```
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
```

```
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 ..
```

```
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
```

```
# initialize a dictionary to be able to find the original word from the stemmed word
stemmed_dict = {}

# how to later access the key by the value
#value = {i for i in dic if dic[i]=="n"}
#print("key by value:",value)
```

```
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()
    # create list of stopwords
    stopwords_list = stopwords.words('english')
    # make the text lowercase
    text = text.lower()
    text = text.replace('-', ' ')
    # 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 0x173059d30>
```

	said	trump	biden	offici	mr	u	israel	presid	tank	hous	...	messr	overlap	vs	convert
0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	1.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
2	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
3	1.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0
4	1.0	0.0	0.0	1.0	0.0	1.0	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]=="marku"}
print("key by value:",value)
```

```
key by value: {'markus'}
```

```
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 u intellig document might begin could turn s
 'hous repres friday vote expel republican rep georg santo histor move happen year santo scan
 'major increas u support ukrain presid joe biden sign send abram tank war torn countri conce
 '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 vo
```

'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 he
'us send power battl tank ukrain join germani send vehicl support fight russia invas decis c
'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 landmark
'us track suspect chines high altitud surveil balloon continent unit state defens offici sai
'one third regist voter approv presid joe biden handl isra palestinian conflict new poll new
'gaza jerusalem cnn israel prime minist benjamin netanyahu declar countri war saturday pales
'man arrest fbi connect massiv us classifi document leak charg boston friday unauthor retent
'hous vote friday expel gop rep georg santo histor vote make new york congressman sixth lawm
'leader unit state germani announc wednesday send conting tank ukrain revers longstand trepi
'donald trump face count relat busi fraud indict manhattan grand juri accord two sourc famil
'u suprem court hand major rule affirm action thursday reject use race factor colleg admiss
'u govern monitor suspect chines surveil balloon move northern state past sever day pentagon
'biden battl rough poll number fox news white hous correspond peter dooci latest presid elec
'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 i
'hous repres vote expel scandal plagu rep georg santo r n friday make first hous lawmak expe
'german chancellor olaf scholz formal announc wednesday week stall frustrat negoti berlin ag
'former presid donald trump indict part manhattan district attorney offic year long investig
'washington suprem court thursday struck affirm action program univers north carolina harvar
'u militari monitor suspect chines surveil balloon hover northern u past day militari defens
'differ poll agre presid joe biden polit stand lower barack obama point elect even lower don
'ashkelon israel israel plung chao saturday palestinian milit group hama launch deadli land
'dozen leak defens depart classifi document post onlin reveal detail u spi russia war machin
'washington hous vote overwhelmingli expel indict rep georg santo friday pull curtain tempes
'ukrain set receiv battl tank germani western countri fierc debat expos fissur among alli al
'grand juri new york citi vote thursday indict donald trump first time former u presid face
'suprem court struck affirm action program harvard univers univers north carolina thursday i
'us track chines spi balloon float northern part countri day pentagon offici announc thursda
'presid biden readi ring new year see number year old command chief end lower approv rate se
'jerusalem ap hama milit fire thousand rocket sent dozen fighter isra town near gaza strip u
'massachusett air nation guardsman jack teixeira leader discord group dozen sensit us intell
'bye georg lie long island rep georg santo r ny becam sixth member ever expel us hous repres
'week excus foot drag german final said ye transfer limit number leopard tank ukrain agre wo
'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 app
'democrat battleground state grow increasingli anxiou presid biden low approv rate worri vot
'israel news site compil list dead miss funer take place around countri weekend attack confu
'washington would year old nation guardsman posit access top secret document begin dramat ar
'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 u offici disclos thursda
'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 u offici foreign alli scrambl understand dozen classifi intellig document end inte
'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 u hi
'thursday decis forc rework admiss criteria throughout american higher educ decad pursuit di
'washington u track offici describ chines reconnaiss balloon continent state week would aggr
'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 u govern scrambl protect secret unauthor disclosur appear provid detail
'lawmak vote remov two third hous supermajor requir constitut almost democrat mani republica
'u germani outlin plan wednesday send dozen modern battl tank ukrain mark signific new infus
'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	trump	biden	offici	mr	u	israel	presid	tank	hous	...	messr	overlap	vs	conver
0	5.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	30.0	0.0	5.0	28.0	0.0	15.0	0.0	4.0	0.0	3.0	...	0.0	0.0	0.0	0.0
2	9.0	17.0	27.0	0.0	0.0	0.0	0.0	8.0	0.0	4.0	...	0.0	0.0	0.0	0.0
3	17.0	0.0	3.0	6.0	0.0	10.0	28.0	2.0	0.0	1.0	...	0.0	0.0	0.0	0.0

	said	trump	biden	offici	mr	u	israel	presid	tank	hous	...	messr	overlap	vs	conver
4	9.0	0.0	0.0	5.0	0.0	20.0	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	trump	biden	offici	mr	u	israel	presid	tank	hous	..
0	1.827036	0.000000	0.000000	0.000000	0.0	2.313275	0.000000	0.000000	0.000000	0.000000	..
1	3.081563	0.000000	2.267700	4.139471	0.0	3.594586	0.000000	1.881491	0.000000	1.871958	..
2	2.238584	5.086179	3.733245	0.000000	0.0	0.000000	0.000000	2.428008	0.000000	2.128570	..
3	2.683881	0.000000	1.823774	2.667558	0.0	3.201528	6.446654	1.334974	0.000000	0.891998	..
4	2.238584	0.000000	0.000000	2.493348	0.0	3.873465	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
 'u 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
```

```
# 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 0x173f00b20>
```

article_source	said	trump	biden	offici	mr	u	israel	presid
ABC News	3.591249	2.959536	3.099282	3.139834	1.173600	3.586612	2.824926	2.8861
BBC	3.280434	2.780642	2.335743	2.460363	3.283902	0.000000	2.460363	2.6543
CNN	3.698474	3.025425	2.976653	3.331000	0.693147	1.173600	3.225542	2.9933
Fox News	3.055995	3.350161	2.824926	2.421451	0.693147	3.027179	2.866350	2.9052
NBC News	3.670441	3.311249	2.866350	2.654383	1.654053	3.445144	3.099282	2.5317
New York Post	3.340652	3.126599	2.757300	2.335743	1.654053	2.924302	3.085175	3.0095
The New York Times	3.584733	3.269825	2.993325	2.654383	4.166255	1.935100	1.347002	2.8249
The Washington Post	3.785551	3.269825	3.269825	3.225542	0.000000	3.376459	2.976653	3.0992
The Wall Street Journal	3.849334	3.213976	2.866350	3.126599	3.919027	3.667067	2.905262	2.8031

