# import just a function   
from urllib.request import urlopen  
from bs4 import BeautifulSoup  
  
# import a whole library as a new name  
import pandas as pd  
  
# import package as its own name  
import nltk  
  
# other packages  
import re  
   
# impurity function  
RE\_SUSPICIOUS = re.compile(r'[&#<>{}\[\]\\]')  
  
def impurity(text, min\_len=10):  
 """returns the share of suspicious characters in a text"""  
 if text == None or len(text) < min\_len:  
 return 0  
 else:  
 return len(RE\_SUSPICIOUS.findall(text))/len(text)  
  
# rest of stuff  
import textacy.preprocessing as tprep  
  
def normalize(text):  
 text = tprep.normalize.hyphenated\_words(text)  
 text = tprep.normalize.quotation\_marks(text)  
 text = tprep.normalize.unicode(text)  
 text = tprep.remove.accents(text)  
 text = tprep.replace.phone\_numbers(text)  
 text = tprep.replace.urls(text)  
 text = tprep.replace.emails(text)  
 text = tprep.replace.user\_handles(text)  
 text = tprep.replace.emojis(text)  
 return text  
   
# install pyspellchecker !!!  
from spellchecker import SpellChecker  
spell = SpellChecker()  
  
import spacy  
nlp = spacy.load("en\_core\_web\_sm")  
import textacy  
  
from itertools import chain  
from collections import Counter

# import just a function   
from urllib.request import urlopen  
from bs4 import BeautifulSoup  
  
# import a whole library as a new name  
import pandas as pd  
  
# import package as its own name  
import nltk  
  
# other packages  
import re  
   
# impurity function  
RE\_SUSPICIOUS = re.compile(r'[&#<>{}\[\]\\]')  
  
def impurity(text, min\_len=10):  
 """returns the share of suspicious characters in a text"""  
 if text == None or len(text) < min\_len:  
 return 0  
 else:  
 return len(RE\_SUSPICIOUS.findall(text))/len(text)  
  
# rest of stuff  
import textacy.preprocessing as tprep  
  
def normalize(text):  
 text = tprep.normalize.hyphenated\_words(text)  
 text = tprep.normalize.quotation\_marks(text)  
 text = tprep.normalize.unicode(text)  
 text = tprep.remove.accents(text)  
 text = tprep.replace.phone\_numbers(text)  
 text = tprep.replace.urls(text)  
 text = tprep.replace.emails(text)  
 text = tprep.replace.user\_handles(text)  
 text = tprep.replace.emojis(text)  
 return text  
   
# install pyspellchecker !!!  
from spellchecker import SpellChecker  
spell = SpellChecker()  
  
import spacy  
nlp = spacy.load("en\_core\_web\_sm")  
import textacy  
  
from itertools import chain  
from collections import Counter

# myurl = "https://www.foxnews.com/sports/patrick-mahomes-fiery-message-win-bills-they-got-what-they-asked-for"  
# #myurl = "https://www.foxnews.com/lifestyle/newly-elected-school-board-pennsylvania-reclaims-indigenous-mascot-rejects-cancel-culture"  
  
# html = urlopen(myurl).read()  
  
# soupified = BeautifulSoup(html, "html.parser")  
# # soupified  
  
# # just try get\_text()  
# try\_text = soupified.get\_text()  
# try\_text[0:100]  
  
from urllib.request import urlopen  
from bs4 import BeautifulSoup  
import socket  
  
# Set the timeout duration in seconds  
timeout\_value = 100 # Specify the desired timeout value  
  
# Set the URL to fetch  
myurl = "https://www.foxnews.com/sports/patrick-mahomes-fiery-message-win-bills-they-got-what-they-asked-for"  
# myurl = "https://themightymarketer.com/10-ways-to-get-500-connections-on-linkedin-fast/"  
  
# Open the URL with a timeout  
try:  
 html = urlopen(myurl, timeout=timeout\_value).read()  
 soupified = BeautifulSoup(html, "html.parser")  
 try\_text = soupified.get\_text()  
 print(try\_text[0:100]) # Print the first 100 characters of the text  
except socket.timeout:  
 print("Timeout error: The request took too long to complete.")

Patrick Mahomes' fiery message after win over Bills: 'They got what they asked for' | Fox News

# find an exact match for the first time this occurs  
text = try\_text[  
 # everything from the end of this sentence and on  
 re.search("To access the content, check your email and follow the instructions provided.", try\_text).end():  
 # re.search("LinkedIn has chosen 500 as the “magic” number of connections.", try\_text).end():  
 # now the end  
 # re.search("And having connections in common with the person who is doing the search is the #2 criterion LinkedIn uses to rank search results.", try\_text).start()  
 re.search("CLICK HERE TO GET THE FOX NEWS APP", try\_text).start()  
]

# break down into sentences and put into DF  
import nltk  
nltk.download('punkt')  
  
sentences = nltk.sent\_tokenize(text)  
type(sentences)  
  
# convert to dataframe  
DF = pd.DataFrame(sentences, columns = ["sentence"])  
DF.head()

[nltk\_data] Downloading package punkt to  
[nltk\_data] C:\Users\user7\AppData\Roaming\nltk\_data...  
[nltk\_data] Unzipping tokenizers\punkt.zip.

sentence  
0 \n Having trouble?  
1 Click here.  
2 Kansas City Chiefs quarterback Patrick Mahomes...  
3 Mahomes, surrounded by Travis Kelce, Chris Jon...  
4 They have the Baltimore Ravens up next.

