "Is this a shock today? Hell yes," the person said, speaking on a condition
as well as his 2024 GOP rivals - have condemned the Manhattan district attorney's office ove

```
stemmed_dict = {}
```

```
def clean_text(text, stem):
    '''Takes in a string of text cleans it by converting
    to lowercase, removing punctuation, and removing stopwords.
    Also takes in a binary value to indicate if stemming should
    be performed. Returns the new string.'''
    if stem not in [0, 1]:
        raise ValueError("Stem must be a binary value (0 or 1)")
    ps = PorterStemmer()
    stemmed_dict = {}
    # create list of stopwords
    stopwords_list = stopwords.words('english')
    # make the text lowercase
    text = text.lower()
    text = text.replace('-', ' ')
    text = text.replace('u.s.', 'us')
    # convert to ascii characters
    text = text.encode("ascii", "ignore").decode()
    for chr in text:
        # only keep characters in the string that are not punctuation symbols
        if (chr in string.punctuation or chr in string.digits):
            text = text.replace(chr, ' ')
    text = text.replace('  ', ' ')
    # stem the tokens within the text
    tokens = text.split()
```

```
    new_tokens = []
    for token in tokens[:-2]:
        # only include new token in the cleaned list if not a stopword
        if token not in stopwords_list:
            if stem == 1:
                stemmed_word = ps.stem(token)
                new_tokens.append(stemmed_word)
                # to be able to map each token to the resulting stemmed word
                if token not in stemmed_dict:
                    stemmed_dict[token] = stemmed_word
            else:
                new_tokens.append(token)
    new_tokens.append(tokens[-2])
    new_tokens.append(tokens[-1])
    cleaned_text = " ".join(new_tokens)
    cleaned_text = cleaned_text.replace('  ', ' ')
    return cleaned_text
```

```
# looping through all text files to apply preprocessing functions
article_docs = []
dir = os.listdir('data/text/')
dir.sort()
for filename in dir:
    filepath = os.path.join('data/text/', filename)
    if filename.split(".")[-1] == "txt":
        article_string = file_to_string(filepath)
        new_string = clean_text(article_string, 1)
        article_docs.append(new_string)

# convert the list of article strings into a binary-value dataframe
t = Tokenizer()
t.fit_on_texts(article_docs)
print(t)
encoded_docs = t.texts_to_matrix(article_docs, mode='binary')
words = [x for x in t.word_index.keys()]
binary_df = pd.DataFrame(data = encoded_docs[:, 1:], columns=words)
# List of conditions
source_conditions = [
      binary_df['abcarticle'] == 1
    , binary_df['bbcarticle'] == 1
    , binary_df['cnnarticle'] == 1
```

```
    , binary_df['foxarticle'] == 1
    , binary_df['nbcarticle'] == 1
    , binary_df['nyparticle'] == 1
    , binary_df['nytarticle'] == 1
    , binary_df['wparticle'] == 1
    , binary_df['wsjarticle'] == 1
]

# List of values to return
source_choices  = [
      "ABC News"
    , "BBC"
    , "CNN"
    , "Fox News"
    , "NBC News"
    , "New York Post"
    , "The New York Times"
    , "The Washington Post"
    , "The Wall Street Journal"
]

# List of conditions
topic_conditions = [
      binary_df['affirmativearticle'] == 1
    , binary_df['balloonarticle'] == 1
    , binary_df['bidenarticle'] == 1
    , binary_df['hamasarticle'] == 1
    , binary_df['pentagonarticle'] == 1
    , binary_df['santosarticle'] == 1
    , binary_df['tanksarticle'] == 1
    , binary_df['trumparticle'] == 1
]
# List of values to return
topic_choices  = [
      "Supreme Court Ruling on Affirmative Action"
    , "Chinese Surveillance Balloon"
    , "Biden's Low Approval Rates in Polls"
    , "The Deadliest Attack by Hamas"
    , "Pentagon Documents Leak"
    , "George Santos' Expulsion from Congress"
    , "U.S. and Germany Send Tanks to Ukraine"
```

```
    , "Trump's Indictment"
]
# create a new source column
binary_df["article_source"] = np.select(source_conditions, source_choices, "ERROR")

# create a new topic column
binary_df["article_topic"] = np.select(topic_conditions, topic_choices, "ERROR")

binary_df.head()
```

```
<keras.src.preprocessing.text.Tokenizer object at 0x179c70400>
```

	said	us	trump	biden	offici	mr	israel	presid	tank	hous	...	messr	overlap	vs	convert
0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
2	1.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
3	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
4	1.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0

```
# check what word a given stemmed word represents
# value = [i for i in stemmed_dict if stemmed_dict[i]=="affirm"]
# print("key by value:",value)
```

```
key by value: []
```

```
article_docs
```

```
['suprem court thursday set new limit affirm action program case involv whether public privat
 'massiv spi balloon believ china seen montana track fli across continent unit state presid
 'two day dismal new poll number presid joe biden show public view unfavor rival donald trump
 'hundr peopl israel report dead thousand injur hama milit fire rocket gaza launch ground inc
 'post social media appear sever highli classifi us intellig document might begin could turn
 'hous repres friday vote expel republican rep georg santo histor move happen year santo scar
 'major increas us support ukrain presid joe biden sign send abram tank war torn countri conc
 'manhattan grand juri indict former presid donald trump make first current former presid fac
 'us suprem court rule race longer consid factor univers admiss landmark rule upend decad old
 'us track suspect chines surveil balloon spot fli sensit site recent day defenc offici said
```

'us presid joe biden run elect next year nation opinion poll weak job approv rate suggest v
'least peopl report kill wound israel palestinian milit group hama launch biggest attack yea
'document includ detail account train provid ukrain foreign power make dozen classifi us de
'us hous repres expel congressman georg santo follow damn ethic report dozen crimin charg h
'us send power battl tank ukrain join germani send vehicl support fight russia invas decis
'former us presid donald trump charg hush money payment made porn star presidenti elect deta
'suprem court say colleg univers longer take race consider specif basi grant admiss landmar
'us track suspect chines high altitud surveil balloon continent unit state defens offici sa
'one third regist voter approv presid joe biden handl isra palestinian conflict new poll ne
'gaza jerusalem cnn israel prime minist benjamin netanyahu declar countri war saturday pale
'man arrest fbi connect massiv us classifi document leak charg boston friday unauthor reten
'hous vote friday expel gop rep georg santo histor vote make new york congressman sixth law
'leader unit state germani announc wednesday send conting tank ukrain revers longstand trep
'donald trump face count relat busi fraud indict manhattan grand juri accord two sourc fami
'us suprem court hand major rule affirm action thursday reject use race factor colleg admis
'us govern monitor suspect chines surveil balloon move northern state past sever day pentag
'biden battl rough poll number fox news white hous correspond peter dooci latest presid ele
'hama widespread coordin attack may suggest outsid help trey yingst foreign correspond trey
'bret baier intel suspect year old govern worker special report bret baier anchor question
'hous repres vote expel scandal plagu rep georg santo r n friday make first hous lawmak exp
'german chancellor olaf scholz formal announc wednesday week stall frustrat negoti berlin a
'former presid donald trump indict part manhattan district attorney offic year long investi
'washington suprem court thursday struck affirm action program univers north carolina harva
'us militari monitor suspect chines surveil balloon hover northern us past day militari def
'differ poll agre presid joe biden polit stand lower barack obama point elect even lower do
'ashkelon israel israel plung chao saturday palestinian milit group hama launch deadli land
'dozen leak defens depart classifi document post onlin reveal detail us spi russia war mach
'washington hous vote overwhelmingli expel indict rep georg santo friday pull curtain tempe
'ukrain set receiv battl tank germani western countri fierc debat expos fissur among alli a
'grand juri new york citi vote thursday indict donald trump first time former us presid fac
'suprem court struck affirm action program harvard univers univers north carolina thursday
'us track chines spi balloon float northern part countri day pentagon offici announc thursd
'presid biden readi ring new year see number year old command chief end lower approv rate s
'jerusalem ap hama milit fire thousand rocket sent dozen fighter isra town near gaza strip
'massachusett air nation guardsman jack teixeira leader discord group dozen sensit us intel
'bye georg lie long island rep georg santo r ny becam sixth member ever expel us hous repre
'week excus foot drag german final said ye transfer limit number leopard tank ukrain agre w
'donald trump indict manhattan grand juri thursday hush money payment made ahead elect mark
'chief justic john g robert jr finish read major opinion suprem court chamber thursday hush
'helena mont larri mayer newspap photograph point camera sky wednesday began snap pictur ap
'democrat battleground state grow increasingli anxiou presid biden low approv rate worri vo
'israel news site compil list dead miss funer take place around countri weekend attack conf
'washington would year old nation guardsman posit access top secret document begin dramat a

```
'georg santo new york republican congressman whose tapestri lie scheme made figur nation ric
'precis militari drill first germani unit state announc wednesday agre provid battl tank hel
'manhattan grand juri indict donald j trump thursday role pay hush money porn star accord pe
'suprem court thursday held race consciou admiss program harvard univers north carolina viol
'chines surveil balloon collect intellig continent unit state right us offici disclos thursd
'night presid biden depart washington celebr thanksgiv nantucket mass gather closest aid mee
'sderot israel israel formal declar war palestinian milit group hama sunday reel surpris att
'saturday us offici foreign alli scrambl understand dozen classifi intellig document end int
'hous vote friday expel rep georg santo r n congress action chamber previous taken five time
'biden administr announc wednesday send premier battl tank ukrain follow agreement germani c
'new york manhattan grand juri vote indict former presid donald trump make first person us h
'thursday decis forc rework admiss criteria throughout american higher educ decad pursuit di
'washington us track offici describ chines reconnaiss balloon continent state week would agg
'mr biden low stand head head gener elect poll reflect voter dour apprais perform presid sur
'tel aviv isra prime minist benjamin netanyahu said countri war hama milit group forc pour a
'crimin case unfold us govern scrambl protect secret unauthor disclosur appear provid detail
'lawmak vote remov two third hous supermajor requir constitut almost democrat mani republica
'us germani outlin plan wednesday send dozen modern battl tank ukrain mark signific new infu
'grand juri return indict mr trump vote thursday kick process former presid expect come new
```

```
encoded_docs_freq = t.texts_to_matrix(article_docs, mode='count')
freq_df = pd.DataFrame(data = encoded_docs_freq[:, 1:], columns=words)
# List of conditions
source_conditions = [
      freq_df['abcarticle'] == 1
    , freq_df['bbcarticle'] == 1
    , freq_df['cnnarticle'] == 1
    , freq_df['foxarticle'] == 1
    , freq_df['nbcarticle'] == 1
    , freq_df['nyparticle'] == 1
    , freq_df['nytarticle'] == 1
    , freq_df['wparticle'] == 1
    , freq_df['wsjarticle'] == 1
]

# List of values to return
source_choices  = [
      "ABC News"
    , "BBC"
    , "CNN"
    , "Fox News"
```

```
    , "NBC News"
    , "New York Post"
    , "The New York Times"
    , "The Washington Post"
    , "The Wall Street Journal"
]

# List of conditions
topic_conditions = [
      freq_df['affirmativearticle'] == 1
    , freq_df['balloonarticle'] == 1
    , freq_df['bidenarticle'] == 1
    , freq_df['hamasarticle'] == 1
    , freq_df['pentagonarticle'] == 1
    , freq_df['santosarticle'] == 1
    , freq_df['tanksarticle'] == 1
    , freq_df['trumparticle'] == 1
]
# List of values to return
topic_choices  = [
      "Supreme Court Ruling on Affirmative Action"
    , "Chinese Surveillance Balloon"
    , "Biden's Low Approval Rates in Polls"
    , "The Deadliest Attack by Hamas"
    , "Pentagon Documents Leak"
    , "George Santos' Expulsion from Congress"
    , "U.S. and Germany Send Tanks to Ukraine"
    , "Trump's Indictment"
]
# create a new source column
freq_df["article_source"] = np.select(source_conditions, source_choices, "ERROR")

# create a new topic column
freq_df["article_topic"] = np.select(topic_conditions, topic_choices, "ERROR")

freq_df.head()
```

	said	us	trump	biden	offici	mr	israel	presid	tank	hous	...	messr	overlap	vs	conver
0	5.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	30.0	15.0	0.0	5.0	28.0	0.0	0.0	4.0	0.0	3.0	...	0.0	0.0	0.0	0.0
2	9.0	0.0	17.0	27.0	0.0	0.0	0.0	8.0	0.0	4.0	...	0.0	0.0	0.0	0.0

	said	us	trump	biden	offici	mr	israel	presid	tank	hous	...	messr	overlap	vs	conver
3	17.0	10.0	0.0	3.0	6.0	0.0	28.0	2.0	0.0	1.0	...	0.0	0.0	0.0	0.0
4	9.0	20.0	0.0	0.0	5.0	0.0	2.0	2.0	3.0	0.0	...	0.0	0.0	0.0	0.0

```
# create dataframe with tf-idf values

encoded_docs_tfidf = t.texts_to_matrix(article_docs, mode='tfidf')
tfidf_df = pd.DataFrame(data = encoded_docs_tfidf[:, 1:], columns=words)
# List of conditions
source_conditions = [
      tfidf_df['abcarticle'] != 0
    , tfidf_df['bbcarticle'] != 0
    , tfidf_df['cnnarticle'] != 0
    , tfidf_df['foxarticle'] != 0
    , tfidf_df['nbcarticle'] != 0
    , tfidf_df['nyparticle'] != 0
    , tfidf_df['nytarticle'] != 0
    , tfidf_df['wparticle'] != 0
    , tfidf_df['wsjarticle'] != 0
]

# List of values to return
source_choices  = [
      "ABC News"
    , "BBC"
    , "CNN"
    , "Fox News"
    , "NBC News"
    , "New York Post"
    , "The New York Times"
    , "The Washington Post"
    , "The Wall Street Journal"
]

# List of conditions
topic_conditions = [
      tfidf_df['affirmativearticle'] != 0
    , tfidf_df['balloonarticle'] != 0
    , tfidf_df['bidenarticle'] != 0
    , tfidf_df['hamasarticle'] != 0
```

```
    , tfidf_df['pentagonarticle'] != 0
    , tfidf_df['santosarticle'] != 0
    , tfidf_df['tanksarticle'] != 0
    , tfidf_df['trumparticle'] != 0
]
# List of values to return
topic_choices  = [
      "Supreme Court Ruling on Affirmative Action"
    , "Chinese Surveillance Balloon"
    , "Biden's Low Approval Rates in Polls"
    , "The Deadliest Attack by Hamas"
    , "Pentagon Documents Leak"
    , "George Santos' Expulsion from Congress"
    , "U.S. and Germany Send Tanks to Ukraine"
    , "Trump's Indictment"
]
# create a new source column
tfidf_df["article_source"] = np.select(source_conditions, source_choices, "ERROR")

# create a new topic column
tfidf_df["article_topic"] = np.select(topic_conditions, topic_choices, "ERROR")

tfidf_df.head()
```