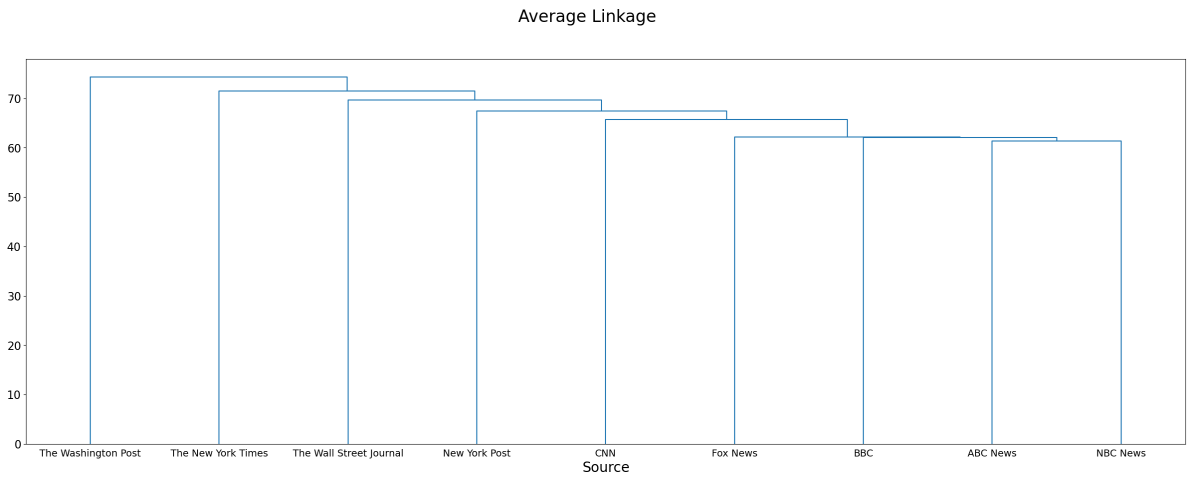
```
# create a dataframe of token binary values with each row representing all articles of one

# 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
```

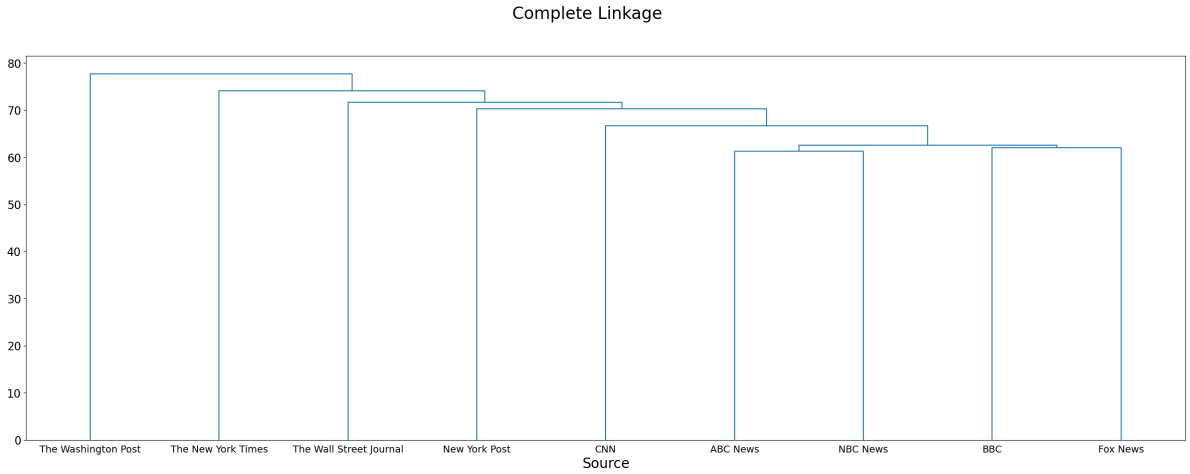
```
<keras.src.preprocessing.text.Tokenizer object at 0x175247b50>
```

article_source	said	trump	biden	offici	mr	u	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	0.0	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	0.0	1.0	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

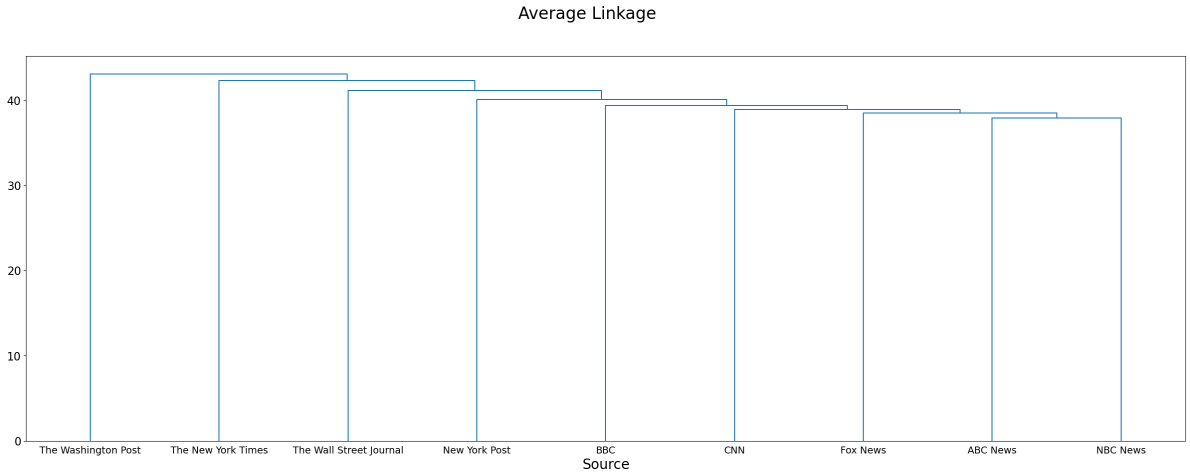
```
# create dendrogram with average linkage from tf_idf scores df, whose rows each represent

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()
```



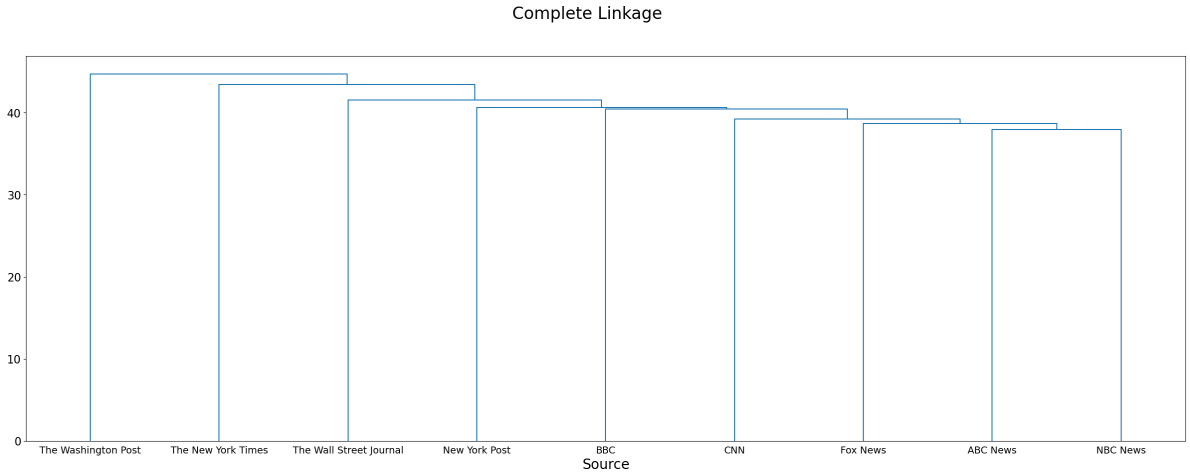
```
# create dendrogram with complete linkage from tf_idf scores df, whose rows each represent

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)
```