# do this on the full text not broken into sentences  
len(nltk.word\_tokenize(text))  
# be sure to import nltk in the proposal

554

DF['score'] = DF['sentence'].apply(impurity)  
DF

sentence score  
0 \n Having trouble? 0.000000  
1 Click here. 0.000000  
2 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000  
3 Mahomes, surrounded by Travis Kelce, Chris Jon... 0.000000  
4 They have the Baltimore Ravens up next. 0.000000  
5 "Hey, they asked for it, and they got what the... 0.000000  
6 "But he (Andy Reid) said it. 0.000000  
7 This s--- ain’t done. 0.000000  
8 We come back next week ready to f---ing go." 0.000000  
9 CLICK HERE FOR MORE SPORTS COVERAGE ON FOXNEWS... 0.000000  
10 VIEW THE MOMENT ON X.Reid said it’s not time f... 0.000000  
11 They still have to get through the Ravens and ... 0.000000  
12 The Ravens are going to be the Chiefs’ toughes... 0.000000  
13 Kansas City will have to do it on the road. 0.000000  
14 The Ravens are hosting a conference championsh... 0.002222  
15 Baltimore boasts one of the toughest defenses ... 0.000000  
16 "There’s no weakness there," the star quarterb... 0.000000  
17 "It’s going to take our best effort. 0.000000  
18 Defense, offense, special teams. 0.000000  
19 They do it all. 0.000000  
20 It’s always a great challenge, and that stadiu... 0.000000  
21 So, we’re excited for the challenge. 0.000000  
22 "On Monday, Mahomes added that he praised the ... 0.000000  
23 "For three quarters offensively, we were movin... 0.000000  
24 "I went over to the defense and told them: ‘Y’... 0.000000  
25 We’ll get to the AFC championship game." 0.000000  
26 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000

DF['clean'] = DF['sentence'].apply(normalize)  
DF

sentence score \  
0 \n Having trouble? 0.000000   
1 Click here. 0.000000   
2 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000   
3 Mahomes, surrounded by Travis Kelce, Chris Jon... 0.000000   
4 They have the Baltimore Ravens up next. 0.000000   
5 "Hey, they asked for it, and they got what the... 0.000000   
6 "But he (Andy Reid) said it. 0.000000   
7 This s--- ain’t done. 0.000000   
8 We come back next week ready to f---ing go." 0.000000   
9 CLICK HERE FOR MORE SPORTS COVERAGE ON FOXNEWS... 0.000000   
10 VIEW THE MOMENT ON X.Reid said it’s not time f... 0.000000   
11 They still have to get through the Ravens and ... 0.000000   
12 The Ravens are going to be the Chiefs’ toughes... 0.000000   
13 Kansas City will have to do it on the road. 0.000000   
14 The Ravens are hosting a conference championsh... 0.002222   
15 Baltimore boasts one of the toughest defenses ... 0.000000   
16 "There’s no weakness there," the star quarterb... 0.000000   
17 "It’s going to take our best effort. 0.000000   
18 Defense, offense, special teams. 0.000000   
19 They do it all. 0.000000   
20 It’s always a great challenge, and that stadiu... 0.000000   
21 So, we’re excited for the challenge. 0.000000   
22 "On Monday, Mahomes added that he praised the ... 0.000000   
23 "For three quarters offensively, we were movin... 0.000000   
24 "I went over to the defense and told them: ‘Y’... 0.000000   
25 We’ll get to the AFC championship game." 0.000000   
26 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000   
  
 clean   
0 \n Having trouble?   
1 Click here.   
2 Kansas City Chiefs quarterback Patrick Mahomes...   
3 Mahomes, surrounded by Travis Kelce, Chris Jon...   
4 They have the Baltimore Ravens up next.   
5 "Hey, they asked for it, and they got what the...   
6 "But he (Andy Reid) said it.   
7 This s--- ain't done.   
8 We come back next week ready to f---ing go."   
9 CLICK HERE FOR MORE SPORTS COVERAGE ON FOXNEWS...   
10 VIEW THE MOMENT ON X.Reid said it's not time f...   
11 They still have to get through the Ravens and ...   
12 The Ravens are going to be the Chiefs' toughes...   
13 Kansas City will have to do it on the road.   
14 The Ravens are hosting a conference championsh...   
15 Baltimore boasts one of the toughest defenses ...   
16 "There's no weakness there," the star quarterb...   
17 "It's going to take our best effort.   
18 Defense, offense, special teams.   
19 They do it all.   
20 It's always a great challenge, and that stadiu...   
21 So, we're excited for the challenge.   
22 "On Monday, Mahomes added that he praised the ...   
23 "For three quarters offensively, we were movin...   
24 "I went over to the defense and told them: 'Y'...   
25 We'll get to the AFC championship game."   
26 Kansas City Chiefs quarterback Patrick Mahomes...