	said	us	trump	biden	offici	mr	israel	presid	tank	hous	..
0	1.827036	1.839130	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	..
1	3.081563	2.857814	0.000000	2.267700	4.139471	0.0	0.000000	1.881491	0.000000	1.871958	..
2	2.238584	0.000000	5.086179	3.733245	0.000000	0.0	0.000000	2.428008	0.000000	2.128570	..
3	2.683881	2.545320	0.000000	1.823774	2.667558	0.0	6.446654	1.334974	0.000000	0.891998	..
4	2.238584	3.079532	0.000000	0.000000	2.493348	0.0	2.519533	1.334974	3.689062	0.000000	..

```
# create a list of strings where each string is all articles from one source (for dendogra
source_docs = []

j = 0

for i in range(9):
    # looping through each article of each source, and only taking up until and not includ
    # token, bc last two tokens represent the name of the source and topic of article
    # for last article, we index up until and not including only the last token bc we don'
```

```
    # the topic of the article, but we do want to include the source name, which is the se
    source = " ".join(article_docs[j].split()[:-2]) + " " + " ".join(article_docs[j+1].spl
        + " ".join(article_docs[j+2].split()[:-2]) + " " + " ".join(article_docs[j+3].spli
        + " ".join(article_docs[j+4].split()[:-2]) + " " + " ".join(article_docs[j+5].spli
        + " ".join(article_docs[j+6].split()[:-2]) + " " + " ".join(article_docs[j+7].spli
    source_docs.append(source)
    j += 8

source_docs
```

```
['suprem court thursday set new limit affirm action program case involv whether public privat
 'us suprem court rule race longer consid factor univers admiss landmark rule upend decad old
 'suprem court say colleg univers longer take race consider specif basi grant admiss landmark
 'us suprem court hand major rule affirm action thursday reject use race factor colleg admiss
 'washington suprem court thursday struck affirm action program univers north carolina harvar
 'suprem court struck affirm action program harvard univers univers north carolina thursday r
 'chief justic john g robert jr finish read major opinion suprem court chamber thursday hush
 'suprem court thursday held race consciou admiss program harvard univers north carolina viol
 'thursday decis forc rework admiss criteria throughout american higher educ decad pursuit di
```

```
for source in source_docs:
    for i in range(len(source)-2):
        if source[i] + source[i+1] + source[i+2] == ' u ':
            print(source[i-10:i+100], source)
```

```
en countri u k also commit challeng battl tank poland ask germani permiss send leopard stock
gia meloni u k prime minist rishi sunak grate chancellor scholz provid german leopard tank le
ear accord u n mideast envoy tor wennesland israel say raid aim milit stone throw protest pec
gust brief u n secur council surpris hama offens come year day israel arab neighbor launch co
h mile ton u k l e mm rifl gun two mm machin gun us offici said abram would enough ukrainian
```

```
# create a dataframe of token tf-idf's with each row representing all articles of one sour

# convert the list of article strings into a tf-idf-value dataframe
t = Tokenizer()
t.fit_on_texts(source_docs)
print(t)
encoded_source_docs = t.texts_to_matrix(source_docs, mode='tfidf')
```

```
words = [x for x in t.word_index.keys()]
tfidf_source_df = pd.DataFrame(data = encoded_source_docs[:, 1:], columns=words)
# List of conditions
source_conditions = [
      tfidf_source_df['abcarticle'] != 0
    , tfidf_source_df['bbcarticle'] != 0
    , tfidf_source_df['cnnarticle'] != 0
    , tfidf_source_df['foxarticle'] != 0
    , tfidf_source_df['nbcarticle'] != 0
    , tfidf_source_df['nyparticle'] != 0
    , tfidf_source_df['nytarticle'] != 0
    , tfidf_source_df['wparticle'] != 0
    , tfidf_source_df['wsjarticle'] != 0
]

# List of values to return
source_choices  = [
      "ABC News"
    , "BBC"
    , "CNN"
    , "Fox News"
    , "NBC News"
    , "New York Post"
    , "The New York Times"
    , "The Washington Post"
    , "The Wall Street Journal"
]

# create a new source column
tfidf_source_df["article_source"] = np.select(source_conditions, source_choices, "ERROR")
tfidf_source_df.set_index('article_source', inplace=True) # removes article_source column
tfidf_source_df.drop(['abcarticle', 'bbcarticle', 'cnnarticle', 'foxarticle',
                      'nbcarticle', 'nyparticle', 'nytarticle', 'wparticle',
                      'wsjarticle'], axis=1, inplace=True)
tfidf_source_df
```

```
<keras.src.preprocessing.text.Tokenizer object at 0x179ad0fa0>
```

article_source	said	us	trump	biden	offici	mr	israel	presid
ABC News	3.591249	3.331000	2.959536	3.099282	3.139834	1.173600	2.824926	2.8861
BBC	3.280434	3.165511	2.780642	2.335743	2.460363	3.283902	2.460363	2.6543
CNN	3.698474	3.446401	3.025425	2.976653	3.331000	0.693147	3.225542	2.9933
Fox News	3.055995	2.866350	3.350161	2.824926	2.421451	0.693147	2.866350	2.9052
NBC News	3.670441	3.213976	3.311249	2.866350	2.654383	1.654053	3.099282	2.5317
New York Post	3.340652	3.202199	3.126599	2.757300	2.335743	1.654053	3.085175	3.0095
The New York Times	3.584733	2.052151	3.269825	2.993325	2.654383	4.166255	1.347002	2.8249
The Washington Post	3.785551	3.202199	3.269825	3.269825	3.225542	0.000000	2.976653	3.0992
The Wall Street Journal	3.849334	3.421552	3.213976	2.866350	3.126599	3.919027	2.905262	2.8031

```
# create a dataframe of token binary values with each row representing all articles of one

# convert the list of article strings into a binary-value dataframe
t = Tokenizer()
t.fit_on_texts(source_docs)
print(t)
encoded_source_docs = t.texts_to_matrix(source_docs, mode='binary')
words = [x for x in t.word_index.keys()]
binary_source_df = pd.DataFrame(data = encoded_source_docs[:, 1:], columns=words)
# List of conditions
source_conditions = [
      binary_source_df['abcarticle'] == 1
    , binary_source_df['bbcarticle'] == 1
    , binary_source_df['cnnarticle'] == 1
    , binary_source_df['foxarticle'] == 1
    , binary_source_df['nbcarticle'] == 1
    , binary_source_df['nyparticle'] == 1
    , binary_source_df['nytarticle'] == 1
    , binary_source_df['wparticle'] == 1
    , binary_source_df['wsjarticle'] == 1
]

# List of values to return
source_choices  = [
      "ABC News"
    , "BBC"
    , "CNN"
    , "Fox News"
```

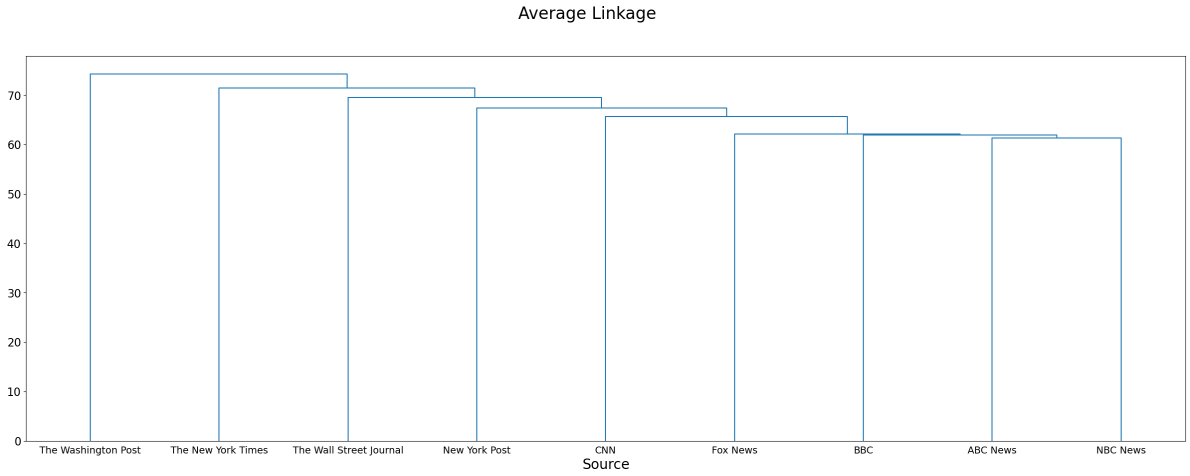
```
    , "NBC News"
    , "New York Post"
    , "The New York Times"
    , "The Washington Post"
    , "The Wall Street Journal"
]

# create a new source column
binary_source_df["article_source"] = np.select(source_conditions, source_choices, "ERROR")
binary_source_df.set_index('article_source', inplace=True)
binary_source_df.drop(['abcarticle', 'bbcarticle', 'cnnarticle', 'foxarticle',
                       'nbcarticle', 'nyparticle', 'nytarticle', 'wparticle',
                       'wsjarticle'], axis=1, inplace=True)
binary_source_df
```

```
<keras.src.preprocessing.text.Tokenizer object at 0x111bc58e0>
```

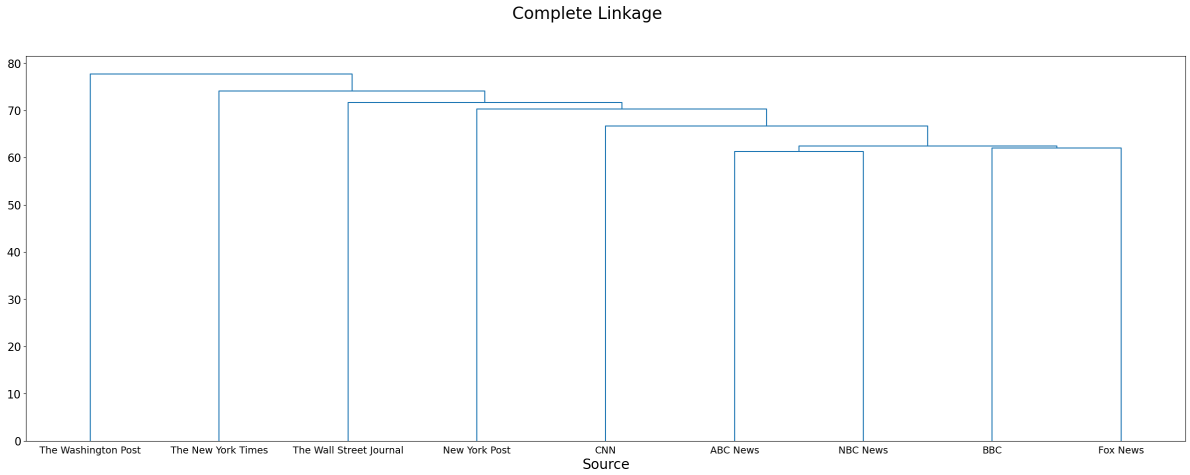
article_source	said	us	trump	biden	offici	mr	israel	presid	tank	hous	...	squelch
ABC News	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0
BBC	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0
CNN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0
Fox News	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0
NBC News	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0
New York Post	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0
The New York Times	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0
The Washington Post	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	...	0.0
The Wall Street Journal	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	1.0

```
# create dendrogram with average linkage from tf_idf scores df, whose rows each represent

from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

Z = linkage(tfidf_source_df, 'average')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Average Linkage", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
```

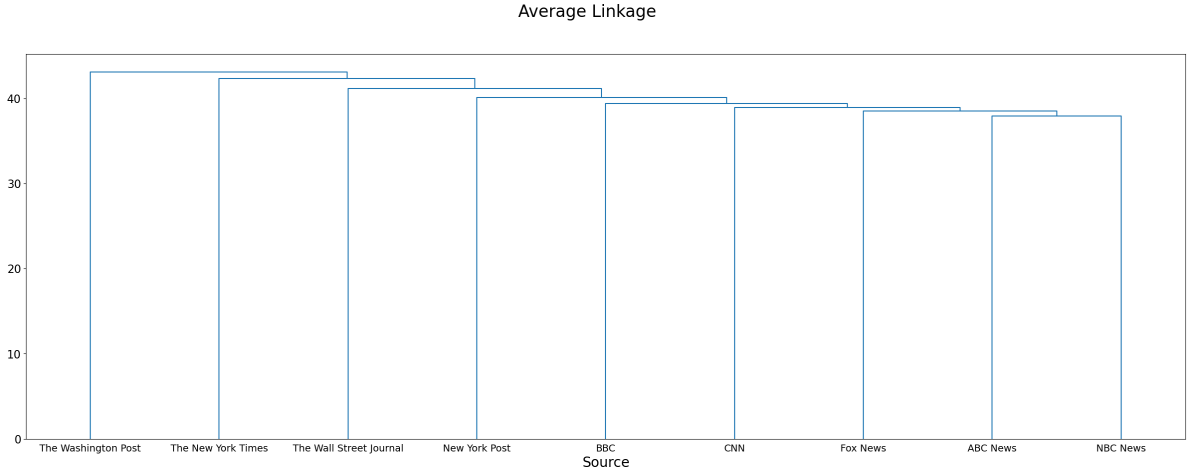
```
dn = dendrogram(Z, labels=tfidf_source_df.index)
plt.xticks(fontsize = 14)
plt.show()
```



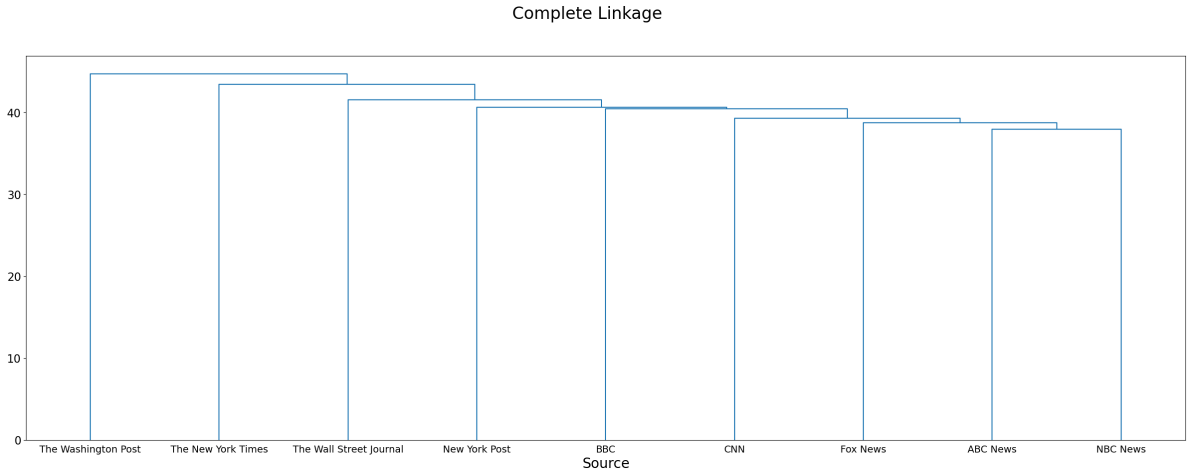

```
# create dendrogram with complete linkage from tf_idf scores df, whose rows each represent

Z = linkage(tfidf_source_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=tfidf_source_df.index)
plt.xticks(fontsize = 14)
plt.show()
```




```
# create dendrogram with average linkage from binary scores df, whose rows each represent

Z = linkage(binary_source_df, 'average')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Average Linkage", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=binary_source_df.index)
plt.xticks(fontsize = 14)
plt.show()
```



```
# create dendrogram with complete linkage from binary scores df, whose rows each represent

Z = linkage(binary_source_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=binary_source_df.index)
plt.xticks(fontsize = 14)
plt.show()
```




```
# create dataframes, each containing articles of one topic for dendrograms for each topic

import warnings
warnings.filterwarnings('ignore')

tfidf_df_dropped = tfidf_df.drop(['abcarticle', 'bbcarticle', 'cnnarticle', 'foxarticle',
                      'nbcarticle', 'nyparticle', 'nytarticle', 'wparticle',
                      'wsjarticle', 'affirmativearticle', 'balloonarticle',
                      'bidenarticle', 'hamasarticle', 'pentagonarticle',
                      'santosarticle', 'tanksarticle', 'trumparticle'],
                      axis=1, inplace=False)

affirm_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "Supreme Court Rul
affirm_tfidf_df.set_index('article_source', inplace=True)
affirm_tfidf_df.drop(['article_topic'], axis=1, inplace=True)
```

```
balloon_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "Chinese Surveill
balloon_tfidf_df.set_index('article_source', inplace=True)
balloon_tfidf_df.drop(['article_topic'], axis=1, inplace=True)

biden_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "Biden's Low Approv
biden_tfidf_df.set_index('article_source', inplace=True)
biden_tfidf_df.drop(['article_topic'], axis=1, inplace=True)

hamas_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "The Deadliest Atta
hamas_tfidf_df.set_index('article_source', inplace=True)
hamas_tfidf_df.drop(['article_topic'], axis=1, inplace=True)

pentagon_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "Pentagon Docume
pentagon_tfidf_df.set_index('article_source', inplace=True)
pentagon_tfidf_df.drop(['article_topic'], axis=1, inplace=True)

santos_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "George Santos' Ex
santos_tfidf_df.set_index('article_source', inplace=True)
santos_tfidf_df.drop(['article_topic'], axis=1, inplace=True)

tanks_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "U.S. and Germany S
tanks_tfidf_df.set_index('article_source', inplace=True)
tanks_tfidf_df.drop(['article_topic'], axis=1, inplace=True)

trump_tfidf_df = tfidf_df_dropped[tfidf_df_dropped['article_topic'] == "Trump's Indictment
trump_tfidf_df.set_index('article_source', inplace=True)
trump_tfidf_df.drop(['article_topic'], axis=1, inplace=True)
```