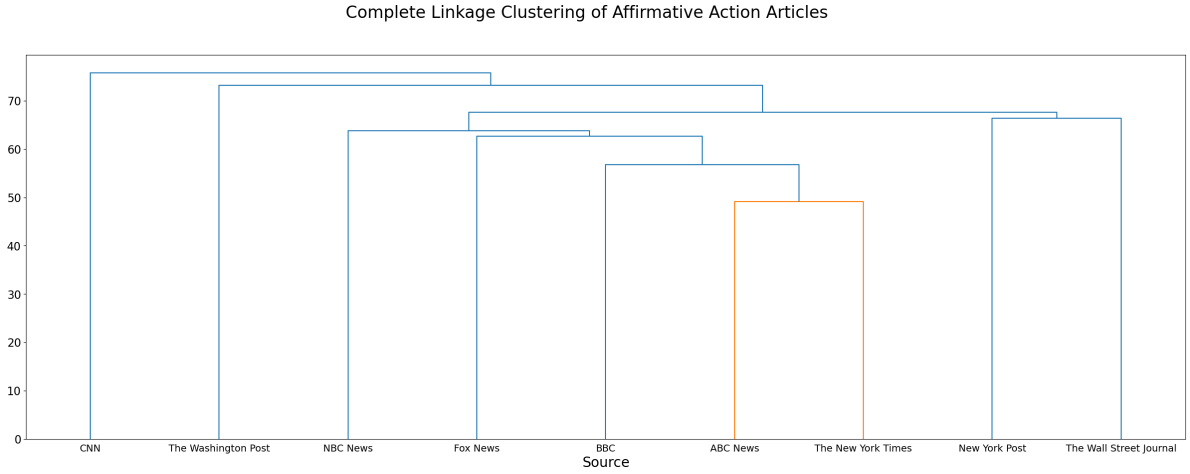
```
affirm_tfidf_df.head()
```

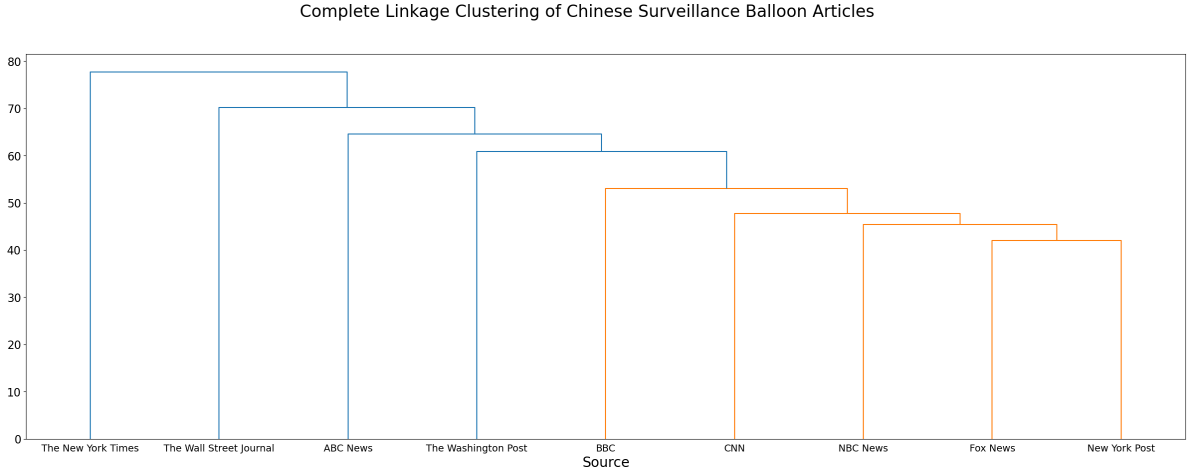
article_source	said	trump	biden	offici	mr	u	israel	presid	tank	hous
ABC News	1.827036	0.000000	0.000000	0.0	0.000000	2.313275	0.0	0.000000	0.0	0.000
BBC	2.379087	1.326871	0.869038	0.0	2.295628	0.000000	0.0	2.057431	0.0	0.891
CNN	2.641434	2.784588	0.000000	0.0	0.000000	0.000000	0.0	1.334974	0.0	0.891
Fox News	1.827036	0.000000	0.869038	0.0	0.000000	1.641338	0.0	0.788457	0.0	0.000
NBC News	2.156116	2.246588	2.073780	0.0	0.000000	1.641338	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)
```

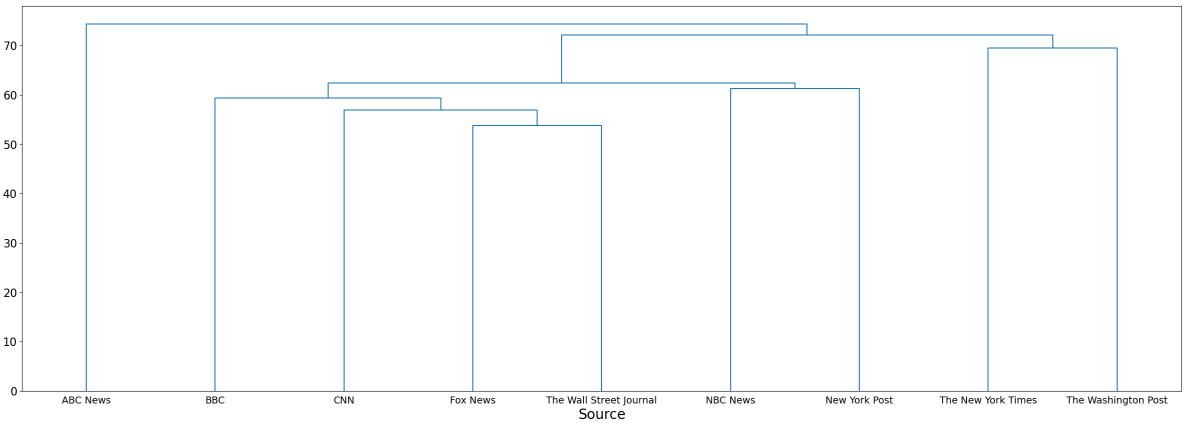
```
plt.xticks(fontsize = 14)
plt.show()