# find all the unique tokens  
# set is find unique  
# nltk.word\_tokenize is break down into words  
# " ".join is combine into one long text  
# .to\_list() is a function to convert to list   
clean\_tokens = set(nltk.word\_tokenize(" ".join(DF['clean'].to\_list())))  
# what is wrong?   
misspelled = spell.unknown(clean\_tokens)  
  
for word in misspelled:  
 # what's the word  
 print(word)  
 print("\n")  
 # Get the one `most likely` answer  
 print(spell.correction(word))  
  
 # Get a list of `likely` options  
 print(spell.candidates(word))  
   
# make a dictionary of the misspelled word and the correction  
# use find and replace in re to fix them

sunday  
  
  
sundae  
{'sundry', 'sundae'}  
jan.  
  
  
jane  
{'jane'}  
``  
  
  
i  
{'mu', 'kw', 'my', 'up', 'si', 'at', 'to', 'h', 'pa', 'on', 'u', 'bi', 'ta', 't', 'ne', 'ls', 'uh', 'dm', 'ha', 'g', 'he', 'hg', 'fa', 's', 'pe', 'us', 'om', 'ps', 'y', 'w', 'c', 'r', 'me', 'er', 'no', 'of', 'el', 'jg', 'pi', 'la', 'jo', 'cw', 'vs', 'is', 'lo', 'oo', 'an', 'id', 'em', 'um', 'de', 'ox', 'if', 'wo', 'd', 'dc', 'hl', 'ho', 'k', 'a', 'lm', 'xi', 're', 'eh', 'mi', 'mf', 'gs', 'ka', 'ki', 'ln', 'am', 'o', 'f', 'n', 'ti', 'ai', 'j', 'it', 'ax', 'ex', 'ut', 'ss', 'hm', 'ay', 'z', 'v', 'en', 'mb', 'b', 'cs', 'be', 'p', 'xu', 'by', 'lx', 'ja', 'li', 'yo', 'hi', 'et', 'es', 'qi', 'kb', 'do', 'ms', 'i', 'nu', 'e', 'od', 'so', 'sh', 'or', 'db', 'ts', 'kc', 'as', 'm', 'l', 'oh', 'go', 'ad', 'ye', 'dg', 'x', 'in', 'ma', 'ow', 'rs', 'ah', 'oi', 'q', 'ya', 'we', 'ks'}  
night.chargers  
  
  
None  
None  
jackson  
  
  
jacks  
{'jackpot', 'jacks'}  
n't  
  
  
not  
{'net', 'not', "an't", 'nut', 'nit'}  
kraus  
  
  
krauts  
{'kraut', 'krauts'}  
jones  
  
  
ones  
{'cones', 'pones', 'hones', 'zones', 'bones', 'sones', 'jokes', 'ones', 'tones', 'nones'}  
n.y.  
  
  
nay  
{'nays', 'nay'}  
ii  
  
  
i  
{'i', 'ti', 'bi', 'ai', 'li', 'it', 'in', 'hi', 'oi', 'is', 'if', 'si', 'xi', 'qi', 'ki', 'pi', 'id', 'mi'}  
andy  
  
  
and  
{'any', 'handy', 'randy', 'sandy', 'dandy', 'candy', 'bandy', 'and'}  
x.reid  
  
  
nereid  
{'nereid'}  
baltimore  
  
  
None  
None  
photo/adrian  
  
  
None  
None  
''  
  
  
i  
{'mu', 'my', 'kw', 'at', 'up', 'si', 'to', 'h', 'pa', 'on', 'u', 'bi', 'ta', 't', 'ne', 'ls', 'uh', "h'm", 'dm', 'ha', 'g', 'fa', 'he', 'hg', 'c', 'pe', 'us', 'om', 'ps', 'y', 'w', 's', 'r', 'me', 'er', 'no', 'of', 'el', 'jg', 'la', 'pi', 'jo', 'cw', 'vs', 'is', 'lo', 'oo', 'an', 'id', 'em', 'um', 'de', 'ox', 'if', 'wo', 'd', 'dc', 'hl', 'ho', 'k', 'a', 'lm', 'xi', 're', 'eh', 'mi', 'mf', 'gs', 'ka', 'ki', 'ln', 'am', 'o', 'f', 'n', 'ti', 'ai', 'it', 'j', 'ax', 'ex', 'ut', 'ss', 'hm', 'ay', 'z', 'v', 'en', 'mb', 'b', 'cs', 'be', 'p', 'xu', 'by', 'lx', 'ja', 'li', 'yo', 'hi', 'et', 'es', 'qi', 'kb', 'do', 'ms', 'i', 'nu', 'e', 'od', 'so', 'sh', 'or', 'db', 'ts', 'kc', 'as', 'm', 'oh', 'l', 'ad', 'go', 'ye', 'dg', 'x', 'in', 'ma', 'ow', 'rs', 'ah', 'oi', 'q', 'ya', 'we', 'ks'}  
ap  
  
  
a  
{'gap', 'pap', 'bap', 'ai', 'dap', 'ax', 'up', 'at', 'lap', 'pa', 'an', 'ay', 'as', 'zap', 'hap', 'tap', 'cap', 'yap', 'asp', 'ape', 'ad', 'alp', 'amp', 'a', 'p', 'ah', 'map', 'apt', 'nap', 'sap', 'rap', 'am'}  
-ing  
  
  
sing  
{'ding', 'sing', 'wing', 'king', 'ting', 'ring', 'ping', 'zing'}  
reid  
  
  
read  
{'raid', 'read', 'redd', 'rein', 'rid', 'reed', 'rend', 'redid', 'red', 'rebid', 'reis'}  
monday  
  