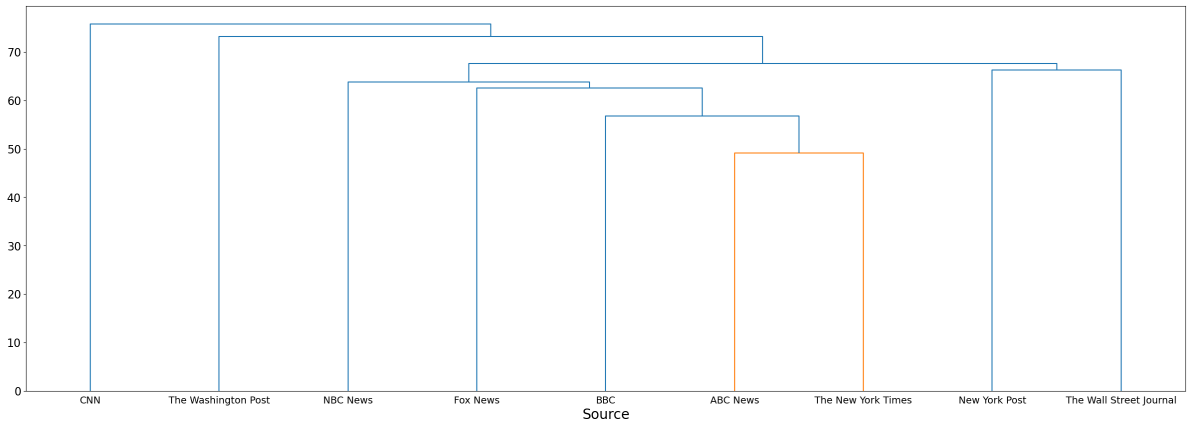
```
affirm_tfidf_df.head()
```

article_source	said	us	trump	biden	offici	mr	israel	presid	tank	hous
ABC News	1.827036	1.839130	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.000
BBC	2.379087	2.618776	1.326871	0.869038	0.0	2.295628	0.0	2.057431	0.0	0.891
CNN	2.641434	2.151624	2.784588	0.000000	0.0	0.000000	0.0	1.334974	0.0	0.891
Fox News	1.827036	1.304918	0.000000	0.869038	0.0	0.000000	0.0	0.788457	0.0	0.000
NBC News	2.156116	1.617412	2.246588	2.073780	0.0	0.000000	0.0	1.881491	0.0	0.000

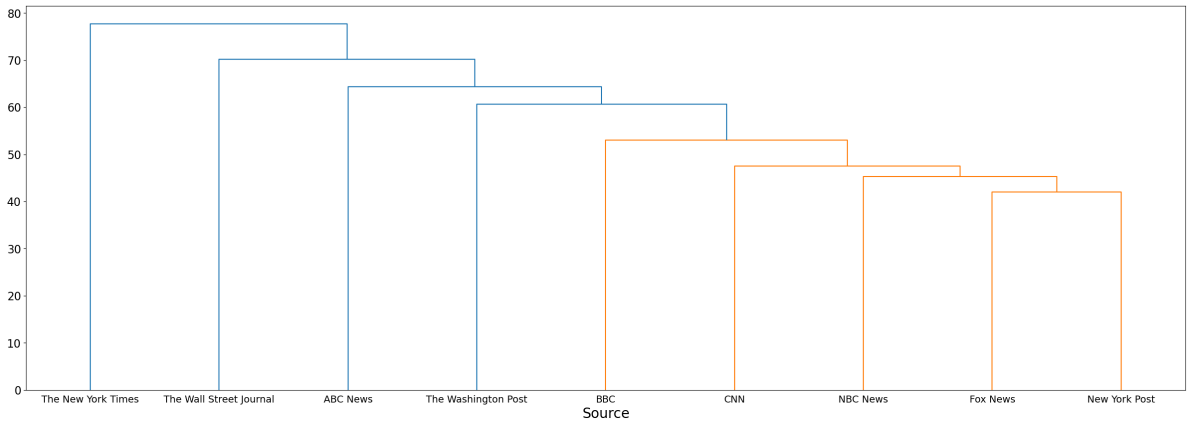
```
# create dendrograms with complete linkage from tfidf scores df's, each one representing a

Z = linkage(affirm_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of Affirmative Action Articles", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=affirm_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()

Z = linkage(balloon_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of Chinese Surveillance Balloon Articles", fonts
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=balloon_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()

Z = linkage(biden_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of Biden's Low Ratings Articles", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=biden_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()

Z = linkage(hamas_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of Hamas Attack Articles", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=hamas_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()

Z = linkage(pentagon_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of Pentagon Document Leak Articles", fontsize=24
```

```
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=pentagon_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()

Z = linkage(santos_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of George Santos Expulsion Articles", fontsize=2
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=santos_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()

Z = linkage(tanks_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of Sending Ukraine Tanks Articles", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=tanks_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()

Z = linkage(trump_tfidf_df, 'complete')
fig = plt.figure(figsize=(30, 10))
fig.suptitle("Complete Linkage Clustering of Trump's Indictment Articles", fontsize=24)
plt.xlabel('Source', fontsize=20)
plt.yticks(fontsize = 16)
dn = dendrogram(Z, labels=trump_tfidf_df.index)
plt.xticks(fontsize = 14)
plt.show()
```

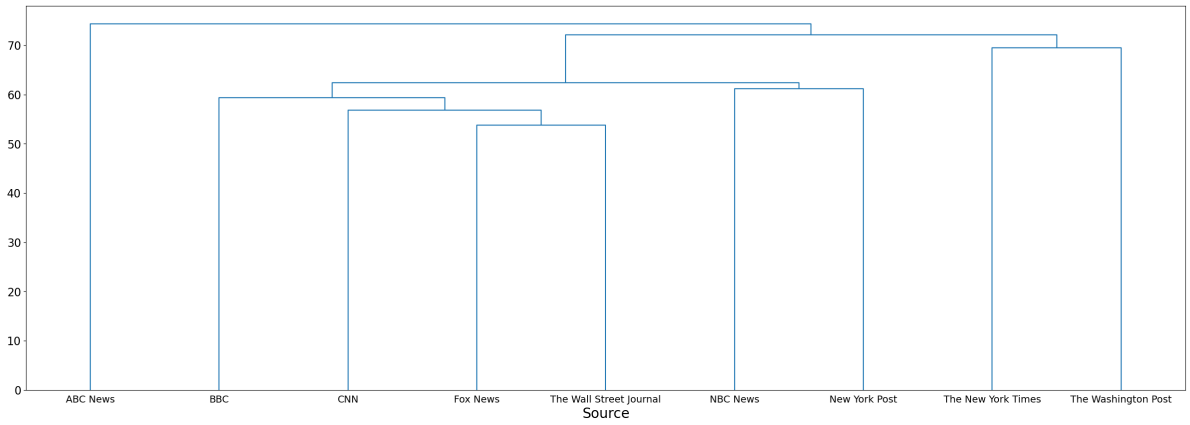
Complete Linkage Clustering of Affirmative Action Articles



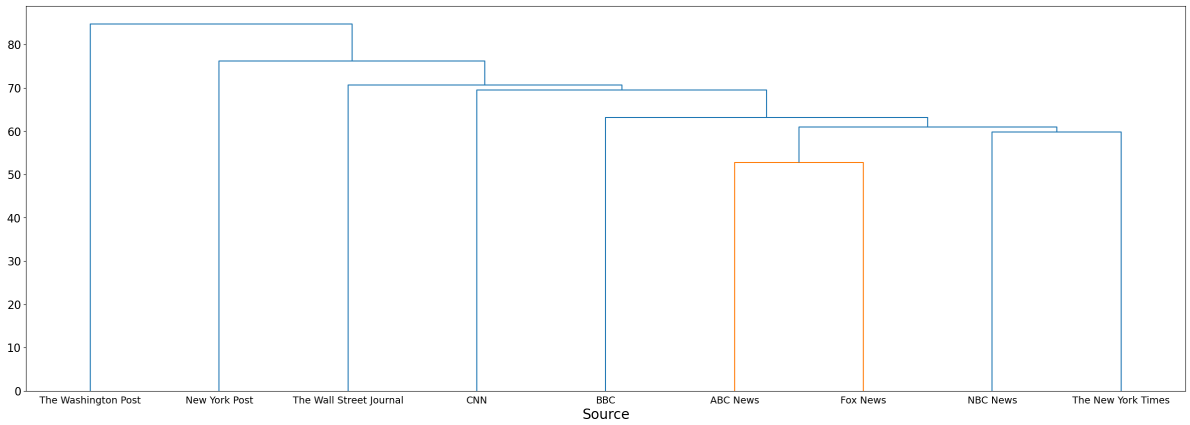
Complete Linkage Clustering of Chinese Surveillance Balloon Articles



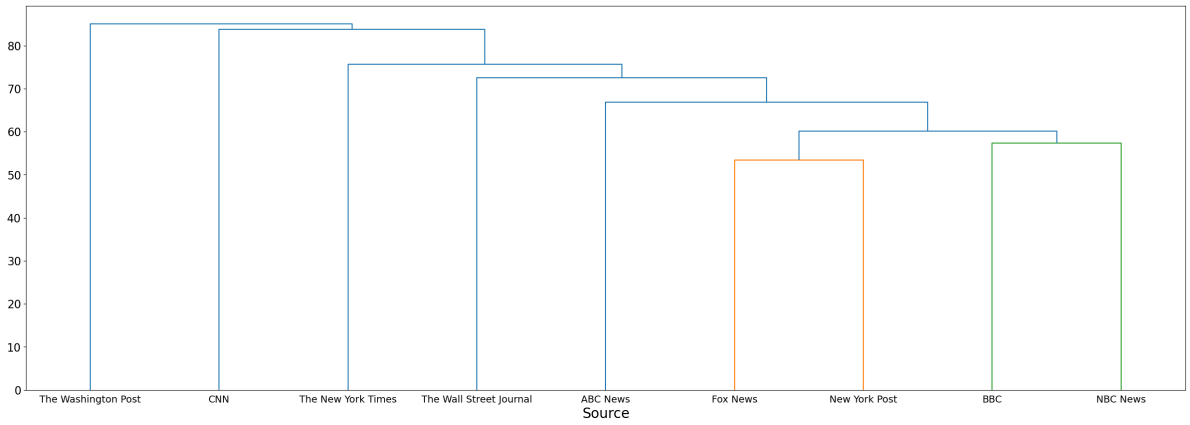
Complete Linkage Clustering of Biden's Low Ratings Articles



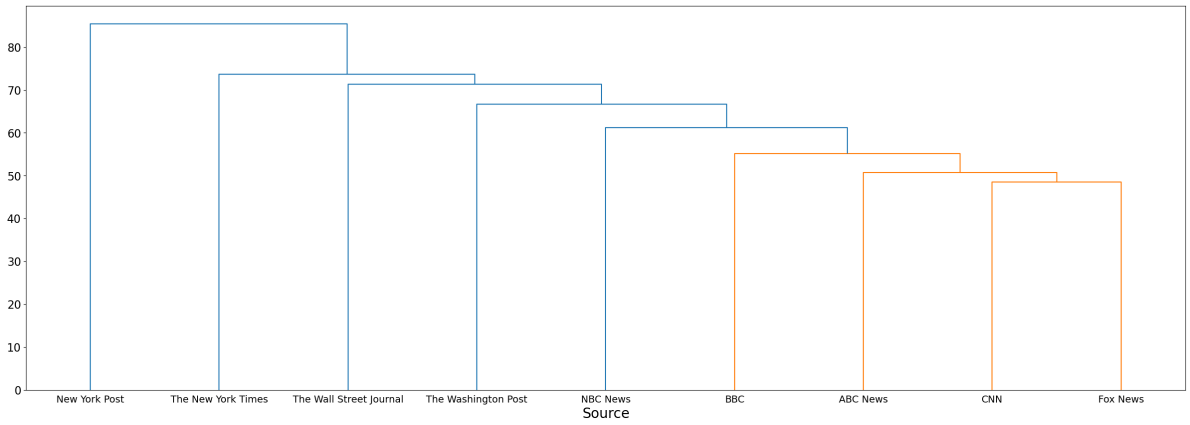
Complete Linkage Clustering of Hamas Attack Articles

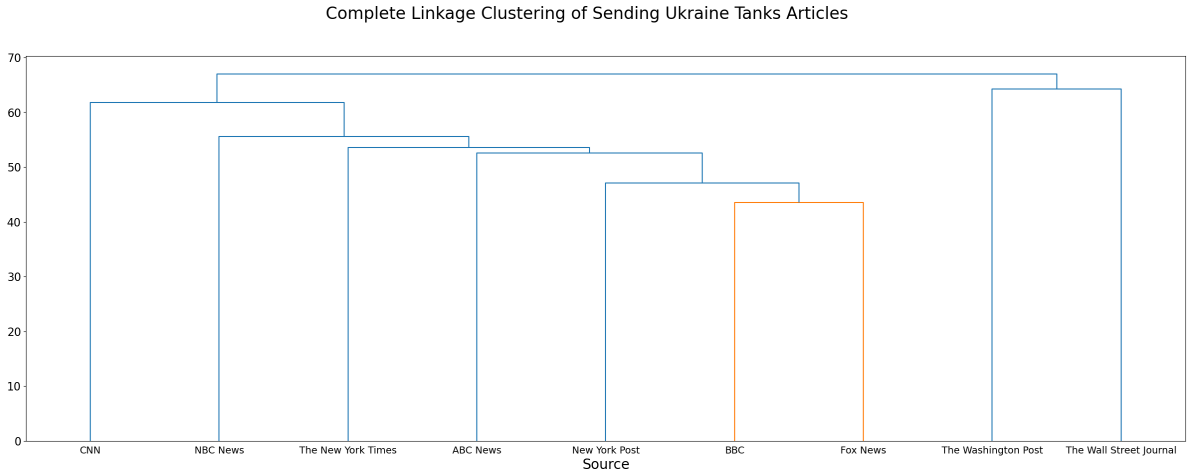


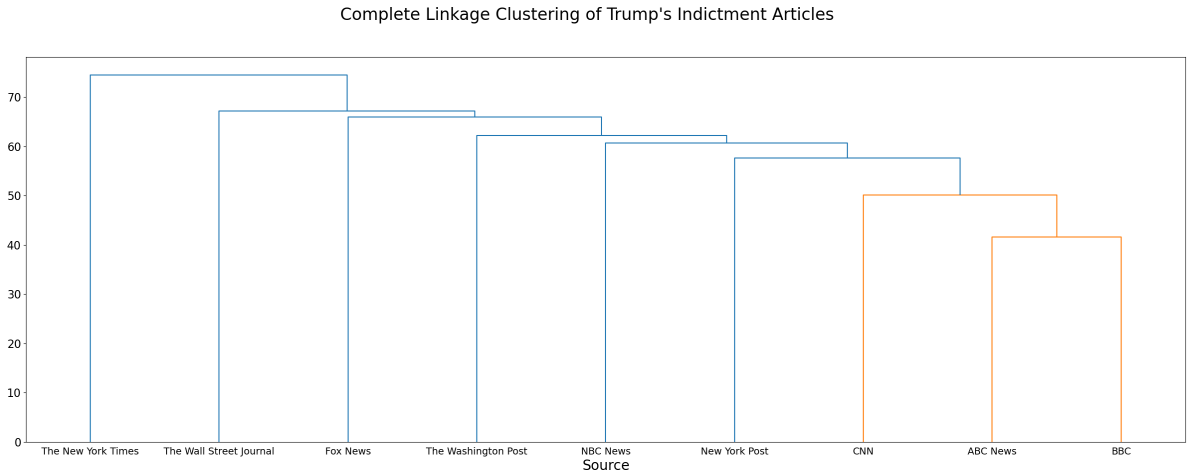
Complete Linkage Clustering of Pentagon Document Leak Articles