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()
```

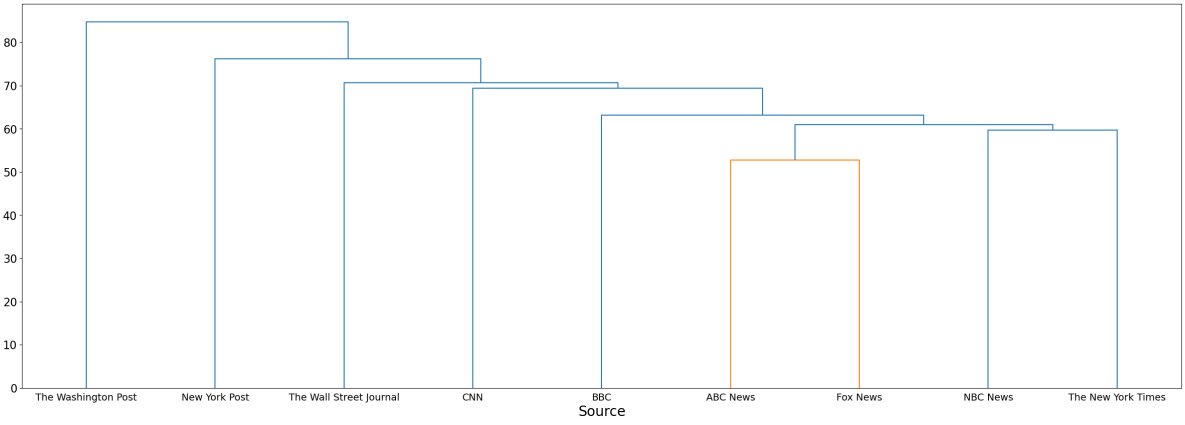




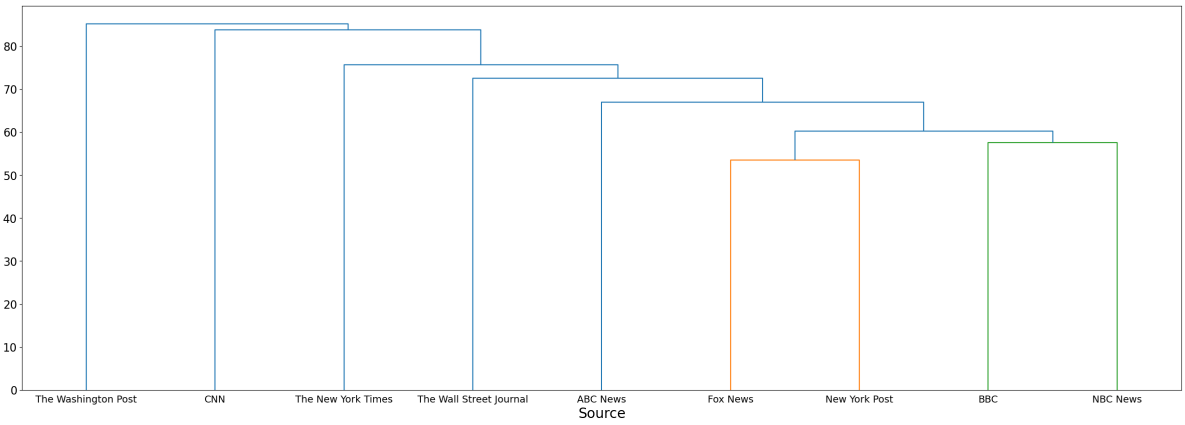
Complete Linkage Clustering of Biden's Low Ratings Articles



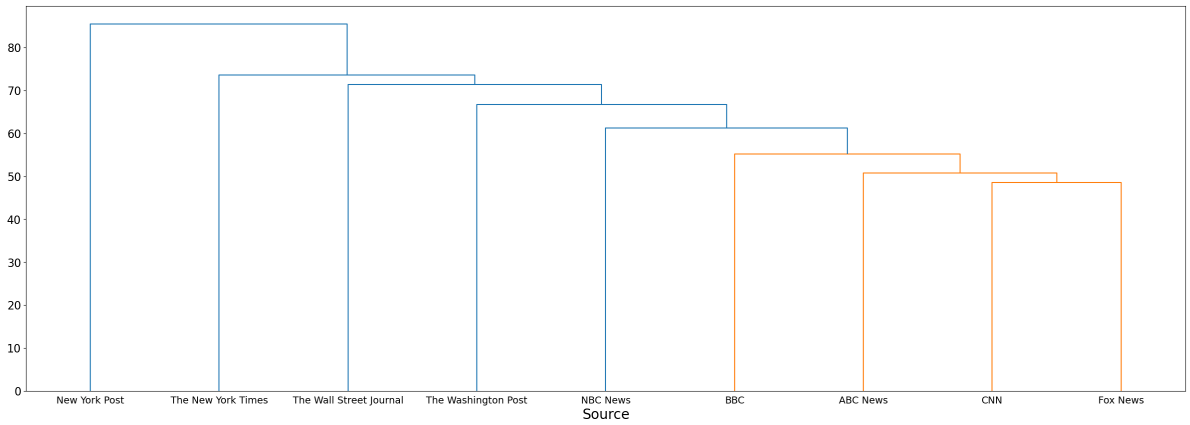
Complete Linkage Clustering of Hamas Attack Articles



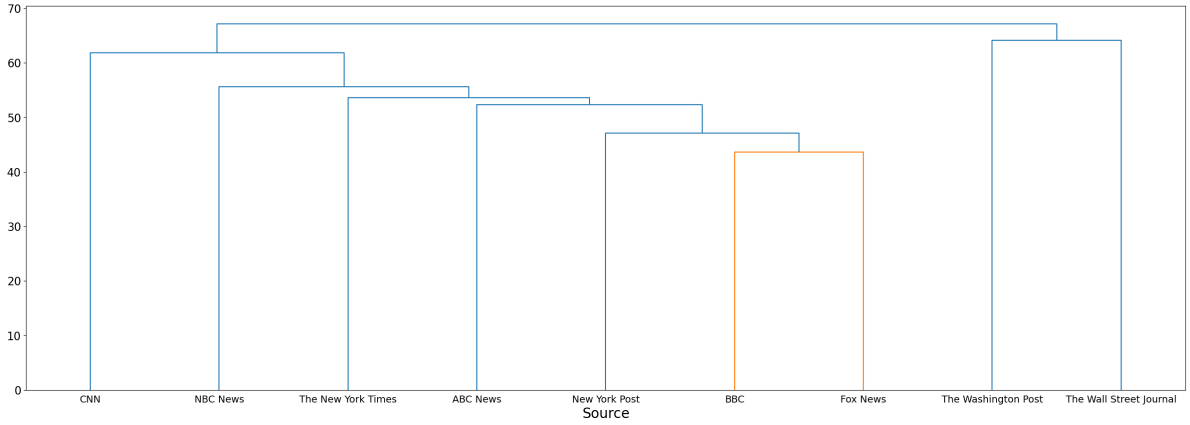
Complete Linkage Clustering of Pentagon Document Leak Articles



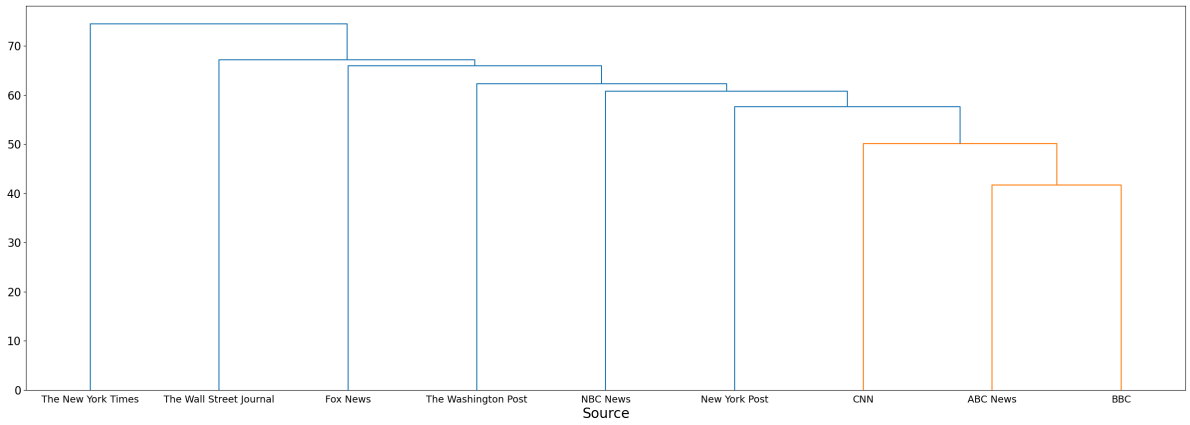
Complete Linkage Clustering of George Santos Expulsion Articles



Complete Linkage Clustering of Sending Ukraine Tanks Articles

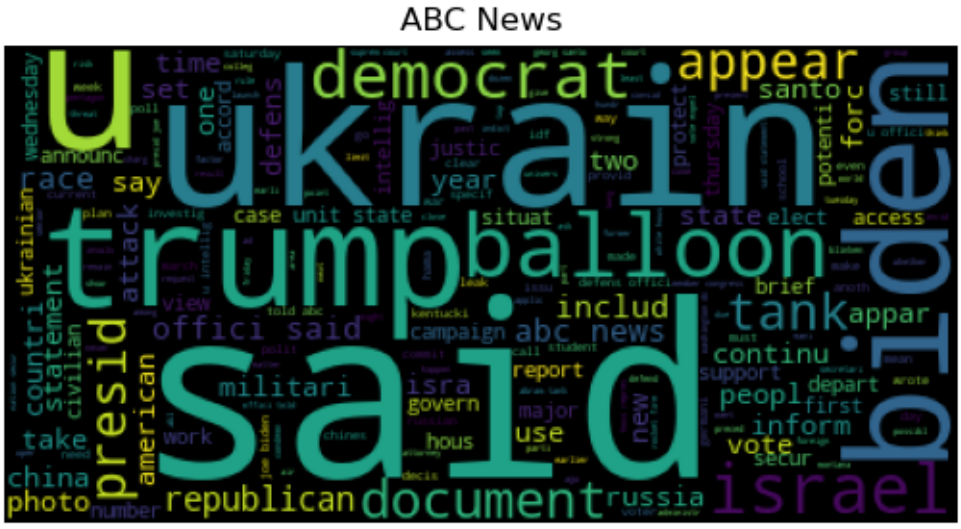


Complete Linkage Clustering of Trump's Indictment Articles