  
money  
{'monad', 'moray', 'monkey', 'monas', 'fondly', 'moody', 'midday', 'modal', 'monads', 'moldy', 'noonday', 'today', 'mayday', 'monody', 'monde', 'money'}  
jim  
  
  
him  
{'jib', 'jig', 'rim', 'jam', 'dim', 'aim', 'nim', 'mim', 'vim', 'him'}  
kansas  
  
  
canvas  
{'ganjas', 'canvas', 'vandas', 'kwanzas', 'anas', 'kanzus', 'kavas', 'annas', 'balsas', 'manias', 'manses', 'paisas', 'kinas', 'pandas', 'mantas', 'pangas', 'salsas', 'kappas', 'tankas', 'kashas', 'anoas', 'cannas', 'kantar', 'kana', 'manas'}  
're  
  
  
are  
{'are', 're', 'ore', 'ire', 'ere'}  
brittany  
  
  
dittany  
{'dittany'}  
patrick  
  
  
trick  
{'pathic', 'prick', 'patriot', 'iatric', 'hayrick', 'trick', 'strick'}  
kelce  
  
  
else  
{'ketch', 'deice', 'delve', 'dele', 'pence', 'kedge', 'else', 'terce', 'keel', 'kerne', 'veloce', 'welch', 'belch', 'kelpie', 'kyle', 'helve', 'fence', 'kelter', 'kale', 'kente', 'kelt', 'deuce', 'kelp', 'keck', 'hence', 'dolce', 'peace', 'belle', 'belie', 'recce', 'kelly'}  
afc  
  
  
aft  
{'aft', 'arc'}  
's  
  
  
is  
{'ts', 'as', 'gs', 'vs', 'is', 'ps', 'es', 'rs', 'cs', 's', 'ls', 'us', 'ks', 'ms', 'ss'}  
'll  
  
  
all  
{'ell', 'all', 'ill'}  
nfl  
  
  
nil  
{'nil'}  
mahomes  
  
  
homes  
{'mahouts', 'manholes', 'radomes', 'homes'}  
photo/frank  
  
  
None  
None  
lamar  
  
  
lama  
{'damar', 'lamer', 'lama', 'lamas', 'lazar'}  
mvp  
  
  
map  
{'map', 'mop'}  
chris  
  
  
chis  
{'chrism', 'chis'}  
travis  
  
  
traves  
{'traves'}  
foxnews.com  
  
  
None  
None  
--  
  
  
i  
{'mu', 'kw', 'my', 'up', 'at', 'si', 'to', 'h', 'pa', 'on', 'u', 'bi', 'ta', 't', 'ne', 'ls', 'uh', 'dm', 'ha', 'g', 'fa', 'he', 'hg', 's', 'c', 'us', 'om', 'ps', 'pe', 'w', 'y', 'r', 'me', 'er', 'no', 'of', 'el', 'jg', 'la', 'pi', 'jo', 'cw', 'vs', 'is', 'lo', 'oo', 'an', 'id', 'em', 'um', 'de', 'ox', 'if', 'wo', 'd', 'dc', 'hl', 'ho', 'k', 'a', 'lm', 'xi', 're', 'eh', 'mi', 'mf', 'gs', 'ka', 'ki', 'ln', 'am', 'o', 'f', 'n', 'ti', 'ai', 'j', 'it', 'ax', 'ex', 'ut', 'ss', 'hm', 'ay', 'z', 'v', 'en', 'mb', 'b', 'cs', 'be', 'p', 'xu', 'by', 'lx', 'ja', 'li', 'yo', 'hi', 'et', 'es', 'qi', 'kb', 'do', 'ms', 'i', 'nu', 'e', 'od', 'so', 'sh', 'or', 'db', 'ts', 'kc', 'as', 'm', 'oh', 'ad', 'l', 'go', 'ye', 'dg', 'x', 'in', 'ma', 'ow', 'rs', 'ah', 'oi', 'q', 'ya', 'we', 'ks'}  
harbaugh  
  
  
None  
None

output = []  
  
# only the tagger and lemmatizer  
for doc in nlp.pipe(DF['clean'].tolist(), disable=["tok2vec", "ner", "parser"]):  
 tokens = textacy.extract.words(doc,  
 filter\_stops = True, # default True, no stopwords  
 filter\_punct = True, # default True, no punctuation  
 filter\_nums = True, # default False, no numbers  
 include\_pos = None, # default None = include all  
 exclude\_pos = None, # default None = exclude none  
 min\_freq = 1) # minimum frequency of words  
 output.append([str(word) for word in tokens]) # close output append   
  
output[0:10]