Complete Linkage Clustering of George Santos Expulsion Articles





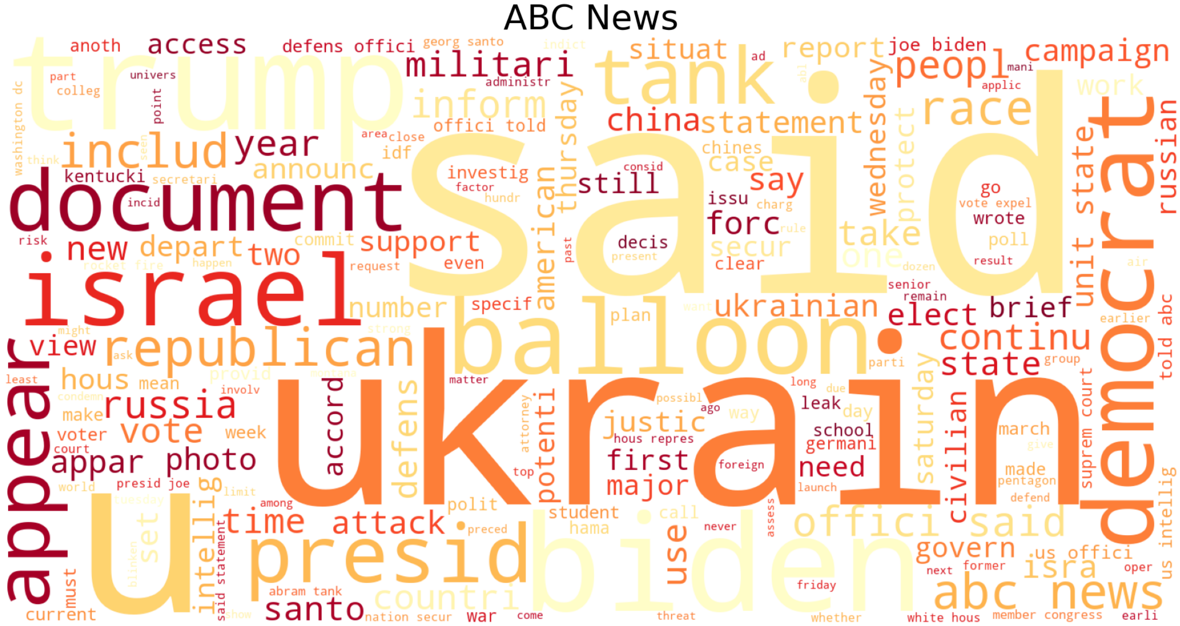



```
# other color themes i liked: magma, PiYG, RdPu (purple/pink), PuRd_r (pink), PRGn, YlOrRd

def create_wordcloud(text, title):
    '''Given a string of all text and a string
    for the title, creates a wordcloud.'''
    plt.figure(figsize = (20,10))
    wc = WordCloud(colormap = "YlOrRd", background_color='white', width=1600, height=800,
    plt.title(title, fontsize=40)
    plt.axis("off")
    title_snake = title.lower().replace(" ", "_")
    wc.to_file(f'wordcloud_{title_snake}.png')
    #plt.savefig(f'wordcloud_{title_snake}.png', dpi = 1000)
```

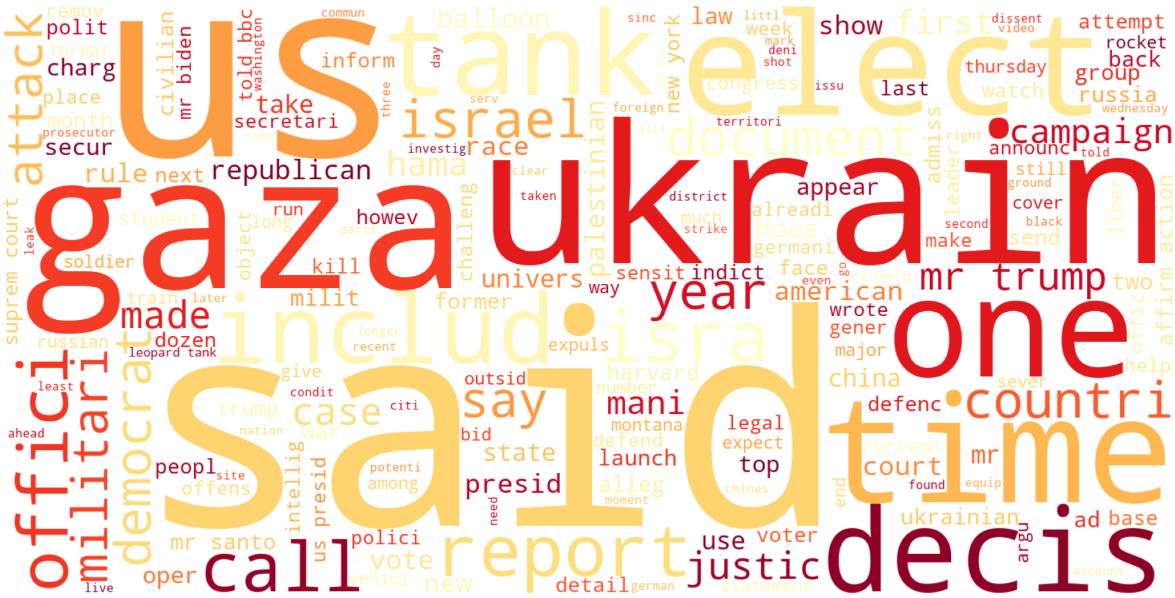
```
    plt.tight_layout(pad=0)
    plt.imshow(wc)
```

```
for i in range(len(source_docs)):
    create_wordcloud(source_docs[i], tfidf_source_df.index[i])
```

ABC News

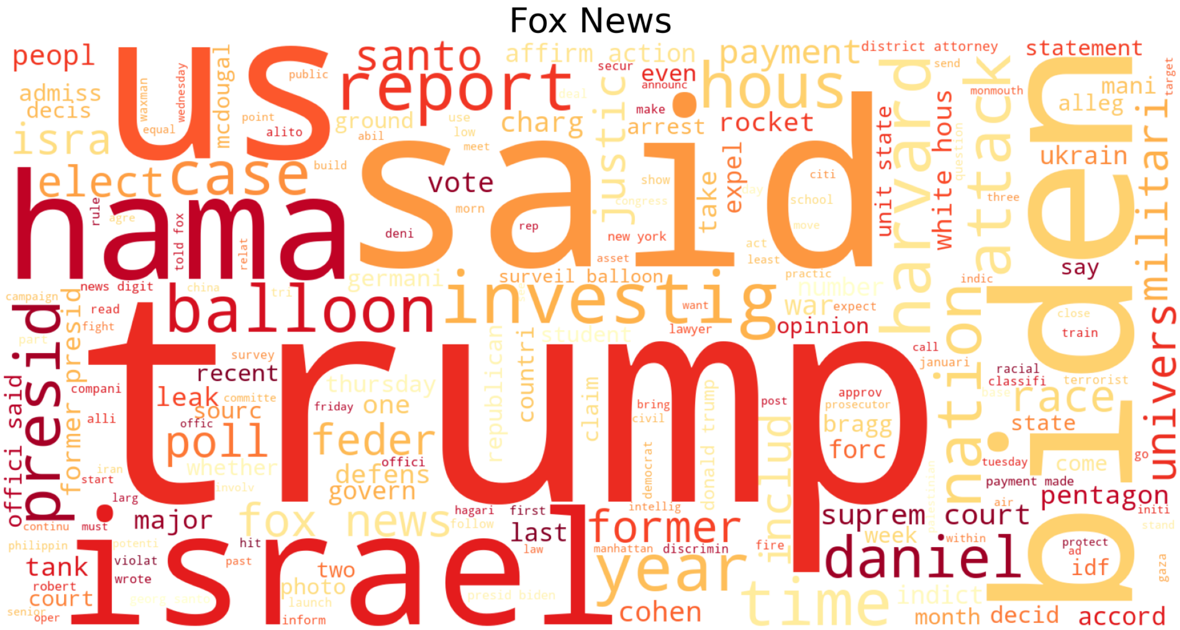


BBC


said
us
ukrain
gaza
tank
elect
time
one
decis
includ
isra
israel
document
report
call
offici
militari
democrat
attack
campaign
countri
year
mr trump
american
justic
republican
say
made
presid
court
mani
launch
hama
balloon
first
polit
charg
secur
inform
take
secretari
race
appear
show
law
attempt
rocket
back
group
last
thursday
russia
milit
kill
univers
former
china
defenc
legal
top
state
case
vote
polici
voter
use
ukrainian
ad base
detail
oper
peopl
dozen
rule
next
run
howev
expect
wrote
gener
major
outsid
expuls
harvard
sensit
indict

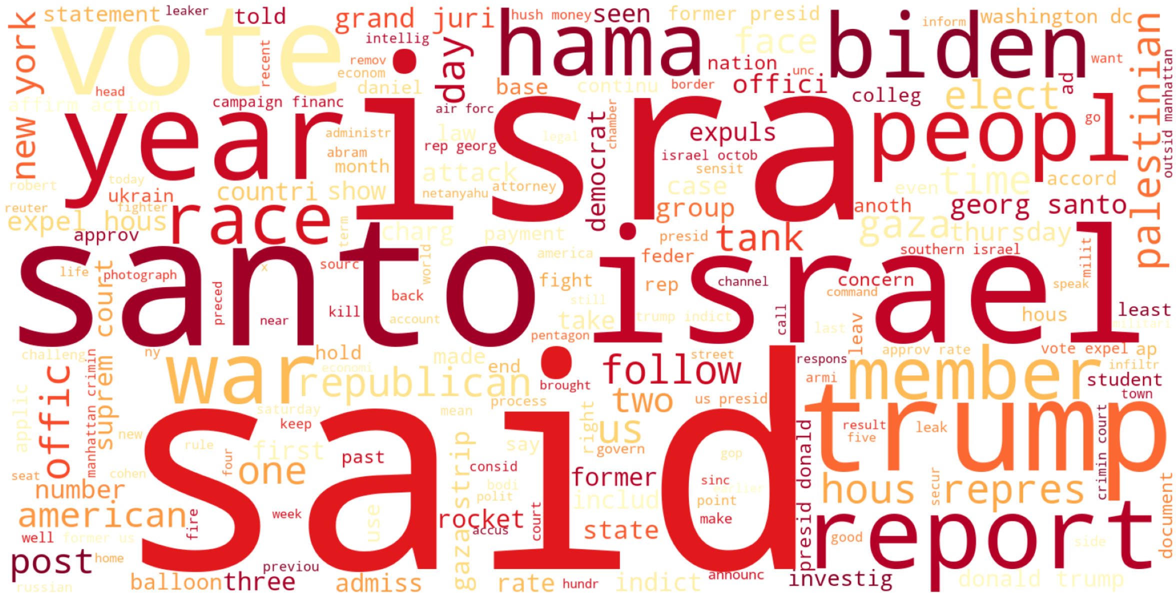
CNN


said
us
israel
document
includ
ukrain
militari
biden
trump
isra
hous
gaza
court
intellig
report
vote
offici
peopl
american
chines
major
countri
defens
say
teixeira
decis
inform
one
take
use
santo
month
effort
hama
two
justic
make
right
day
assess
leak
accord
senior
support
provid
come
call
balloon
person
republican
follow
fight
violat
school
first
govern
investig
alli
face
attack
war
appear
presid
effort
ukrainian
system

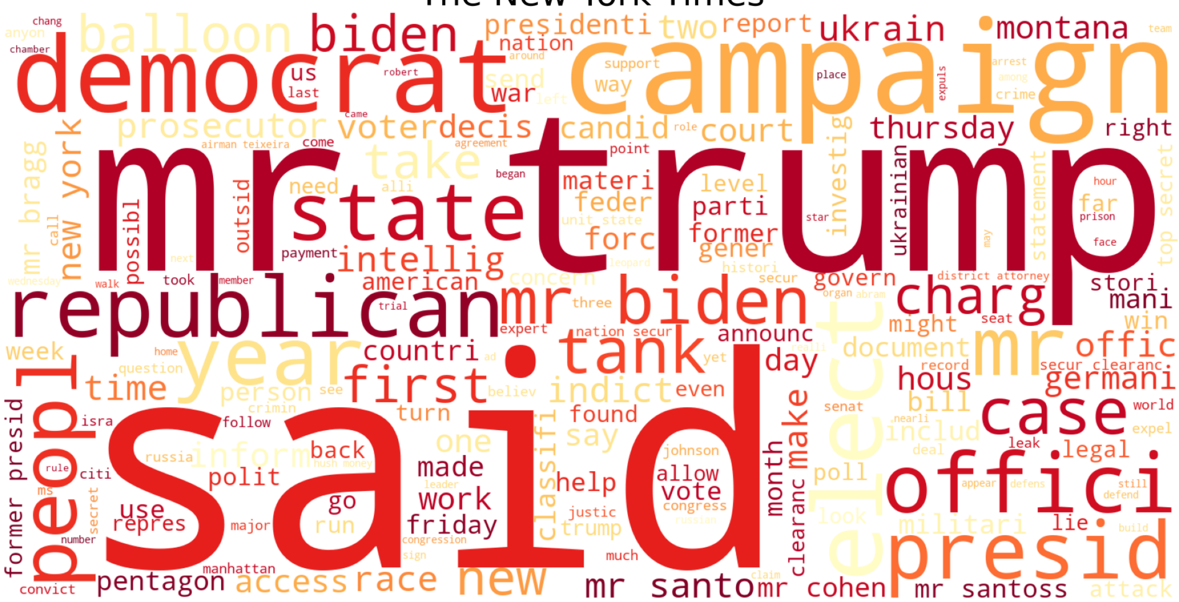

Fox News
trump
said
biden
israel
us
hama
report
santo
balloon
investig
harvard
attack
nation
militari
univers
hous
case
elect
daniel
former
year
time
presid
poll
feder
rocket
vote
payment
statement
suprem court
pentagon
race
state
ukrain
accord
decid
cohen
fox news
tank
major
leak
recent
opinion
defens
govern
month
idf
say


NBC News
said
trump
israel
document
rule
elect
vote
ukrain
balloon
court
santo
year
includ
democrat
offic
univers
biden
tank
one
first
indict
militari
race
month
thursday
nation
isra
hous
call
base
war
major
day
gaza
ad
two
provid
polit
intellig
peopl
govern
republican
russian
time

New York Post

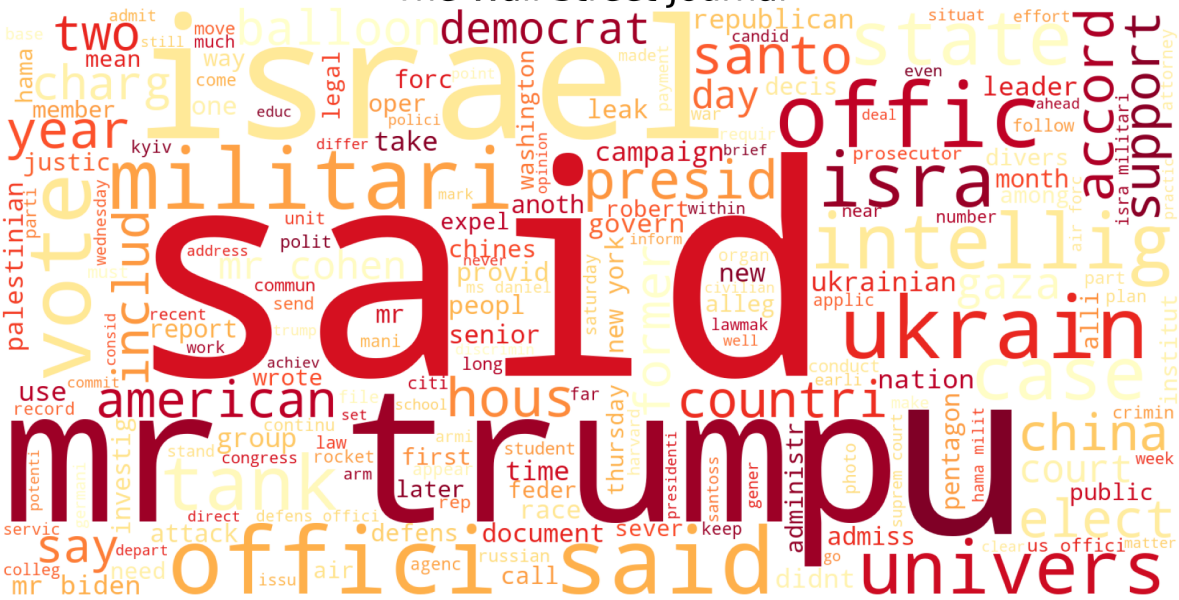


The New York Times



The Washington Post

The Wall Street Journal




```
# create a list of unstemmed article document texts
# looping through all text files to apply preprocessing functions
article_docs_unstemmed = []
dir = os.listdir('data/text/')
```

```
dir.sort()
for filename in dir:
    filepath = os.path.join('data/text/', filename)
    if filename.split(".")[-1] == "txt":
        article_string = file_to_string(filepath)
        new_string = clean_text(article_string, 0)
        article_docs_unstemmed.append(new_string)

article_docs_unstemmed
```

```
['supreme court thursday set new limits affirmative action programs cases involving whether p
 'massive spy balloon believed china seen montana tracked flies across continental united sta
 'two days dismal new polling numbers president joe biden showed public views unfavorably riv
 'hundreds people israel reported dead thousands injured hamas militants fired rockets gaza l
 'posting social media appears several highly classified us intelligence documents might begi
 'house representatives friday voted expel republican rep george santos historic move happene
 'major increase us support ukraine president joe biden signed sending abrams tanks war torn
 'manhattan grand jury indicted former president donald trump making first current former pre
 'us supreme court ruled race longer considered factor university admissions landmark ruling
 'us tracking suspected chinese surveillance balloon spotted flying sensitive sites recent da
 'us president joe biden running election next year national opinion polls weak job approval
 'least people reported killed wounded israel palestinian militant group hamas launched bigge
 'documents include detailed accounts training provided ukraine foreign powers make dozens cl
 'us house representatives expelled congressman george santos following damning ethics report
 'us send powerful battle tanks ukraine joining germany sending vehicles support fight russia
 'former us president donald trump charged hush money payments made porn star presidential el
 'supreme court says colleges universities longer take race consideration specific basis gran
 'us tracking suspected chinese high altitude surveillance balloon continental united states
 'one third registered voters approve president joe bidens handling israeli palestinian confl
 'gaza jerusalem cnn israels prime minister benjamin netanyahu declared country war saturday
 'man arrested fbi connection massive us classified documents leak charged boston friday unau
 'house voted friday expel gop rep george santos historic vote makes new york congressman six
 'leaders united states germany announced wednesday send contingents tanks ukraine reversing
 'donald trump faces counts related business fraud indictment manhattan grand jury according
 'us supreme court handed major ruling affirmative action thursday rejecting use race factor
 'us government monitoring suspected chinese surveillance balloon moving northern states past
 'biden battles rough poll numbers fox news white house correspondent peter doocy latest pres
 'hamas widespread coordinated attack may suggest outside help trey yingst foreign correspond
 'bret baier intel suspect year old government worker special report bret baier anchor questi
 'house representatives voted expel scandal plagued rep george santos r n friday making first
 'german chancellor olaf scholz formally announced wednesday weeks stalling frustrating negot
 'former president donald trump indicted part manhattan district attorney office years long i
```