```
def create_wordcloud(text, title):
    '''Given a string of all text and a string
    for the title, creates a wordcloud.'''
    plt.figure()
    wc = WordCloud().generate(text)
    plt.title(title)
    plt.axis("off")
    plt.imshow(wc)
```

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



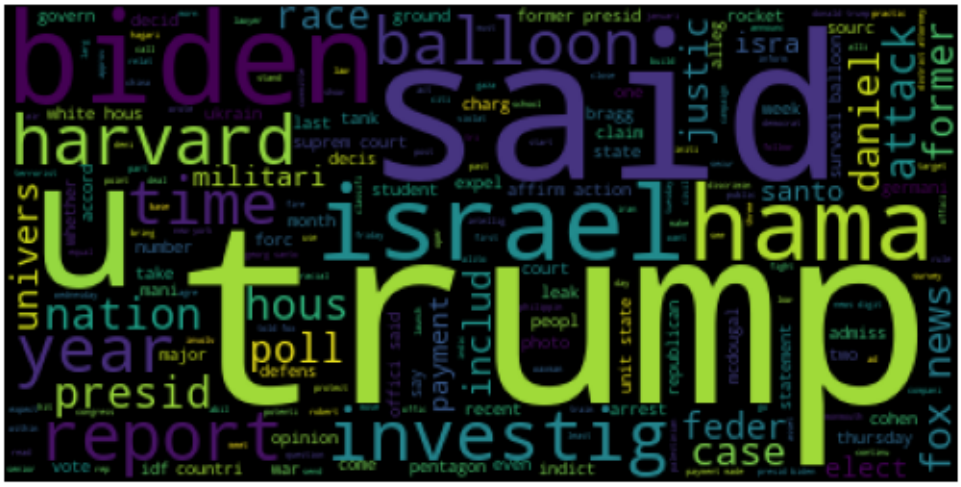
BBC


said
us
ukrain
elect
gaza
time
one
includ
document
israel
isra
militari
offici
report
decis
call
countri
tank
attack
democrat
trump
justic
year
mr
say
case
made
balloon
republican
campaign
hama
palestinian
first
former
presid

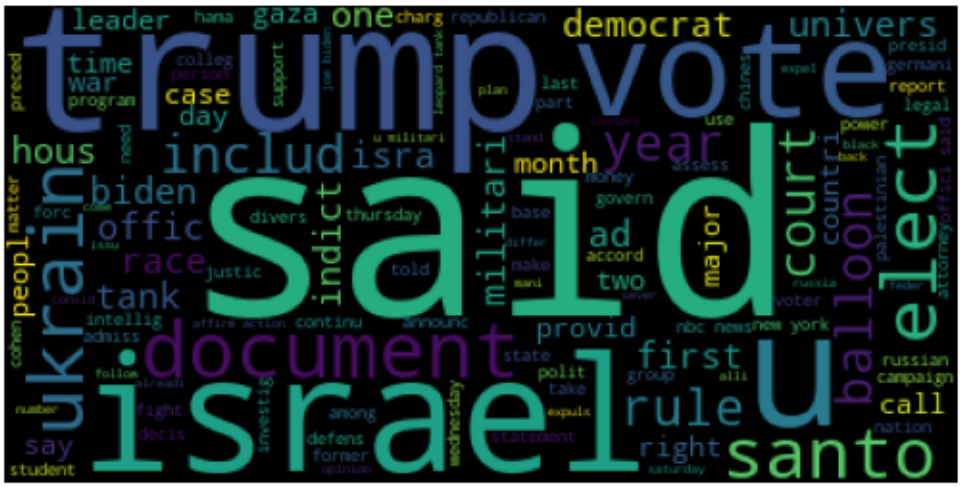
CNN


said
u
israel
ukrain
document
includ
report
trump
militari
intellig
gaza
biden
court
isra
offici said
decis
hama
hous
chines
countri
american
system
vote
peopl
day
leak
balloon
call
respons
major
inform
two
person
say
cnn
first
harvard
defens
govern
indict
justic
senior
teixeira
month
support
use
effort
accord
told cnn
alli
take
right
assess
come
investig
one
santo
make
democrat

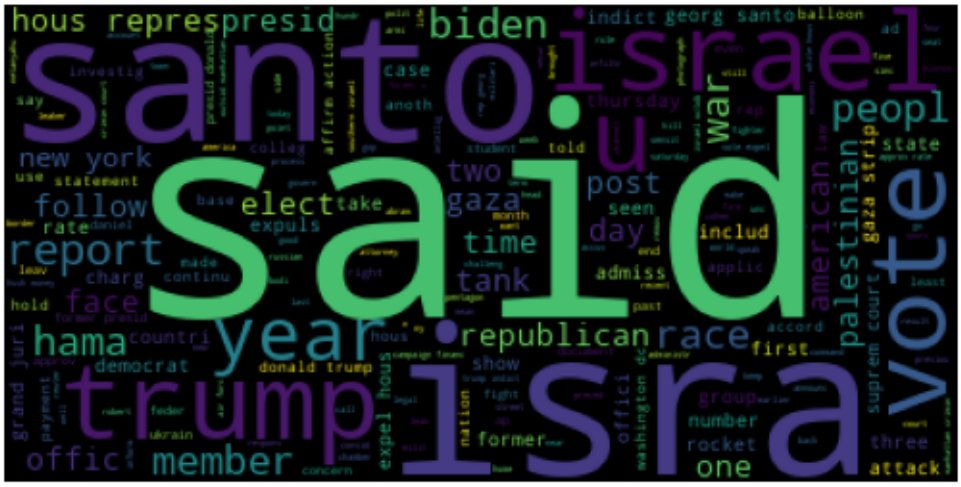
Fox News



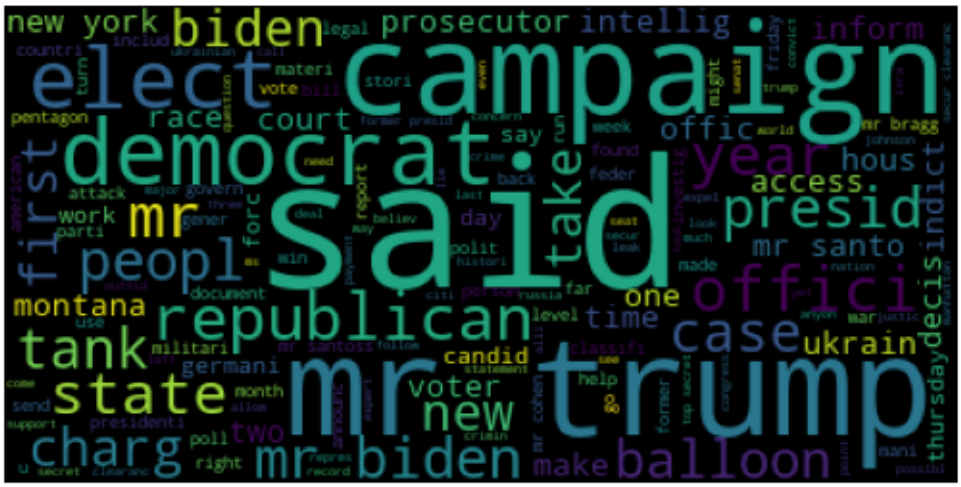
NBC News



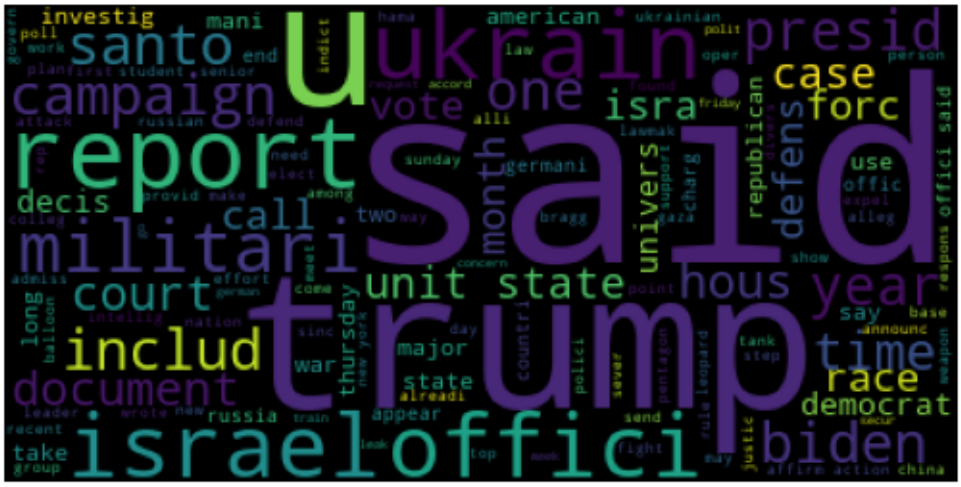
New York Post



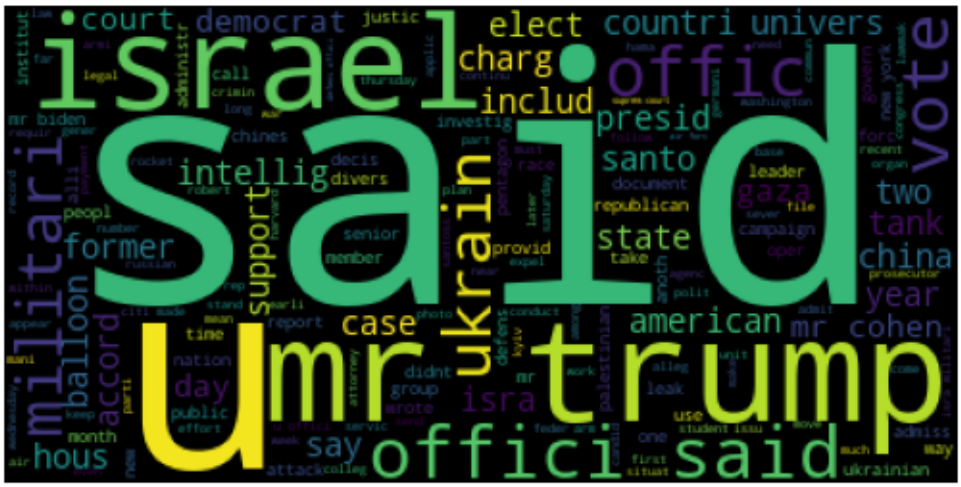
The New York Times



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 u intelligence documents might begin
 'house representatives friday voted expel republican rep george santos historic move happene
 'major increase u support ukraine president joe biden signed sending abrams tanks war torn c
 '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
 'u supreme court handed major ruling affirmative action thursday rejecting use race factor c
 'u 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
 'u military monitoring suspected chinese surveillance balloon hovering northern u past days
 '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 u spying
'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 u president i
'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 appi
'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 u offi
'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 u officials foreign allies scrambled understand dozens classified intelligence doc
'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 u 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 u government scrambles protect secrets unauthorized disclosures app
'lawmakers voted remove two thirds house supermajority required constitution almost democrat
'u 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
```

```
# 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
 'u 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 thu
 'supreme court thursday held race conscious admissions programs harvard university north ca
 'thursdays decision force reworking admissions criteria throughout american higher educatio
```