[['Having', 'trouble'],  
 ['Click'],  
 ['Kansas',  
 'City',  
 'Chiefs',  
 'quarterback',  
 'Patrick',  
 'Mahomes',  
 'delivered',  
 'epic',  
 'pep',  
 'talk',  
 'teammates',  
 'team',  
 'defeated',  
 'Buffalo',  
 'Bills',  
 'advance',  
 'AFC',  
 'championship',  
 'game',  
 'Sunday'],  
 ['Mahomes',  
 'surrounded',  
 'Travis',  
 'Kelce',  
 'Chris',  
 'Jones',  
 'agreed',  
 'coach',  
 'Andy',  
 'Reid',  
 'job'],  
 ['Baltimore', 'Ravens'],  
 ['Hey', 'asked', 'got', 'asked', 'Mahomes', 'said'],  
 ['Andy', 'Reid', 'said'],  
 ['s---', 'ai'],  
 ['come', 'week', 'ready', 'f', 'ing'],  
 ['CLICK',  
 'SPORTS',  
 'COVERAGE',  
 'FOXNEWS.COM',  
 'Kansas',  
 'City',  
 'Chiefs',  
 'quarterback',  
 'Patrick',  
 'Mahomes',  
 'reacts',  
 'beating',  
 'Buffalo',  
 'Bills',  
 'AFC',  
 'divisional',  
 'playoff',  
 'game',  
 'Sunday',  
 'Jan.',  
 'Orchard',  
 'Park',  
 'N.Y.',  
 'AP',  
 'Photo',  
 'Adrian',  
 'Kraus',  
 'Kelce',  
 'added',  
 'case',  
 'know',  
 'got',  
 'week']]

# all items  
type(output)  
# first list  
type(output[0])  
# first list, first item (this is the issue!)  
type(output[0][0])  
  
Counter(chain.from\_iterable(output))

Counter({'Mahomes': 9,  
 'Chiefs': 6,  
 'game': 6,  
 'Kansas': 5,  
 'City': 5,  
 'quarterback': 5,  
 'Bills': 5,  
 'AFC': 5,  
 'Sunday': 5,  
 'said': 5,  
 'Patrick': 4,  
 'Buffalo': 4,  
 'Ravens': 4,  
 'championship': 3,  
 'got': 3,  
 'divisional': 3,  
 'playoff': 3,  
 'Jan.': 3,  
 'Orchard': 3,  
 'Park': 3,  
 'N.Y.': 3,  
 'AP': 3,  
 'Photo': 3,  
 'going': 3,  
 'Kelce': 2,  
 'Andy': 2,  
 'Reid': 2,  
 'Baltimore': 2,  
 'asked': 2,  
 'week': 2,  
 'Adrian': 2,  
 'Kraus': 2,  
 'added': 2,  
 'time': 2,  
 'win': 2,  
 'toughest': 2,  
 'NFL': 2,  
 'offense': 2,  
 'challenge': 2,  
 'defense': 2,  
 'shut': 2,  
 'quarter': 2,  
 'Having': 1,  
 'trouble': 1,  
 'Click': 1,  
 'delivered': 1,  
 'epic': 1,  
 'pep': 1,  
 'talk': 1,  
 'teammates': 1,  
 'team': 1,  
 'defeated': 1,  
 'advance': 1,  
 'surrounded': 1,  
 'Travis': 1,  
 'Chris': 1,  
 'Jones': 1,  
 'agreed': 1,  
 'coach': 1,  
 'job': 1,  
 'Hey': 1,  
 's---': 1,  
 'ai': 1,  
 'come': 1,  
 'ready': 1,  
 'f': 1,  
 'ing': 1,  
 'CLICK': 1,  
 'SPORTS': 1,  
 'COVERAGE': 1,  
 'FOXNEWS.COM': 1,  
 'reacts': 1,  
 'beating': 1,  
 'case': 1,  
 'know': 1,  
 'VIEW': 1,  
 'MOMENT': 1,  
 'X.Reid': 1,  
 'dancing': 1,  
 'Super': 1,  
 'Bowl': 1,  
 'celebrations': 1,  
 'happen': 1,  
 'matchup': 1,  
 'road': 1,  
 'hosting': 1,  
 'conference': 1,  
 'history': 1,  
 'M&T': 1,  
 'Bank': 1,  
 'Stadium': 1,  
 'night': 1,  
 'CHARGERS': 1,  
 'PURSUIT': 1,  
 'JIM': 1,  
 'HARBAUGH': 1,  
 'ADVANCING': 1,  
 'FINAL': 1,  
 'STAGES': 1,  
 'REPORT': 1,  
 'Brittany': 1,  
 'right': 1,  
 'hugs': 1,  
 'knows': 1,  
 'work': 1,  
 'cut': 1,  
 'boasts': 1,  
 'defenses': 1,  
 'likely': 1,  
 'MVP': 1,  
 'Lamar': 1,  
 'Jackson': 1,  
 'weakness': 1,  
 'star': 1,  
 'best': 1,  
 'effort': 1,  
 'Defense': 1,  
 'special': 1,  
 'teams': 1,  
 'great': 1,  
 'stadium': 1,  
 'rocking': 1,  
 'excited': 1,  
 'Monday': 1,  
 'praised': 1,  
 'coming': 1,  
 'clutch': 1,  
 'quarters': 1,  
 'offensively': 1,  
 'moving': 1,  
 'ball': 1,  
 'field': 1,  
 'went': 1,  
 'told': 1,  
 "Y'all": 1,  
 'football': 1,  
 'rolls': 1,  
 'pocket': 1,  
 'Frank': 1,  
 'Franklin': 1,  
 'II': 1})