'washington supreme court thursday struck affirmative action programs university north carol
'us military monitoring suspected chinese surveillance balloon hovering northern us past day
'different polls agree president joe bidens political standing lower barack obamas point ele
'ashkelon israel israel plunged chaos saturday palestinian militant group hamas launched dea
'dozens leaked defense department classified documents posted online reveal details us spyin
'washington house voted overwhelmingly expel indicted rep george santos friday pulling curta
'ukraine set receive battle tanks germany western countries fierce debate exposed fissures a
'grand jury new york city voted thursday indict donald trump first time former us president
'supreme court struck affirmative action programs harvard university university north caroli
'us tracking chinese spy balloon floating northern part country days pentagon officials anno
'president biden ready ring new year seeing numbers year old commander chief ends lower appr
'jerusalem ap hamas militants fired thousands rockets sent dozens fighters israeli towns nea
'massachusetts air national guardsman jack teixeira leader discord group dozens sensitive us
'bye george lying long island rep george santos r ny became sixth member ever expelled us ho
'weeks excuses foot dragging germans finally said yes transferring limited number leopard ta
'donald trump indicted manhattan grand jury thursday hush money payments made ahead election
'chief justice john g roberts jr finished reading majority opinion supreme court chamber thu
'helena mont larry mayer newspaper photographer pointed camera sky wednesday began snapping
'democrats battleground states growing increasingly anxious president bidens low approval ra
'israels news sites compiling lists dead missing funerals taking place around country weeken
'washington would year old national guardsman position access top secret documents begin dra
'george santos new york republican congressman whose tapestry lies schemes made figure natio
'precision military drill first germany united states announced wednesday agreed provide bat
'manhattan grand jury indicted donald j trump thursday role paying hush money porn star acco
'supreme court thursday held race conscious admissions programs harvard university north car
'chinese surveillance balloon collecting intelligence continental united states right us off
'night president biden departed washington celebrate thanksgiving nantucket mass gathered cl
'sderot israel israel formally declared war palestinian militant group hamas sunday reeled s
'saturday us officials foreign allies scrambled understand dozens classified intelligence do
'house voted friday expel rep george santos r n congress action chamber previously taken fiv
'biden administration announced wednesday send premier battle tanks ukraine following agreem
'new york manhattan grand jury voted indict former president donald trump making first perso
'thursdays decision force reworking admissions criteria throughout american higher educatio
'washington us tracked officials described chinese reconnaissance balloon continental states
'mr bidens low standing head head general election polls reflects voters dour appraisal peri
'tel aviv israeli prime minister benjamin netanyahu said country war hamas militant groups i
'criminal case unfolding us government scrambles protect secrets unauthorized disclosures ap
'lawmakers voted remove two thirds house supermajority required constitution almost democrat
'us germany outlined plans wednesday send dozens modern battle tanks ukraine marking signifi
'grand jury returned indictment mr trump vote thursday kicking process former president expe

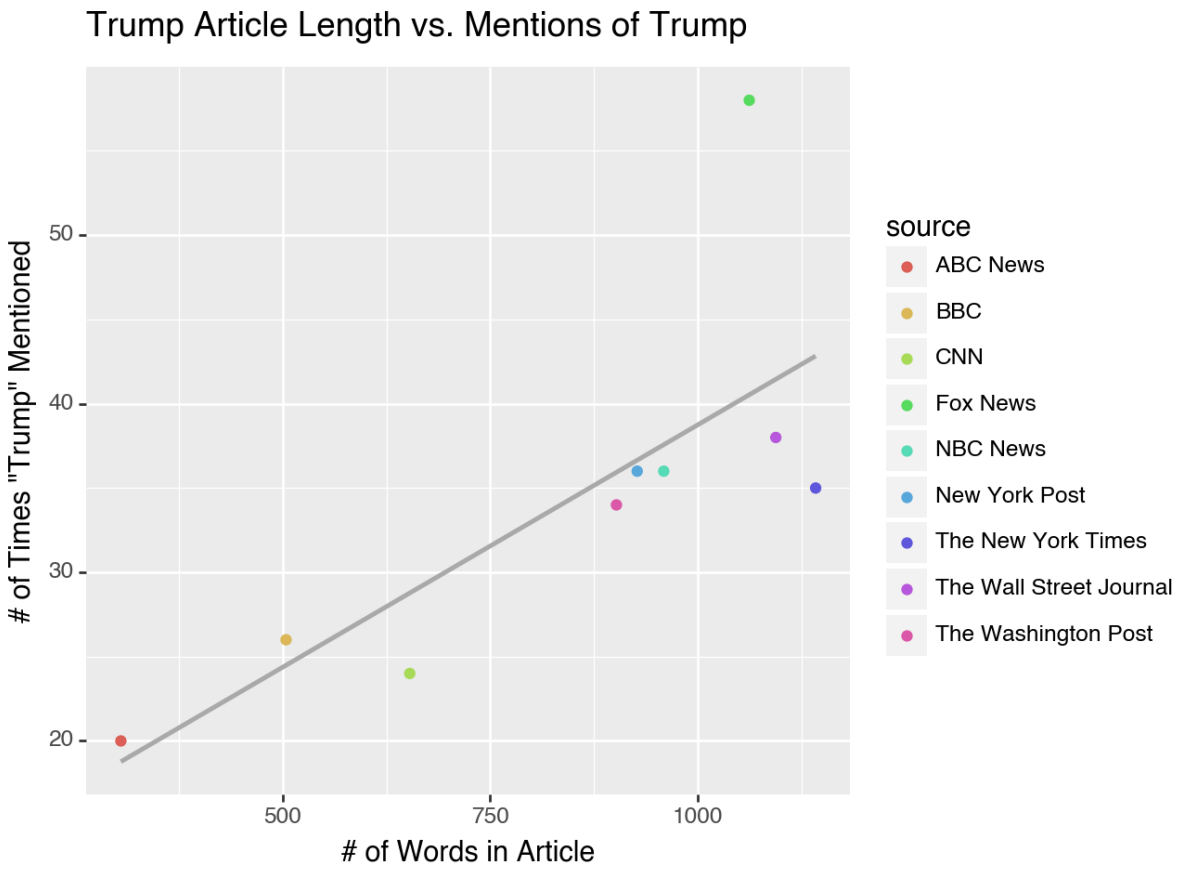
```
# investigating why Fox News has Trump as most frequent word, to see if it's related to le

trump_article_lengths = []
trump_word_counts = []

print("Source", "|", "Trump Article Length", "|", "Trump Word Count")
i = 1
for article in article_docs_unstemmed:
    if i%8 == 0:
        trump_count = 0
        for word in article.split():
            if word == "trump":
                trump_count += 1
        trump_word_counts.append(trump_count)
        article_length = len(article.split())
        trump_article_lengths.append(article_length)
        print(article.split()[-2], "|", article_length, "|", trump_count)
    i += 1

# let's do the same for the word "biden" in the biden articles and see if there are outlie

biden_article_lengths = []
biden_word_counts = []

print("Source", "|", "Biden Article Length", "|", "Biden Word Count")
i = 1
for article in article_docs_unstemmed:
    if article.split()[-1] == "bidenarticle":
        biden_count = 0
        for word in article.split():
            if word == "biden":
                biden_count += 1
        biden_word_counts.append(biden_count)
        article_length = len(article.split())
        biden_article_lengths.append(article_length)
        print(article.split()[-2], "|", article_length, "|", biden_count)
    i += 1
```

```
Source | Trump Article Length | Trump Word Count
abcarticle | 305 | 20
bbcarticle | 504 | 26
```

```
cnnarticle | 653 | 24
foxarticle | 1062 | 58
nbcarticle | 959 | 36
nyparticle | 927 | 36
nytarticle | 1142 | 35
wparticle | 902 | 34
wsjarticle | 1094 | 38
Source | Biden Article Length | Biden Word Count
abcarticle | 834 | 27
bbcarticle | 396 | 6
cnnarticle | 521 | 11
foxarticle | 406 | 19
nbcarticle | 565 | 11
nyparticle | 545 | 12
nytarticle | 920 | 30
wparticle | 911 | 32
wsjarticle | 404 | 10
```

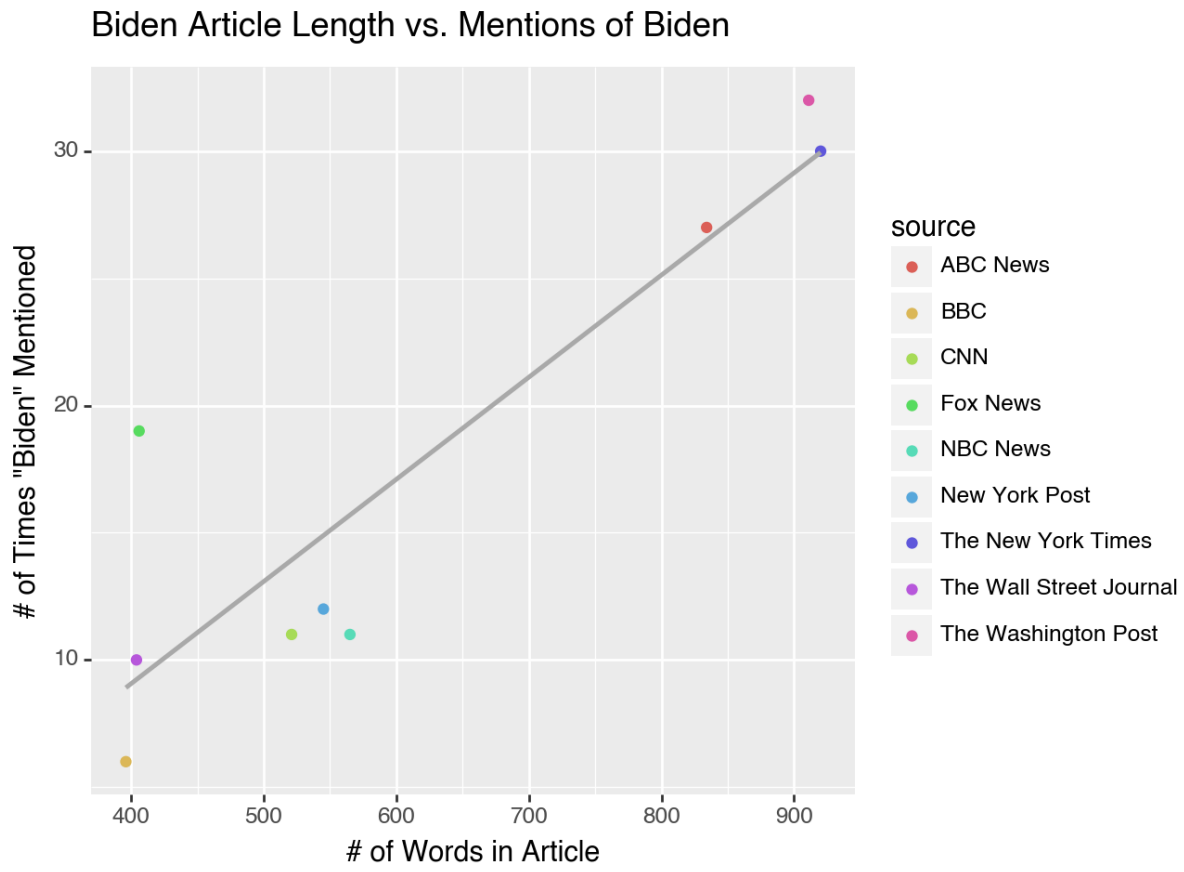
```
# create a df with sources, their trump article lengths, and the frequency of the word "tr

trump_words = pd.DataFrame(columns=['source', 'article_length', 'trump_word_count'])
trump_words['source'] = tfidf_source_df.index
trump_words['article_length'] = trump_article_lengths
trump_words['trump_word_count'] = trump_word_counts
trump_words

biden_words = pd.DataFrame(columns=['source', 'article_length', 'biden_word_count'])
biden_words['source'] = tfidf_source_df.index
biden_words['article_length'] = biden_article_lengths
biden_words['biden_word_count'] = biden_word_counts
```

```
(
ggplot(data=trump_words,
       mapping=aes(x='article_length', y='trump_word_count', color='source'))
       + geom_point(show_legend=True)
       + geom_smooth(method = "lm", se=False, color="darkgrey")
       + labs(title="Trump Article Length vs. Mentions of Trump",
              x='# of Words in Article',
              y='# of Times "Trump" Mentioned')
)
```



<Figure Size: (640 x 480)>

```
(
ggplot(data=biden_words,
       mapping=aes(x='article_length', y='biden_word_count', color='source'))
       + geom_point(show_legend=True)
       + geom_smooth(method = "lm", se=False, color="darkgrey")
       + labs(title="Biden Article Length vs. Mentions of Biden",
              x='# of Words in Article',
              y='# of Times "Biden" Mentioned')
)
```



<Figure Size: (640 x 480)>

```
# not using this for anything as of 3/14, don't remember why i created it
# create a list of strings where each string is all articles from one source (unstemmed)
source_docs_unstemmed = []

j = 0

for i in range(9):
    source = " ".join(article_docs_unstemmed[j].split()[:-2]) + " " + " ".join(article_doc
        + " ".join(article_docs_unstemmed[j+2].split()[:-2]) + " " + " ".join(article_docs
        + " ".join(article_docs_unstemmed[j+4].split()[:-2]) + " " + " ".join(article_docs
        + " ".join(article_docs_unstemmed[j+6].split()[:-2]) + " " + " ".join(article_docs
    source_docs_unstemmed.append(source)
    j += 8
```

```
source_docs_unstemmed
```

```
['supreme court thursday set new limits affirmative action programs cases involving whether p
 'us supreme court ruled race longer considered factor university admissions landmark ruling
 'supreme court says colleges universities longer take race consideration specific basis gra
 'us supreme court handed major ruling affirmative action thursday rejecting use race factor
 'washington supreme court thursday struck affirmative action programs university north caro
 'supreme court struck affirmative action programs harvard university university north caroli
 'chief justice john g roberts jr finished reading majority opinion supreme court chamber th
 'supreme court thursday held race conscious admissions programs harvard university north ca
 'thursdays decision force reworking admissions criteria throughout american higher educatio
```

```
first = article_docs_unstemmed[0]
" ".join(first.split()[:-2])
```

```
'supreme court thursday set new limits affirmative action programs cases involving whether pu
```

```
# calculate polarity and subjectivity scores, create dataframe

scores_df = pd.DataFrame({'source':tfidf_df['article_source'], 'topic':tfidf_df['article_t

scores_df['source'] = np.where(scores_df['source'] == "The New York Times", "New York Time
scores_df['source'] = np.where(scores_df['source'] == "The Washington Post", "Washington P
scores_df['source'] = np.where(scores_df['source'] == "The Wall Street Journal", "Wall Str

polarity_scores = []
subjectivity_scores = []

for article in article_docs_unstemmed:
    polarity_scores.append(round(TextBlob(" ".join(article.split()[:-2])).sentiment.polari
    subjectivity_scores.append(round(TextBlob(" ".join(article.split()[:-2])).sentiment.su

scores_df['polarity_score'] = polarity_scores
scores_df['subjectivity_score'] = subjectivity_scores

average_topic_polarity_scores = []
average_source_polarity_scores = []

for topic in scores_df['topic'].value_counts().index:
```

```
    mean_score = round(scores_df[scores_df['topic'] == topic]['polarity_score'].mean(), 2)
    average_topic_polarity_scores.append(mean_score)

for source in scores_df['source'].value_counts().index:
    mean_score = round(scores_df[scores_df['source'] == source]['polarity_score'].mean(),
    for i in range(8):
        average_source_polarity_scores.append(mean_score)

scores_df['average_polarity_for_topic'] = average_topic_polarity_scores * 9
scores_df['average_polarity_for_source'] = average_source_polarity_scores

average_topic_subjectivity_scores = []
average_source_subjectivity_scores = []

for topic in scores_df['topic'].value_counts().index:
    mean_score = round(scores_df[scores_df['topic'] == topic]['subjectivity_score'].mean()
    average_topic_subjectivity_scores.append(mean_score)

for source in scores_df['source'].value_counts().index:
    mean_score = round(scores_df[scores_df['source'] == source]['subjectivity_score'].mean
    for i in range(8):
        average_source_subjectivity_scores.append(mean_score)

scores_df['average_subjectivity_for_topic'] = average_topic_subjectivity_scores * 9
scores_df['average_subjectivity_for_source'] = average_source_subjectivity_scores

scores_df['polarity_diff_from_topic_mean'] = scores_df['polarity_score'] - scores_df['aver
scores_df['subjectivity_diff_from_topic_mean'] = scores_df['subjectivity_score'] - scores_
scores_df['polarity_diff_from_source_mean'] = scores_df['polarity_score'] - scores_df['ave
scores_df['subjectivity_diff_from_source_mean'] = scores_df['subjectivity_score'] - scores

sources_polarity_dev = []

for source in scores_df['source'].value_counts().index:
    total_dev = scores_df[scores_df['source'] == source]['polarity_diff_from_topic_mean'].
    for i in range(8):
        sources_polarity_dev.append(total_dev/8)

sources_subjectivity_dev = []

for source in scores_df['source'].value_counts().index:
```

```
    total_dev = scores_df[scores_df['source'] == source]['subjectivity_diff_from_topic_mea
    for i in range(8):
        sources_subjectivity_dev.append(total_dev/8)

scores_df['average_source_polarity_deviation_from_topic_mean'] = sources_polarity_dev
scores_df['average_source_subjectivity_deviation_from_topic_mean'] = sources_subjectivity_

scores_df['polarity_sign'] = np.where(scores_df['polarity_score'] > 0, 'pos', 'neg')

scores_df.tail(10)
```