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

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

polarity_scores = []
subjectivity_scores = []

for article in article_docs_unstemmed:
    polarity_scores.append(round(TextBlob(article).sentiment.polarity, 2))
    subjectivity_scores.append(round(TextBlob(article).sentiment.subjectivity, 2))

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.iloc[64:, :]
```

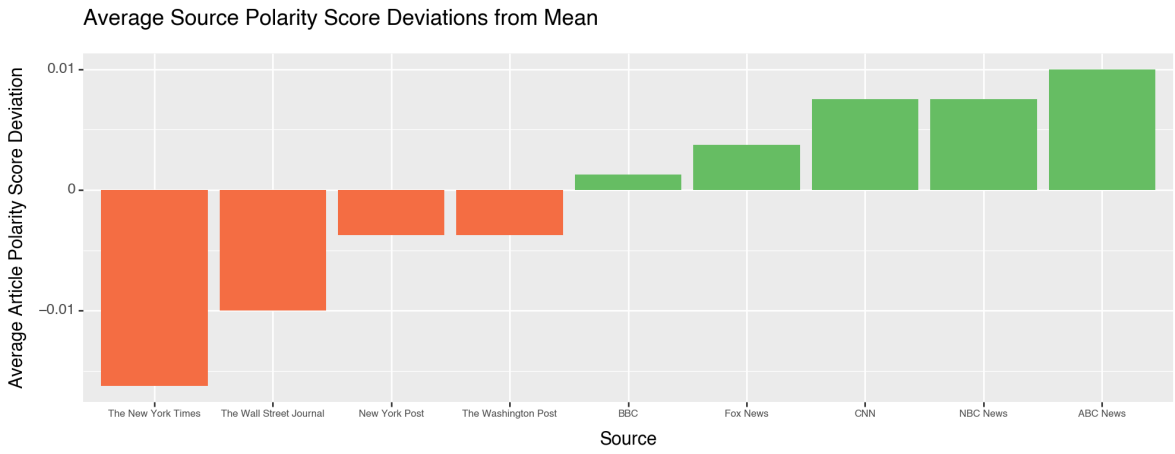
	source	topic	polarity_score	subjectivity_s
64	The Wall Street Journal	Supreme Court Ruling on Affirmative Action	0.07	0.39
65	The Wall Street Journal	Chinese Surveillance Balloon	0.02	0.28
66	The Wall Street Journal	Biden's Low Approval Rates in Polls	0.07	0.42
67	The Wall Street Journal	The Deadliest Attack by Hamas	0.04	0.32
68	The Wall Street Journal	Pentagon Documents Leak	0.05	0.39
69	The Wall Street Journal	George Santos' Expulsion from Congress	-0.04	0.35
70	The Wall Street Journal	U.S. and Germany Send Tanks to Ukraine	0.11	0.32
71	The 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.sort_values(by='polarity_dev', ascending=True, inplace=True)
polarity_devs.reset_index(drop=True, inplace=True)

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

polarity_devs
```

	source	polarity_dev	sign
0	The New York Times	-0.01625	neg
1	The Wall Street Journal	-0.01000	neg
2	New York Post	-0.00375	neg
3	The Washington Post	-0.00375	neg
4	BBC	0.00125	pos

	source	polarity_dev	sign
5	Fox News	0.00375	pos
6	CNN	0.00750	pos
7	NBC News	0.00750	pos
8	ABC News	0.01000	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

colors = {"pos":'#66bd63', "neg":'#f46d43'}

(
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=6, face='bold'), legend_posit
+ scale_fill_manual(values = colors)
)
```



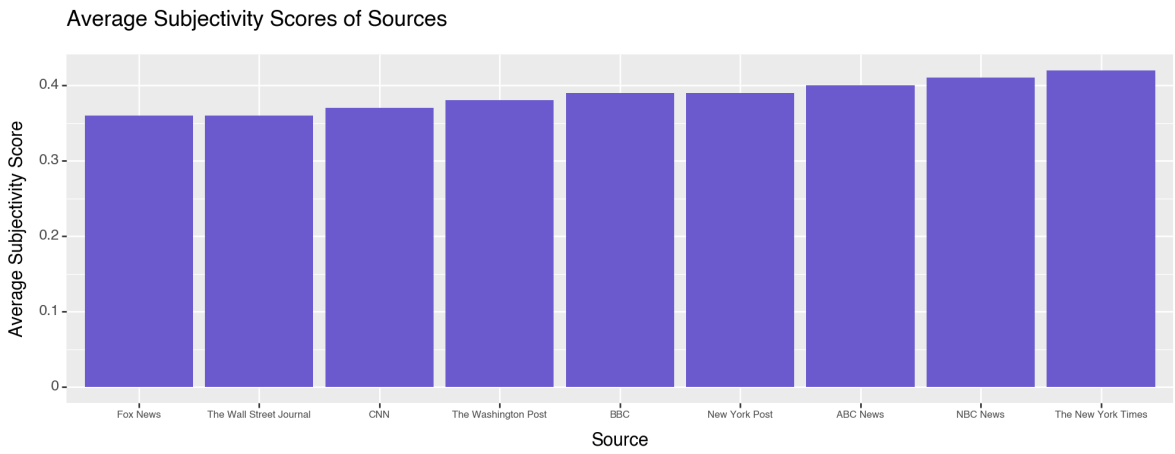
<Figure Size: (1000 x 400)>

```
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", "The Wall Street Journal", "CNN",
     "The Washington Post", "BBC", "New York Post", "ABC News", "NBC News", "The New York
    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	The Wall Street Journal	0.36
2	CNN	0.37
3	The Washington Post	0.38
4	BBC	0.39
5	New York Post	0.39
6	ABC News	0.40
7	NBC News	0.41
8	The New York Times	0.42

```
(
ggplot(source_avg_subj, aes(x="source", y="avg_subj"))
+ geom_col(stat="identity", fill="slateblue")
+ labs(x='Source', y='Average Subjectivity Score', title='Average Subjectivity Scores of S
+ 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)>

```
# 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), ('The New York Times', 'Supreme
6
```

```
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:8: FutureWarni
/var/folders/52/nc0rbx9j5dz6nlj70gy7hwjr0000gn/T/ipykernel_23386/4033784056.py:9: FutureWarni
```

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.