# libraries  
import PyPDF2  
import pandas as pd  
import nltk  
# nltk.download("punkt")  
import re  
  
import spacy  
# only for datalore   
import subprocess  
#%%  
print(subprocess.getoutput("python -m spacy download en\_core\_web\_sm"))  
  
nlp = spacy.load("en\_core\_web\_sm")  
  
import textacy  
import summa  
from summa import keywords  
  
from snorkel.preprocess import preprocessor  
from snorkel.types import DataPoint  
from itertools import combinations  
from snorkel.labeling import labeling\_function  
from snorkel.labeling import PandasLFApplier  
  
import networkx as nx  
from matplotlib import pyplot as plt

Defaulting to user installation because normal site-packages is not writeable  
Collecting en-core-web-sm==3.7.1  
 Downloading https://github.com/explosion/spacy-models/releases/download/en\_core\_web\_sm-3.7.1/en\_core\_web\_sm-3.7.1-py3-none-any.whl (12.8 MB)  
 ---------------------------------------- 0.0/12.8 MB ? eta -:--:--  
 - -------------------------------------- 0.5/12.8 MB 10.5 MB/s eta 0:00:02  
 --- ------------------------------------ 1.0/12.8 MB 11.0 MB/s eta 0:00:02  
 ---- ----------------------------------- 1.5/12.8 MB 10.8 MB/s eta 0:00:02  
 ------ --------------------------------- 2.1/12.8 MB 11.1 MB/s eta 0:00:01  
 -------- ------------------------------- 2.6/12.8 MB 11.1 MB/s eta 0:00:01  
 --------- ------------------------------ 3.1/12.8 MB 10.9 MB/s eta 0:00:01  
 ----------- ---------------------------- 3.6/12.8 MB 11.1 MB/s eta 0:00:01  
 ------------ --------------------------- 4.1/12.8 MB 11.0 MB/s eta 0:00:01  
 -------------- ------------------------- 4.7/12.8 MB 11.5 MB/s eta 0:00:01  
 ---------------- ----------------------- 5.2/12.8 MB 11.5 MB/s eta 0:00:01  
 ----------------- ---------------------- 5.7/12.8 MB 11.4 MB/s eta 0:00:01  
 ------------------- -------------------- 6.3/12.8 MB 11.5 MB/s eta 0:00:01  
 --------------------- ------------------ 6.8/12.8 MB 11.5 MB/s eta 0:00:01  
 ---------------------- ----------------- 7.4/12.8 MB 11.5 MB/s eta 0:00:01  
 ------------------------ --------------- 7.9/12.8 MB 11.5 MB/s eta 0:00:01  
 -------------------------- ------------- 8.4/12.8 MB 11.5 MB/s eta 0:00:01  
 ---------------------------- ----------- 9.0/12.8 MB 11.5 MB/s eta 0:00:01  
 ----------------------------- ---------- 9.5/12.8 MB 11.5 MB/s eta 0:00:01  
 ------------------------------ -------- 10.1/12.8 MB 11.5 MB/s eta 0:00:01  
 -------------------------------- ------ 10.6/12.8 MB 11.5 MB/s eta 0:00:01  
 --------------------------------- ----- 11.1/12.8 MB 11.5 MB/s eta 0:00:01  
 ----------------------------------- --- 11.7/12.8 MB 11.5 MB/s eta 0:00:01  
 ------------------------------------ -- 12.1/12.8 MB 11.7 MB/s eta 0:00:01  
 -------------------------------------- 12.6/12.8 MB 11.5 MB/s eta 0:00:01  
 --------------------------------------- 12.8/12.8 MB 11.3 MB/s eta 0:00:00  
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in c:\users\user7\appdata\roaming\python\python39\site-packages (from en-core-web-sm==3.7.1) (3.7.4)  
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.0.12)  
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.0.5)  
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.0.10)  
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.0.8)  
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.0.9)  
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (8.2.3)  
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.1.2)  
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.4.8)  
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.0.10)  
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.3.4)  
Requirement already satisfied: typer<0.10.0,>=0.3.0 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.9.0)  
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (6.4.0)  
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\program files\python39\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (4.66.1)  
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\program files\python39\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.31.0)  
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in c:\program files\python39\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.5.2)  
Requirement already satisfied: jinja2 in c:\program files\python39\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.1.2)  
Requirement already satisfied: setuptools in c:\program files\python39\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (58.1.0)  
Requirement already satisfied: packaging>=20.0 in c:\program files\python39\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (23.2)  
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in c:\users\user7\appdata\roaming\python\python39\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.3.0)  
Requirement already satisfied: numpy>=1.19.0 in c:\program files\python39\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.26.2)  
Requirement already satisfied: annotated-types>=0.4.0 in c:\program files\python39\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.6.0)  
Requirement already satisfied: pydantic-core==2.14.5 in c:\program files\python39\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.14.5)  
Requirement already satisfied: typing-extensions>=4.6.1 in c:\program files\python39\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (4.9.0)  
Requirement already satisfied: charset-normalizer<4,>=2 in c:\program files\python39\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.3.2)  
Requirement already satisfied: idna<4,>=2.5 in c:\program files\python39\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.6)  
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\program files\python39\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.1.0)  
Requirement already satisfied: certifi>=2017.4.17 in c:\program files\python39\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2023.11.17)  
Requirement already satisfied: blis<0.8.0,>=0.7.8 in c:\users\user7\appdata\roaming\python\python39\site-packages (from thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.7.11)  
Requirement already satisfied: confection<1.0.0,>=0.0.1 in c:\users\user7\appdata\roaming\python\python39\site-packages (from thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.1.4)  
Requirement already satisfied: colorama in c:\program files\python39\lib\site-packages (from tqdm<5.0.0,>=4.38.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.4.6)  
Requirement already satisfied: click<9.0.0,>=7.1.1 in c:\program files\python39\lib\site-packages (from typer<0.10.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (8.1.7)  
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in c:\users\user7\appdata\roaming\python\python39\site-packages (from weasel<0.4.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.16.0)  
Requirement already satisfied: MarkupSafe>=2.0 in c:\program files\python39\lib\site-packages (from jinja2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.1.3)  
вњ” Download and installation successful  
You can now load the package via spacy.load('en\_core\_web\_sm')