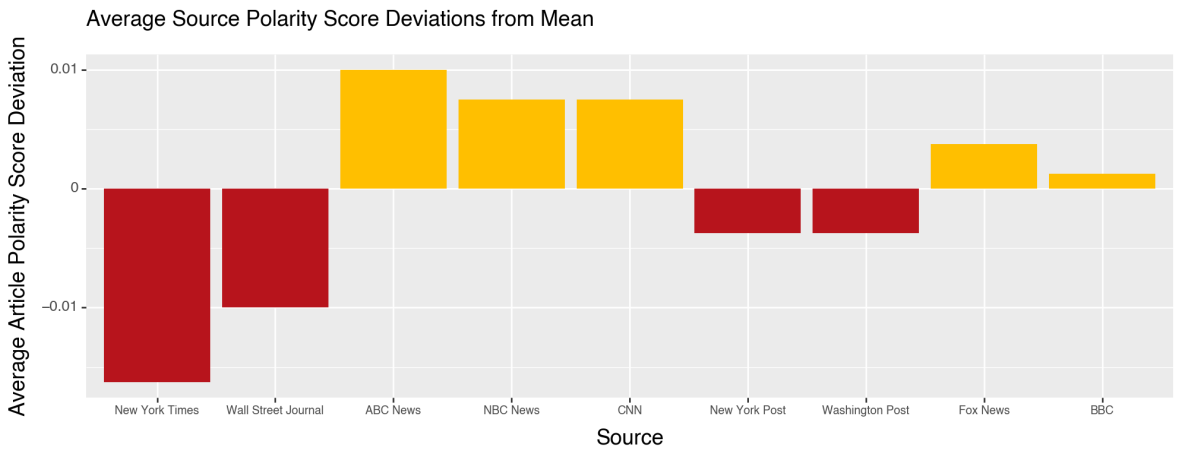
	source	topic	polarity_score	subjectivity_score
62	Washington Post	U.S. and Germany Send Tanks to Ukraine	0.04	0.38
63	Washington Post	Trump's Indictment	0.02	0.44
64	Wall Street Journal	Supreme Court Ruling on Affirmative Action	0.07	0.39
65	Wall Street Journal	Chinese Surveillance Balloon	0.02	0.28
66	Wall Street Journal	Biden's Low Approval Rates in Polls	0.07	0.42
67	Wall Street Journal	The Deadliest Attack by Hamas	0.04	0.32
68	Wall Street Journal	Pentagon Documents Leak	0.05	0.39
69	Wall Street Journal	George Santos' Expulsion from Congress	-0.04	0.35
70	Wall Street Journal	U.S. and Germany Send Tanks to Ukraine	0.11	0.32
71	Wall Street Journal	Trump's Indictment	0.03	0.42

```
# create a df for each source and its polarity score total deviations from the topic means

polarity_devs = pd.DataFrame({'source':scores_df['source'].value_counts().index,
                              'polarity_dev':scores_df[scores_df['topic'] == 'Chinese Surv
                                ['average_source_polarity_deviation_from_topic_mean']})
polarity_devs['sign'] = np.where(polarity_devs['polarity_dev'] > 0, 'pos', 'neg')

polarity_devs = polarity_devs.reindex(polarity_devs['polarity_dev'].abs().sort_values(asce
polarity_devs.reset_index(drop=True, inplace=True)

source_order = CategoricalDtype(
    ["New York Times", "Wall Street Journal", "ABC News", "NBC News",
     "CNN", "New York Post", "Washington Post", "Fox News", "BBC"],
    ordered=False
)
polarity_devs['source'] = polarity_devs['source'].astype(source_order)
```

```
polarity_devs
```

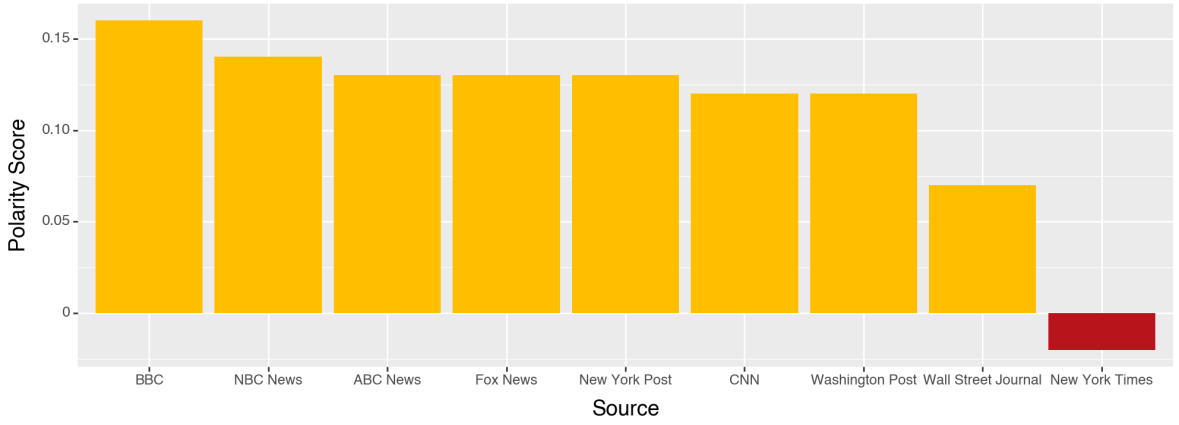
	source	polarity_dev	sign
0	New York Times	-0.01625	neg
1	ABC News	0.01000	pos
2	Wall Street Journal	-0.01000	neg
3	NBC News	0.00750	pos
4	CNN	0.00750	pos
5	New York Post	-0.00375	neg
6	Washington Post	-0.00375	neg
7	Fox News	0.00375	pos
8	BBC	0.00125	pos

```
# create bar chart representing how much, on average, a source's article
# deviates away from that topic's mean polarity score
# intended to show if a source tends to be more negative/positive than average,
# even accounting for the nature of the topic

# salmon and green: "pos":'#66bd63', "neg":'#f46d43'
# yellow: f9d62e
# dark red/pink: a31743
colors = {"pos":'#ffbf00', "neg":'#b7141c'}

(
ggplot(polarity_devs, aes(x="source", y="polarity_dev", fill="sign"))
+ geom_col(stat="identity")
+ labs(x="Source", y='Average Article Polarity Score Deviation', title='Average Source Pol
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=7, face='bold'),
        title = element_text(size=13),legend_position = "none")
+ scale_fill_manual(values = colors)
)
```



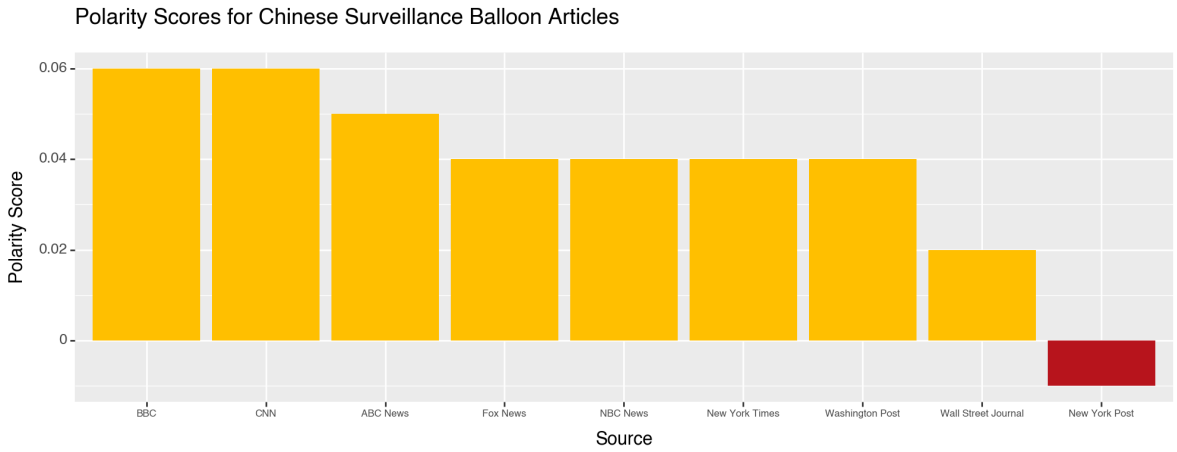
<Figure Size: (1000 x 400)>

```
(
ggplot(scores_df[scores_df['topic']=="Supreme Court Ruling on Affirmative Action"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title='')
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=8.5, face='bold'),
        title = element_text(size=13), legend_position = "none")
+ scale_fill_manual(values = colors)
)
```



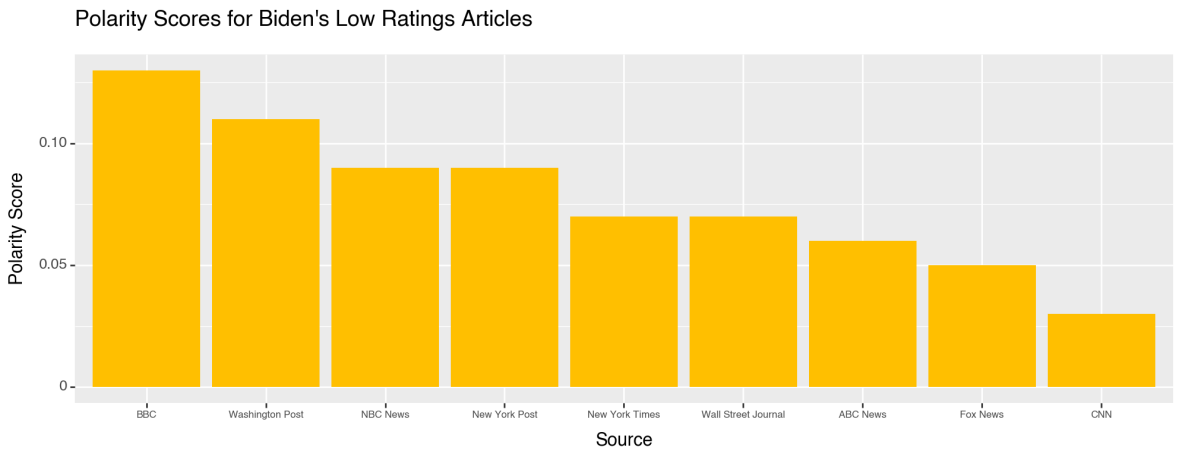
<Figure Size: (1000 x 400)>

```
(
ggplot(scores_df[scores_df['topic']=="Chinese Surveillance Balloon"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title='Polarity Scores for Chinese Surveillance Bal
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```



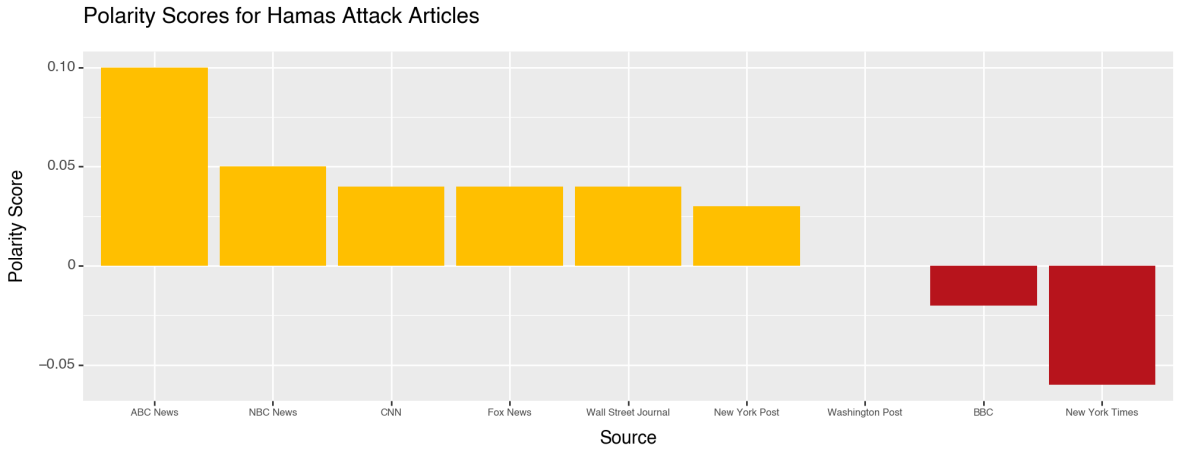

```
<Figure Size: (1000 x 400)>
```

```
(
ggplot(scores_df[scores_df['topic']=="Biden's Low Approval Rates in Polls"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title="Polarity Scores for Biden's Low Ratings Arti
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```



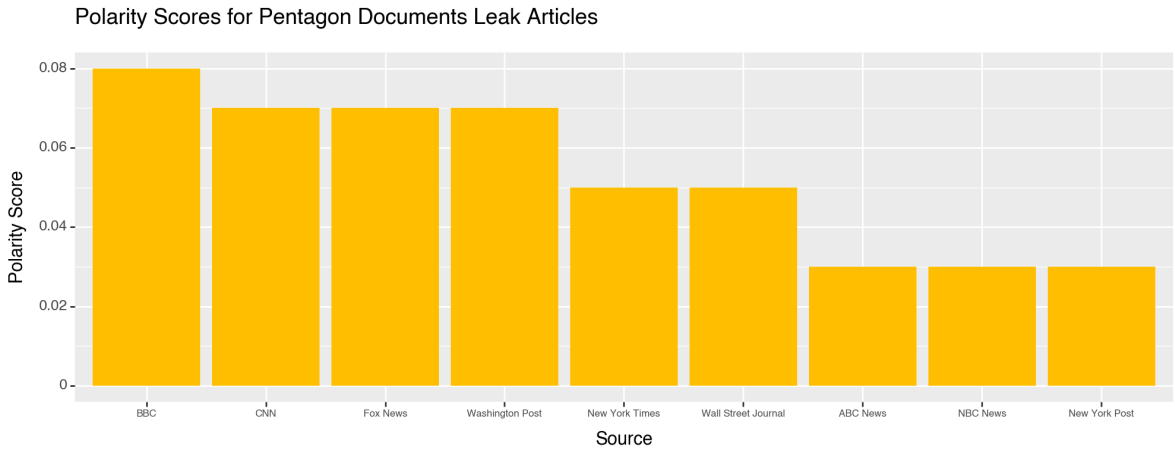
<Figure Size: (1000 x 400)>

```
(
ggplot(scores_df[scores_df['topic']=="The Deadliest Attack by Hamas"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title="Polarity Scores for Hamas Attack Articles")
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```



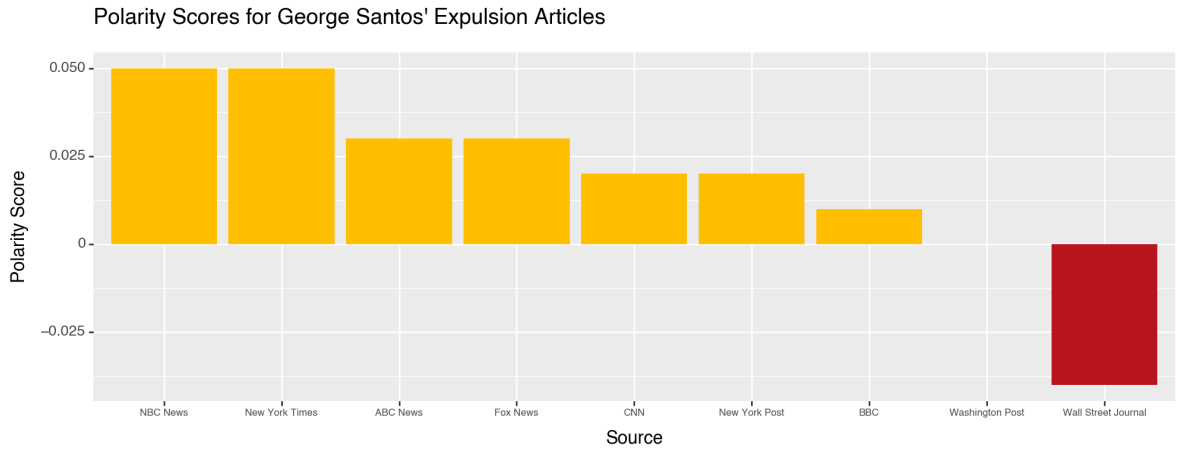
<Figure Size: (1000 x 400)>