```python
# standardize the frequency df before doing PCA

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)
```

```
{'n_components': None,
 'copy': True,
 'whiten': False,
 'svd_solver': 'auto',
```

```
'tol': 0.0,
'iterated_power': 'auto',
'n_oversamples': 10,
'power_iteration_normalizer': 'auto',
'random_state': None,
'n_features_in_': 5306,
'_fit_svd_solver': 'full',
'mean_': array([-1.48029737e-16, -4.93432455e-17,  2.46716228e-17, ...,
        1.23358114e-17,  1.23358114e-17,  1.23358114e-17]),
'noise_variance_': 0.0,
'n_samples_': 72,
'components_': array([[-1.88359604e-02,  8.24539565e-03,  1.56857262e-04, ...,
         2.02861862e-03,  2.02861862e-03,  2.02861862e-03],
       [-1.10805660e-02,  2.10463795e-02,  2.99679367e-03, ...,
         7.22054293e-03,  7.22054293e-03,  7.22054293e-03],
       [-5.02156260e-03,  1.16691854e-03,  4.85557051e-03, ...,
         3.85641727e-04,  3.85641727e-04,  3.85641727e-04],
       ...,
       [-6.88162567e-03,  1.14631663e-03, -7.06141147e-03, ...,
         9.22884264e-04,  9.22884264e-04,  9.22884264e-04],
       [-6.90424766e-03,  1.23208279e-02, -1.07385147e-02, ...,
         5.38411364e-04,  5.38411364e-04,  5.38411364e-04],
       [ 6.38193715e-01,  2.59287176e-01,  1.52262058e-01, ...,
        -1.42203817e-03, -1.42203817e-03, -1.42203817e-03]]),
'n_components_': 72,
'explained_variance_': array([2.26902950e+02, 2.04939693e+02, 1.89283924e+02, 1.65001148e+02
       1.56614852e+02, 1.48311624e+02, 1.33946292e+02, 1.33577832e+02,
       1.25391931e+02, 1.21385801e+02, 1.15013036e+02, 1.11256773e+02,
       1.06803248e+02, 1.04310241e+02, 1.02676573e+02, 9.97698162e+01,
       9.96061771e+01, 9.81565633e+01, 9.65658878e+01, 9.53559665e+01,
       9.04267126e+01, 8.66536798e+01, 8.53005648e+01, 8.40421697e+01,
       7.93288462e+01, 7.73646863e+01, 7.65067762e+01, 7.59952841e+01,
       7.31591258e+01, 7.28623385e+01, 7.12317978e+01, 6.97398188e+01,
       6.88548969e+01, 6.61881001e+01, 6.55980534e+01, 6.48360060e+01,
       6.38739108e+01, 6.20888397e+01, 6.08520700e+01, 6.01956334e+01,
       5.92800974e+01, 5.81872510e+01, 5.67463487e+01, 5.60134997e+01,
       5.50923266e+01, 5.40228749e+01, 5.32679947e+01, 5.17009494e+01,
       5.03730813e+01, 4.92228264e+01, 4.84409295e+01, 4.58700650e+01,
       4.46134474e+01, 4.39175778e+01, 4.34407892e+01, 4.21325109e+01,
       4.08260615e+01, 4.03927443e+01, 3.96177025e+01, 3.76327889e+01,
       3.75816028e+01, 3.69445949e+01, 3.40401678e+01, 3.30681653e+01,
       3.21037262e+01, 3.14309735e+01, 2.97196802e+01, 2.72051613e+01,
       2.43372938e+01, 1.97340694e+01, 1.38054516e+01, 7.28092092e-27]),
```

```
'explained_variance_ratio_': array([4.21695289e-02, 3.80876947e-02, 3.51780967e-02, 3.066518
       2.91066049e-02, 2.75634641e-02, 2.48936915e-02, 2.48252137e-02,
       2.33038780e-02, 2.25593455e-02, 2.13749779e-02, 2.06768827e-02,
       1.98492026e-02, 1.93858816e-02, 1.90822672e-02, 1.85420513e-02,
       1.85116393e-02, 1.82422310e-02, 1.79466067e-02, 1.77217448e-02,
       1.68056513e-02, 1.61044396e-02, 1.58529655e-02, 1.56190949e-02,
       1.47431317e-02, 1.43780959e-02, 1.42186547e-02, 1.41235948e-02,
       1.35964996e-02, 1.35413422e-02, 1.32383090e-02, 1.29610272e-02,
       1.27965659e-02, 1.23009463e-02, 1.21912871e-02, 1.20496619e-02,
       1.18708581e-02, 1.15391057e-02, 1.13092541e-02, 1.11872565e-02,
       1.10171057e-02, 1.08140020e-02, 1.05462128e-02, 1.04100140e-02,
       1.02388156e-02, 1.00400598e-02, 9.89976657e-03, 9.60853387e-03,
       9.36175183e-03, 9.14797890e-03, 9.00266469e-03, 8.52487387e-03,
       8.29133362e-03, 8.16200744e-03, 8.07339707e-03, 7.83025577e-03,
       7.58745437e-03, 7.50692310e-03, 7.36288289e-03, 6.99399006e-03,
       6.98447720e-03, 6.86609038e-03, 6.32630752e-03, 6.14566250e-03,
       5.96642313e-03, 5.84139317e-03, 5.52335221e-03, 5.05603313e-03,
       4.52304482e-03, 3.66754337e-03, 2.56571979e-03, 1.35314682e-30]),
'singular_values_': array([1.26925606e+02, 1.20626358e+02, 1.15927385e+02, 1.08236230e+02,
       1.05449772e+02, 1.02616399e+02, 9.75201863e+01, 9.73859644e+01,
       9.43547938e+01, 9.28352945e+01, 9.03655110e+01, 8.88776173e+01,
       8.70805982e+01, 8.60582774e+01, 8.53817118e+01, 8.41644637e+01,
       8.40954135e+01, 8.34812314e+01, 8.28020412e+01, 8.22816725e+01,
       8.01267533e+01, 7.84373079e+01, 7.78224910e+01, 7.72463206e+01,
       7.50489712e+01, 7.41140522e+01, 7.37019749e+01, 7.34551916e+01,
       7.20714780e+01, 7.19251419e+01, 7.11158045e+01, 7.03670885e+01,
       6.99192225e+01, 6.85518425e+01, 6.82455991e+01, 6.78480392e+01,
       6.73427625e+01, 6.63950873e+01, 6.57304874e+01, 6.53749950e+01,
       6.48759348e+01, 6.42751493e+01, 6.34743315e+01, 6.30631309e+01,
       6.25424271e+01, 6.19324157e+01, 6.14981920e+01, 6.05868584e+01,
       5.98037522e+01, 5.91170083e+01, 5.86455966e+01, 5.70681576e+01,
       5.62810338e+01, 5.58403799e+01, 5.55364388e+01, 5.46937681e+01,
       5.38391156e+01, 5.35526362e+01, 5.30363731e+01, 5.16906956e+01,
       5.16555302e+01, 5.12158788e+01, 4.91614881e+01, 4.84545120e+01,
       4.77426912e+01, 4.72398044e+01, 4.59357953e+01, 4.39495899e+01,
       4.15685922e+01, 3.74315232e+01, 3.13079393e+01, 7.18989141e-13])}
```

```
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)
```