# creating a pdf file object  
pdfFileObj = open('The\_Shadow\_Over\_Innsmouth.pdf', 'rb')  
   
# creating a pdf reader object  
pdfReader = PyPDF2.PdfReader(pdfFileObj)  
  
# how many pages  
len(pdfReader.pages)  
  
# creating a page object  
pageObj = pdfReader.pages  
   
# extracting text from page  
# loop here to get it all   
text = []  
for page in pageObj:  
 text.append(page.extract\_text())  
  
# closing the pdf file object  
pdfFileObj.close()

---------------------------------------------------------------------------  
FileNotFoundError Traceback (most recent call last)  
Cell In[33], line 2  
 1 # creating a pdf file object  
----> 2 pdfFileObj = open('The\_Shadow\_Over\_Innsmouth.pdf', 'rb')  
 4 # creating a pdf reader object  
 5 pdfReader = PyPDF2.PdfReader(pdfFileObj)  
  
File ~\AppData\Roaming\Python\Python39\site-packages\IPython\core\interactiveshell.py:310, in \_modified\_open(file, \*args, \*\*kwargs)  
 303 if file in {0, 1, 2}:  
 304 raise ValueError(  
 305 f"IPython won't let you open fd={file} by default "  
 306 "as it is likely to crash IPython. If you know what you are doing, "  
 307 "you can use builtins' open."  
 308 )  
--> 310 return io\_open(file, \*args, \*\*kwargs)  
  
FileNotFoundError: [Errno 2] No such file or directory: 'The\_Shadow\_Over\_Innsmouth.pdf'

# create a place to save the text  
saved\_words = []  
  
# loop over each word  
for word in nltk.word\_tokenize(book):  
 # if the word starts with a number and ends with a letter  
 if (re.search(r'^[0-9].\*[a-zA-Z]$', word) != "None"):   
 # take out the numbers and save into our text  
 saved\_words.append(re.sub(r'[0-9]', '', word))  
 # if not then save just the word   
 else:  
 saved\_words.append(word)  
book = ' '.join(saved\_words)  
DF = pd.DataFrame(  
 nltk.sent\_tokenize(book),  
 columns = ["sentences"]  
)  
  
DF.head()  
  
# for IE, we want sentence and/or paragraph level structure

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[13], line 5  
 2 saved\_words = []  
 4 # loop over each word  
----> 5 for word in nltk.word\_tokenize(book):  
 6 # if the word starts with a number and ends with a letter  
 7 if (re.search(r'^[0-9].\*[a-zA-Z]$', word) != "None"):   
 8 # take out the numbers and save into our text  
 9 saved\_words.append(re.sub(r'[0-9]', '', word))  
  
NameError: name 'nltk' is not defined

# easier to loop over the big text file than loop over words AND rows in pandas   
spacy\_pos\_tagged = [(str(word), word.tag\_, word.pos\_) for word in nlp(book)]  
# each row represents one token   
DF\_POS = pd.DataFrame(  
 spacy\_pos\_tagged,  
 columns = ["token", "specific\_tag", "upos"]  
)

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[14], line 2  
 1 # easier to loop over the big text file than loop over words AND rows in pandas   
----> 2 spacy\_pos\_tagged = [(str(word), word.tag\_, word.pos\_) for word in nlp(book)]  
 3 # each row represents one token   
 4 DF\_POS = pd.DataFrame(  
 5 spacy\_pos\_tagged,  
 6 columns = ["token", "specific\_tag", "upos"]  
 7 )  
  
NameError: name 'nlp' is not defined

DF\_POS['upos'].value\_counts()

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[15], line 1  
----> 1 DF\_POS['upos'].value\_counts()  
  
NameError: name 'DF\_POS' is not defined

DF\_POS2 = pd.crosstab(DF\_POS['token'], DF\_POS['upos'])  
# convert to true false to add up how many times not zero   
DF\_POS2['total'] = DF\_POS2.astype(bool).sum(axis=1)  
#print out the rows that aren't 1   
DF\_POS2[DF\_POS2['total'] > 1]

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[16], line 1  
----> 1 DF\_POS2 = pd.crosstab(DF\_POS['token'], DF\_POS['upos'])  
 2 # convert to true false to add up how many times not zero   
 3 DF\_POS2['total'] = DF\_POS2.astype(bool).sum(axis=1)  
  
NameError: name 'pd' is not defined

# textacy KPE  
# build an english language for textacy pipe  
en = textacy.load\_spacy\_lang("en\_core\_web\_sm", disable=("parser"))  
  