```
(
ggplot(scores_df[scores_df['topic']=="Pentagon Documents Leak"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title="Polarity Scores for Pentagon Documents Leak
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```



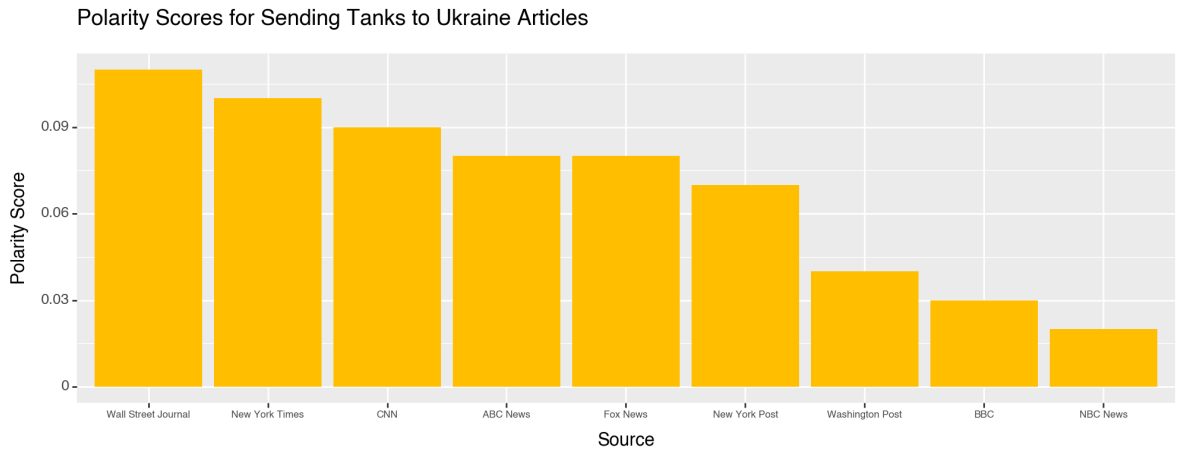
<Figure Size: (1000 x 400)>

```
(
ggplot(scores_df[scores_df['topic']=="George Santos' Expulsion from Congress"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title="Polarity Scores for George Santos' Expulsion
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```



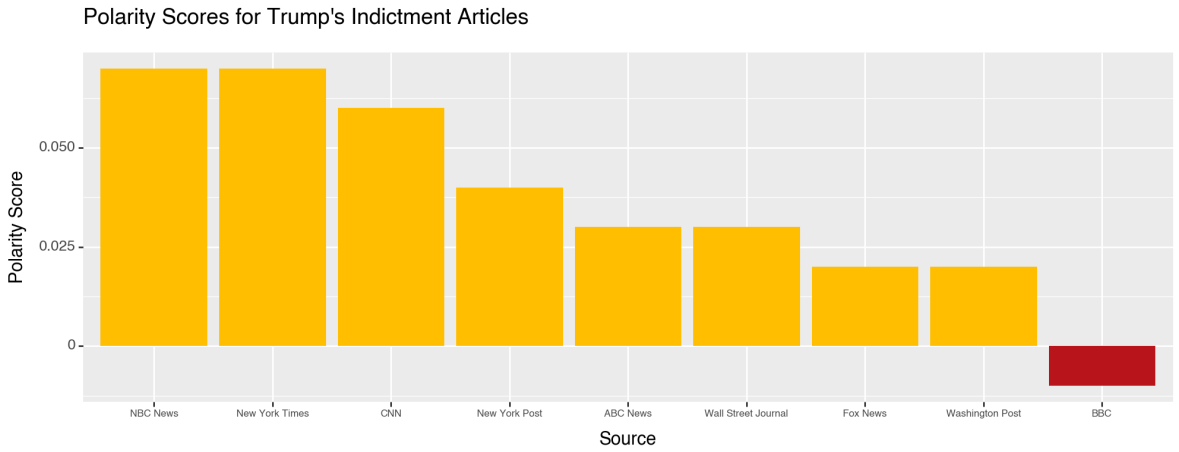
<Figure Size: (1000 x 400)>

```
(
ggplot(scores_df[scores_df['topic']=="U.S. and Germany Send Tanks to Ukraine"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title="Polarity Scores for Sending Tanks to Ukraine
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```



<Figure Size: (1000 x 400)>

```
(
ggplot(scores_df[scores_df['topic']=="Trump's Indictment"],
       aes(x="reorder(source, -polarity_score)", y="polarity_score", fill="polarity_sign")
+ geom_col(stat="identity")
+ labs(x='Source', y='Polarity Score', title="Polarity Scores for Trump's Indictment Artic
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```




```
<Figure Size: (1000 x 400)>
```

Do the bar charts above mean for example that ABC News was portraying the attack by hamas in a positive light? no, this is just supposed to give insight into the source's word choices.

```
scores_df[scores_df['topic']=='Supreme Court Ruling on Affirmative Action']
```

	source	topic	polarity_score	subjectivity_score
0	ABC News	Supreme Court Ruling on Affirmative Action	0.13	0.42
8	BBC	Supreme Court Ruling on Affirmative Action	0.16	0.37
16	CNN	Supreme Court Ruling on Affirmative Action	0.12	0.38
24	Fox News	Supreme Court Ruling on Affirmative Action	0.13	0.37
32	NBC News	Supreme Court Ruling on Affirmative Action	0.14	0.43
40	New York Post	Supreme Court Ruling on Affirmative Action	0.13	0.40
48	New York Times	Supreme Court Ruling on Affirmative Action	-0.02	0.37
56	Washington Post	Supreme Court Ruling on Affirmative Action	0.12	0.44

	source	topic	polarity_score	subjectivity_score
64	Wall Street Journal	Supreme Court Ruling on Affirmative Action	0.07	0.39

```
# check if normally distributed
```

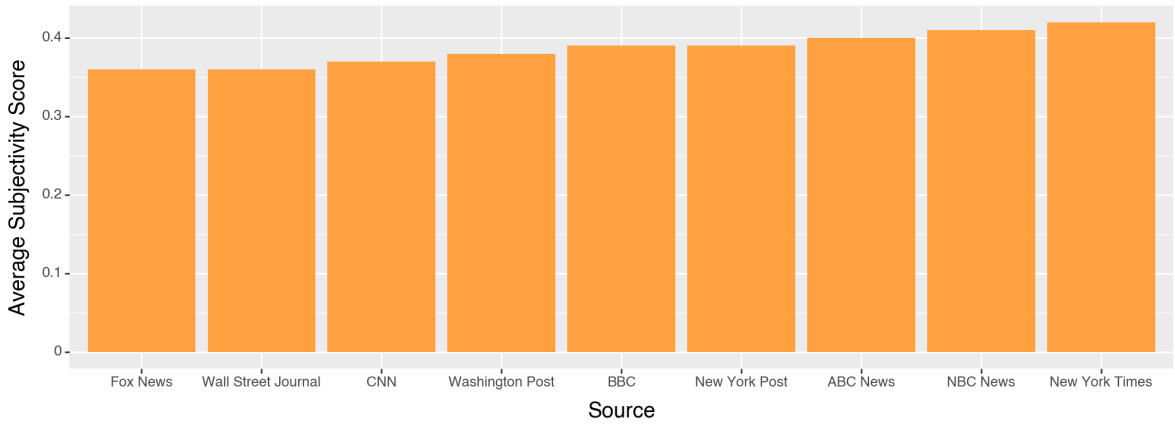
```
# hypothesis test for difference in mean polarity score deviations

import statsmodels.api as sm
from statsmodels.formula.api import ols

# formula = 'mood_gain ~ drug'

# model = ols(formula, data=df).fit()

# aov_table = sm.stats.anova_lm(model, typ=2)
# aov_table
```

```
source_avg_subj = pd.DataFrame({'source':scores_df['source'].value_counts().index,
                                'avg_subj':scores_df[scores_df['topic'] == 'Chinese Surveill
                                  ['average_subjectivity_for_source']})
source_avg_subj.sort_values(by='avg_subj', ascending=True, inplace=True)
source_avg_subj.reset_index(drop=True, inplace=True)

source_order = CategoricalDtype(
    ["Fox News", "Wall Street Journal", "CNN",
     "Washington Post", "BBC", "New York Post", "ABC News", "NBC News", "New York Times"],
    ordered=False
)
source_avg_subj['source'] = source_avg_subj['source'].astype(source_order)

source_avg_subj
```

	source	avg_subj
0	Fox News	0.36
1	Wall Street Journal	0.36
2	CNN	0.37
3	Washington Post	0.38
4	BBC	0.39
5	New York Post	0.39

	source	avg_subj
6	ABC News	0.40
7	NBC News	0.41
8	New York Times	0.42

```
# old color: slateblue
(
ggplot(source_avg_subj, aes(x="source", y="avg_subj"))
+ geom_col(stat="identity", fill="#ffa140")
+ labs(x='Source', y='Average Subjectivity Score', title='')
+ theme(figure_size = (10, 4), axis_text_x=element_text(size=8.5, face='bold'),
        title = element_text(size=13), legend_position = "none")
)
```



<Figure Size: (1000 x 400)>

```
# create a list of tuples that contain articles that have an outlier polarity score,
# compared to other articles about the same topic

articles =[]

for topic in scores_df['topic'].value_counts().index:
    topic_df = scores_df[scores_df['topic'] == topic]
    Q1 = topic_df['polarity_score'].describe()[4]
    Q3 = topic_df['polarity_score'].describe()[6]
```

```
    IQR = Q3-Q1
    lower = Q1-(1.5*IQR)
    upper = Q3+(1.5*IQR)
    print(topic, "lower bound:", round(lower, 2), "upper bound:", round(upper,2))
    for row in topic_df.itertuples():
        if row.polarity_score < lower or row.polarity_score > upper:
            articles.append((row.source, row.topic, row.polarity_score))

print(articles)
print(len(articles))
```

```
Supreme Court Ruling on Affirmative Action lower bound: 0.1 upper bound: 0.15
Chinese Surveillance Balloon lower bound: 0.02 upper bound: 0.06
Biden's Low Approval Rates in Polls lower bound: 0.02 upper bound: 0.14
The Deadliest Attack by Hamas lower bound: -0.06 upper bound: 0.1
Pentagon Documents Leak lower bound: -0.03 upper bound: 0.13
George Santos' Expulsion from Congress lower bound: -0.02 upper bound: 0.06
U.S. and Germany Send Tanks to Ukraine lower bound: -0.03 upper bound: 0.16
Trump's Indictment lower bound: -0.04 upper bound: 0.12
[('BBC', 'Supreme Court Ruling on Affirmative Action', 0.16), ('New York Times', 'Supreme Cou
6
```

By the outlier detection above, there are 6 articles that are considered to have an outlier polarity score for their topic. Three of those are articles from the Wall Street Journal. Also, three of the outlier articles are about the Supreme Court Ruling on Affirmative Action.

```
for j in range(8,8):
    print(j)
```

```
8
```

```
# function to apply Naive Bayed model using different holdout article topics each time

topic_list = topic_choices  = [
      "Supreme Court Ruling on Affirmative Action"
    , "Chinese Surveillance Balloon"
    , "Biden's Low Approval Rates in Polls"
    , "The Deadliest Attack by Hamas"
    , "Pentagon Documents Leak"
    , "George Santos' Expulsion from Congress"
```

```
    , "U.S. and Germany Send Tanks to Ukraine"
    , "Trump's Indictment"
]

def naive_bayes_cf(df, value_type):
    '''Takes in a dataframe of tokens and the desired values
    to train the model on (i.e. tfidf scores, frequencies)
    and a string representing what the values are.
    Trains, tests, and outputs performance metrics of a series
    of Multinomial Naive Bayes classifiers, one for each
    possible holdout set of two article topics.'''

    accuracy_list = []
    f1_score_list = []

    for i in range(len(topic_list)):
        for j in range(i+1, len(topic_list)):
            print("Holdout article topics: ")
            print(topic_list[i])
            print(topic_list[j])
            X_train_df = df[(df['article_topic'] != topic_list[i]) &
                    (df['article_topic'] != topic_list[j])].iloc[:, :-2]
            X_test_df = df[(df['article_topic'] == topic_list[i]) |
                    (df['article_topic'] == topic_list[j])].iloc[:, :-2]
            y_train_df = df[(df['article_topic'] != topic_list[i]) &
                    (df['article_topic'] != topic_list[j])]['article_source']
            y_test_df = df[(df['article_topic'] == topic_list[i]) |
                    (df['article_topic'] == topic_list[j])]['article_source']
            X_train = X_train_df.to_numpy()
            X_test = X_test_df.to_numpy()
            y_train = y_train_df.to_numpy()
            y_test = y_test_df.to_numpy()

            nb_model = MultinomialNB()
            nb_model.fit(X_train, y_train)

            predictions_array = nb_model.predict(X_test)

            prediction_list = []

            for pred in predictions_array:
```

```
                prediction_list.append(pred)

            actuals_list = []

            for source in y_test:
                actuals_list.append(source)

            preds_actuals_df = pd.DataFrame({'Actual':actuals_list, 'Predicted':prediction

            #display(preds_actuals_df)

            accuracy = accuracy_score(predictions_array, y_test)
            f1 = f1_score(predictions_array, y_test, average="weighted")

            accuracy_list.append(accuracy)
            f1_score_list.append(f1)

            print("Accuracy of Multinomial Naive Bayes Model:", accuracy)
            print("F1 score of Multinomial Naive Bayes Model:", f1)
            print("\n\n")

    print(f"Minimum Accuracy Score of any Multinomial Naive Bayes Model using {value_type}
        min(accuracy_list))
    print(f"Maximum Accuracy Score of any Multinomial Naive Bayes Model using {value_type}
        max(accuracy_list))
    print(f"Maximum F1 Score of any Multinomial Naive Bayes Model using {value_type}:",
        max(f1_score_list))
```

```
naive_bayes_cf(tfidf_df, "TF-IDF Score")
```

```
Holdout article topics:
Supreme Court Ruling on Affirmative Action
Chinese Surveillance Balloon
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4222222222222223



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Biden's Low Approval Rates in Polls
```

```
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.4802469135802469



Holdout article topics:
Supreme Court Ruling on Affirmative Action
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.28518518518518515



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.39382716049382716



Holdout article topics:
Supreme Court Ruling on Affirmative Action
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4246913580246914



Holdout article topics:
Supreme Court Ruling on Affirmative Action
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.36243386243386244



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.46049382716049386
```

```
Holdout article topics:
Chinese Surveillance Balloon
Biden's Low Approval Rates in Polls
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.2073112073112073



Holdout article topics:
Chinese Surveillance Balloon
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.3287749287749288



Holdout article topics:
Chinese Surveillance Balloon
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4336700336700337



Holdout article topics:
Chinese Surveillance Balloon
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4222222222222223



Holdout article topics:
Chinese Surveillance Balloon
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.3148148148148148



Holdout article topics:
```

```
Chinese Surveillance Balloon
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4049382716049383



Holdout article topics:
Biden's Low Approval Rates in Polls
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.4666666666666667



Holdout article topics:
Biden's Low Approval Rates in Polls
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.46666666666666656



Holdout article topics:
Biden's Low Approval Rates in Polls
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.37037037037037035



Holdout article topics:
Biden's Low Approval Rates in Polls
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3714285714285715



Holdout article topics:
Biden's Low Approval Rates in Polls
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
```

```
F1 score of Multinomial Naive Bayes Model: 0.45185185185185184



Holdout article topics:
The Deadliest Attack by Hamas
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.38148148148148153



Holdout article topics:
The Deadliest Attack by Hamas
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.29074074074074074



Holdout article topics:
The Deadliest Attack by Hamas
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.27380952380952384



Holdout article topics:
The Deadliest Attack by Hamas
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.45379188712522045



Holdout article topics:
Pentagon Documents Leak
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.41887125220458554
```

```
Holdout article topics:
Pentagon Documents Leak
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.38148148148148153



Holdout article topics:
Pentagon Documents Leak
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.5
F1 score of Multinomial Naive Bayes Model: 0.6031746031746033



Holdout article topics:
George Santos' Expulsion from Congress
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3074074074074074



Holdout article topics:
George Santos' Expulsion from Congress
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.44814814814814824



Holdout article topics:
U.S. and Germany Send Tanks to Ukraine
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.5
F1 score of Multinomial Naive Bayes Model: 0.5037037037037037



Minimum Accuracy Score of any Multinomial Naive Bayes Model using TF-IDF Score: 0.16666666660
Maximum Accuracy Score of any Multinomial Naive Bayes Model using TF-IDF Score: 0.5
```

```
Maximum F1 Score of any Multinomial Naive Bayes Model using TF-IDF Score: 0.6031746031746033
```

Since we have the same number of articles from each of 9 sources, if we chose a random source as our guess for where an article came from, we would expect to guess correctly about 1/9, or 11% of the time (baseline accuracy). Even the model with the lowest accuracy score outperformed the accuracy we would expect if we took a random guess.

```
naive_bayes_cf(freq_df, "Frequencies")
```

```
Holdout article topics:
Supreme Court Ruling on Affirmative Action
Chinese Surveillance Balloon
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.2143658810325477



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Biden's Low Approval Rates in Polls
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.3074074074074074



Holdout article topics:
Supreme Court Ruling on Affirmative Action
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.1111111111111111
F1 score of Multinomial Naive Bayes Model: 0.17732884399551066



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.21683501683501682



Holdout article topics:
```

```
Supreme Court Ruling on Affirmative Action
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.2796296296296296



Holdout article topics:
Supreme Court Ruling on Affirmative Action
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.24434824434824437



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.26286676286676286



Holdout article topics:
Chinese Surveillance Balloon
Biden's Low Approval Rates in Polls
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.2938271604938272



Holdout article topics:
Chinese Surveillance Balloon
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.23586744639376217



Holdout article topics:
Chinese Surveillance Balloon
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
```

```
F1 score of Multinomial Naive Bayes Model: 0.2851851851851852



Holdout article topics:
Chinese Surveillance Balloon
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4303350970017637



Holdout article topics:
Chinese Surveillance Balloon
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.3271604938271605



Holdout article topics:
Chinese Surveillance Balloon
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.22777777777777775



Holdout article topics:
Biden's Low Approval Rates in Polls
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.27037037037037037



Holdout article topics:
Biden's Low Approval Rates in Polls
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.42592592592592593
```

```
Holdout article topics:
Biden's Low Approval Rates in Polls
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.288888888888889



Holdout article topics:
Biden's Low Approval Rates in Polls
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3108465608465608



Holdout article topics:
Biden's Low Approval Rates in Polls
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.22407407407407406



Holdout article topics:
The Deadliest Attack by Hamas
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.2514029180695847



Holdout article topics:
The Deadliest Attack by Hamas
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.28720538720538724



Holdout article topics:
The Deadliest Attack by Hamas
```

```
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.1111111111111111
F1 score of Multinomial Naive Bayes Model: 0.17424242424242423



Holdout article topics:
The Deadliest Attack by Hamas
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.24090909090909088



Holdout article topics:
Pentagon Documents Leak
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.4225589225589225



Holdout article topics:
Pentagon Documents Leak
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3648148148148148



Holdout article topics:
Pentagon Documents Leak
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.2768959435626102



Holdout article topics:
George Santos' Expulsion from Congress
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.40740740740740744
```

```
Holdout article topics:
George Santos' Expulsion from Congress
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.35132275132275137



Holdout article topics:
U.S. and Germany Send Tanks to Ukraine
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.33306878306878307



Minimum Accuracy Score of any Multinomial Naive Bayes Model using Frequencies: 0.11111111111
Maximum Accuracy Score of any Multinomial Naive Bayes Model using Frequencies: 0.38888888888
Maximum F1 Score of any Multinomial Naive Bayes Model using Frequencies: 0.4303350970017637
```

```
naive_bayes_cf(binary_df, "Binary values")
```

```
Holdout article topics:
Supreme Court Ruling on Affirmative Action
Chinese Surveillance Balloon
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.3703703703703704



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Biden's Low Approval Rates in Polls
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3068783068783069
```

```
Holdout article topics:
Supreme Court Ruling on Affirmative Action
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.42037037037037034



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.2978835978835979



Holdout article topics:
Supreme Court Ruling on Affirmative Action
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.41358024691358025



Holdout article topics:
Supreme Court Ruling on Affirmative Action
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3740740740740741



Holdout article topics:
Supreme Court Ruling on Affirmative Action
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.4444444444444444
F1 score of Multinomial Naive Bayes Model: 0.4962962962962963



Holdout article topics:
Chinese Surveillance Balloon
Biden's Low Approval Rates in Polls
```

```
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.35185185185185186



Holdout article topics:
Chinese Surveillance Balloon
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.37851037851037855



Holdout article topics:
Chinese Surveillance Balloon
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.16666666666666666
F1 score of Multinomial Naive Bayes Model: 0.23456790123456792



Holdout article topics:
Chinese Surveillance Balloon
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.3888888888888889



Holdout article topics:
Chinese Surveillance Balloon
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.2716049382716049



Holdout article topics:
Chinese Surveillance Balloon
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3547008547008547
```

```
Holdout article topics:
Biden's Low Approval Rates in Polls
The Deadliest Attack by Hamas
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.32098765432098764



Holdout article topics:
Biden's Low Approval Rates in Polls
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.3037037037037037



Holdout article topics:
Biden's Low Approval Rates in Polls
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.4444444444444444
F1 score of Multinomial Naive Bayes Model: 0.5061728395061728



Holdout article topics:
Biden's Low Approval Rates in Polls
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3966329966329966



Holdout article topics:
Biden's Low Approval Rates in Polls
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.31481481481481477



Holdout article topics:
```

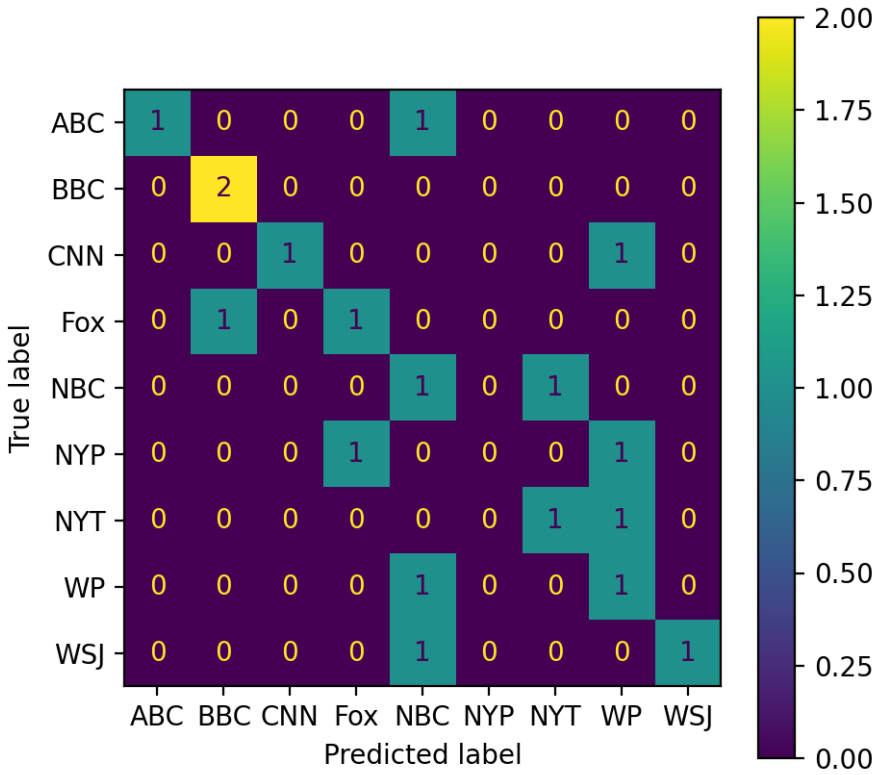
```
The Deadliest Attack by Hamas
Pentagon Documents Leak
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3386243386243386



Holdout article topics:
The Deadliest Attack by Hamas
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4534391534391534



Holdout article topics:
The Deadliest Attack by Hamas
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.273977873977874



Holdout article topics:
The Deadliest Attack by Hamas
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.5
F1 score of Multinomial Naive Bayes Model: 0.5216931216931218



Holdout article topics:
Pentagon Documents Leak
George Santos' Expulsion from Congress
Accuracy of Multinomial Naive Bayes Model: 0.3888888888888889
F1 score of Multinomial Naive Bayes Model: 0.4



Holdout article topics:
Pentagon Documents Leak
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
```

```
F1 score of Multinomial Naive Bayes Model: 0.3796296296296296



Holdout article topics:
Pentagon Documents Leak
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.3740740740740741



Holdout article topics:
George Santos' Expulsion from Congress
U.S. and Germany Send Tanks to Ukraine
Accuracy of Multinomial Naive Bayes Model: 0.2777777777777778
F1 score of Multinomial Naive Bayes Model: 0.35185185185185186



Holdout article topics:
George Santos' Expulsion from Congress
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.2222222222222222
F1 score of Multinomial Naive Bayes Model: 0.2839506172839506



Holdout article topics:
U.S. and Germany Send Tanks to Ukraine
Trump's Indictment
Accuracy of Multinomial Naive Bayes Model: 0.3333333333333333
F1 score of Multinomial Naive Bayes Model: 0.4296296296296297



Minimum Accuracy Score of any Multinomial Naive Bayes Model using Binary values: 0.166666666
Maximum Accuracy Score of any Multinomial Naive Bayes Model using Binary values: 0.5
Maximum F1 Score of any Multinomial Naive Bayes Model using Binary values: 0.521693121693121
```

Looking at the performance metrics of the various Naive Bayes models, the models trained on the dataframes with tf-idf scores and the binary values had similar performance. For each

of those sets of models, the maximum accuracy score of any of the models was 50% and the minimum approximately 17%. The dataframe with frequencies of tokens created the worst performing models, the highest one having an accuracy score of approximately 39%.

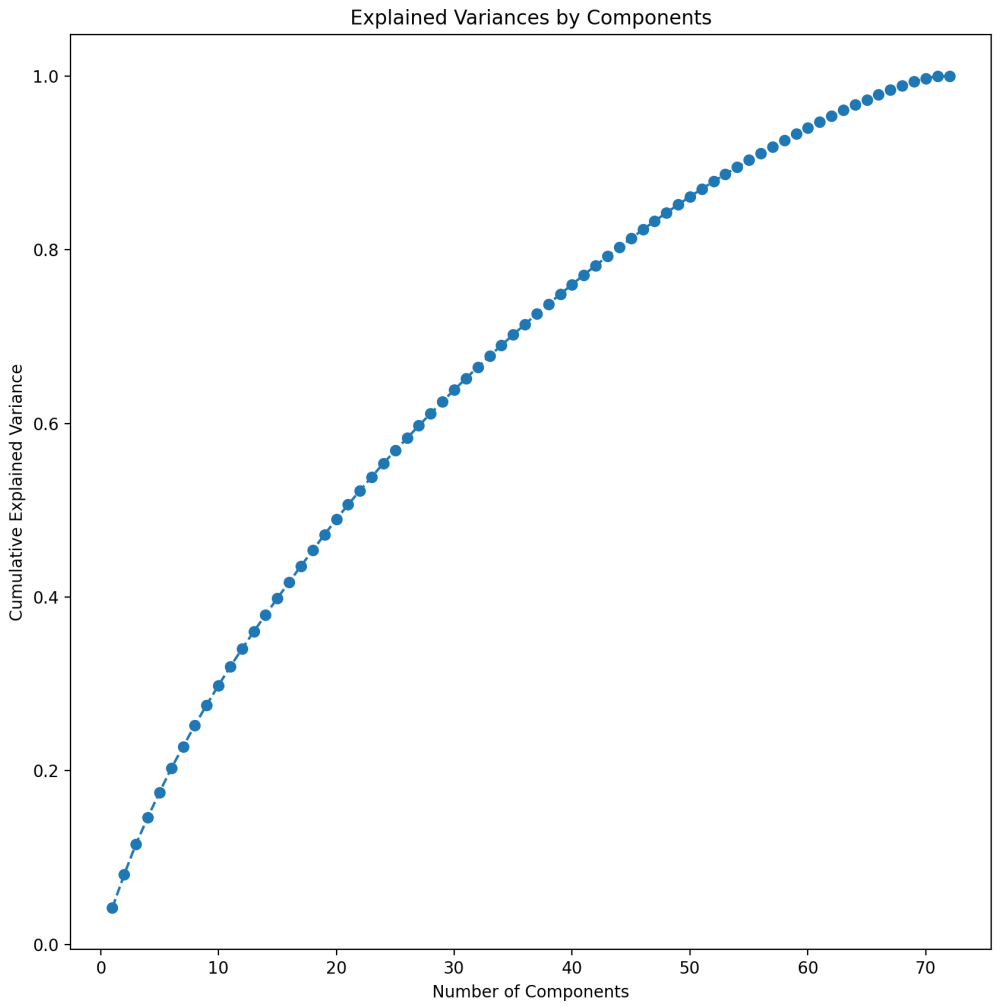
```
# confusion matrix code

labels = ["ABC News", "BBC", "CNN", "Fox News", "NBC News", "New York Post",
          "The New York Times", "The Washington Post", "The Wall Street Journal"]

labels_short = ["ABC", "BBC", "CNN", "Fox", "NBC", "NYP",
          "NYT", "WP", "WSJ"]

cm = confusion_matrix(y_test, predictions_array, labels=labels)
#fig, ax = plt.subplots(figsize=(10, 10))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels_short)
disp.plot()
fig = disp.ax_.get_figure()
fig.set_figwidth(5)
fig.set_figheight(5)
```




```
# standardize the frequency df before doing PCA
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
scaler = StandardScaler()

freq_standardized = pd.DataFrame(scaler.fit_transform(freq_df.iloc[:, :-2]), columns = fre
#freq_standardized = preprocessing.scale(freq_df.iloc[:, :-2], axis=0)  # standardize data

# # Instantiate PCA estimator
# pca = decomposition.PCA()
# # fit : Run PCA
# tokens_pc = pca.fit(freq_standardized)
# vars(tokens_pc)
```

```
# Defining the number of principal components to generate
n = freq_standardized.shape[1]

# Finding principal components for the data
pca = PCA(n_components = 72, random_state = 1) # Apply the PCA algorithm with random_state

data_pca1 = pd.DataFrame(pca.fit_transform(freq_standardized))   # Fit and transform the p

# The percentage of variance explained by each principal component
exp_var = pca.explained_variance_ratio_
```

```
# Visualize the explained variance by individual components
plt.figure(figsize = (10, 10))

plt.plot(range(1, 73), exp_var.cumsum(), marker = 'o', linestyle = '--')

plt.title("Explained Variances by Components")

plt.xlabel("Number of Components")

plt.ylabel("Cumulative Explained Variance")

plt.show()
```




```
# Finding the least number of components that can explain more than 90% variance
sum = 0

for ix, i in enumerate(exp_var):
    sum = sum + i
    if(sum>0.90):
        print("Number of PCs that explain at least 90% variance: ", ix + 1)
```

```
        break
```

```
Number of PCs that explain at least 90% variance:  55
```

```
PCs = pd.DataFrame(tokens_pc.components_.T[:10],
            columns=['PC1', 'PC2','PC3','PC4', 'PC5', 'PC6', 'PC7','PC8','PC9', 'PC10'])

PCs                 # PC loadings saved with Scikit-learn
```

```
ValueError: Shape of passed values is (10, 72), indices imply (10, 10)
```