# build a processor for textacy using spacy and process text  
doc = textacy.make\_spacy\_doc(book, lang = en)  
  
# text rank algorithm   
print([kps for kps, weights in textacy.extract.keyterms.textrank(doc, normalize = "lemma", topn = 5)])  
  
terms = set([term for term, weight in textacy.extract.keyterms.textrank(doc)])  
print(textacy.extract.utils.aggregate\_term\_variants(terms))

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[17], line 3  
 1 # textacy KPE  
 2 # build an english language for textacy pipe  
----> 3 en = textacy.load\_spacy\_lang("en\_core\_web\_sm", disable=("parser"))  
 5 # build a processor for textacy using spacy and process text  
 6 doc = textacy.make\_spacy\_doc(book, lang = en)  
  
NameError: name 'textacy' is not defined

#TR\_keywords = keywords.keywords(book, scores = True)  
#print(TR\_keywords[0:10])

# easier to loop over the big text file than loop over words AND rows in pandas   
spacy\_ner\_tagged = [(str(word.text), word.label\_) for word in nlp(book).ents]  
  
# each row represents one token   
DF\_NER = pd.DataFrame(  
 spacy\_ner\_tagged,  
 columns = ["token", "entity"]  
)  
print(DF\_NER['entity'].value\_counts())  
  
DF\_NER2 = pd.crosstab(DF\_NER['token'], DF\_NER['entity'])  
print(DF\_NER2)  
# convert to true false to add up how many times not zero   
DF\_NER2['total'] = DF\_NER2.astype(bool).sum(axis=1)  
#print out the rows that aren't 1   
DF\_NER2[DF\_NER2['total'] > 1]

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[20], line 2  
 1 # easier to loop over the big text file than loop over words AND rows in pandas   
----> 2 spacy\_ner\_tagged = [(str(word.text), word.label\_) for word in nlp(book).ents]  
 4 # each row represents one token   
 5 DF\_NER = pd.DataFrame(  
 6 spacy\_ner\_tagged,  
 7 columns = ["token", "entity"]  
 8 )  
  
NameError: name 'nlp' is not defined

stored\_entities = []  
  
# first get the entities, must be two for relationship matches  
def get\_entities(x):  
 """  
 Grabs the names using spacy's entity labeler  
 """  
 # get all the entities in this row   
 processed = nlp(x)  
 # get the tokens for each sentence  
 tokens = [word.text for word in processed]  
 # get all the entities - notice this is only for persons   
 temp = [(str(ent), ent.label\_) for ent in processed.ents if ent.label\_ != ""]  
 # only move on if this row has at least two  
 if len(temp) > 1:   
 # finds all the combinations of pairs   
 temp2 = list(combinations(temp, 2))  
 # for each pair combination   
 for (person1, person2) in temp2:  
 # find the names in the person 1  
 person1\_words = [word.text for word in nlp(person1[0])]  
 # find the token numbers for person 1  
 person1\_ids = [i for i, val in enumerate(tokens) if val in person1\_words]  
 # output in (start, stop) token tuple format   
 if len(person1\_words) > 1:  
 person1\_ids2 = tuple(idx for idx in person1\_ids[0:2])  
 else:  
 id\_1 = [idx for idx in person1\_ids]  
 person1\_ids2 = (id\_1[0], id\_1[0])  
   
 # do the same thing with person 2  
 person2\_words = [word.text for word in nlp(person2[0])]  
 person2\_ids = [i for i, val in enumerate(tokens) if val in person2\_words[0:2]]  
 if len(person2\_words) > 1:  
 person2\_ids2 = tuple(idx for idx in person2\_ids)  
 else:  
 id\_2 = [idx for idx in person2\_ids[0:2]]  
 person2\_ids2 = (id\_2[0], id\_2[0])   
   
 # store all this in a list   
 stored\_entities.append(  
 [x, # original text  
 tokens, # tokens  
 person1[0], # person 1 name  
 person2[0], # person 2 name  
 person1\_ids2, # person 1 id token tuple   
 person2\_ids2 # person 2 id token tuple  
 ])  
  
DF['sentences'].apply(get\_entities)  
  
# create dataframe in snorkel structure   
DF\_dev = pd.DataFrame(stored\_entities, columns = ["sentence", "tokens", "person1", "person2", "person1\_word\_idx", "person2\_word\_idx"])

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[21], line 50  
 40 # store all this in a list   
 41 stored\_entities.append(  
 42 [x, # original text  
 43 tokens, # tokens  
 (...)  
 47 person2\_ids2 # person 2 id token tuple  
 48 ])  
---> 50 DF['sentences'].apply(get\_entities)  
 52 # create dataframe in snorkel structure   
 53 DF\_dev = pd.DataFrame(stored\_entities, columns = ["sentence", "tokens", "person1", "person2", "person1\_word\_idx", "person2\_word\_idx"])  
  
NameError: name 'DF' is not defined

# live locate home road roads in at street (locations tied together)  
# family terms for people   
  
# get words between the data points   
@preprocessor()  
def get\_text\_between(cand: DataPoint) -> DataPoint:  
 """  
 Returns the text between the two person mentions in the sentence  
 """  
 start = cand.person1\_word\_idx[1] + 1  
 end = cand.person2\_word\_idx[0]  
 cand.between\_tokens = cand.tokens[start:end]  
 return cand  
  
# get words next to the data points  
@preprocessor()  
def get\_left\_tokens(cand: DataPoint) -> DataPoint:  
 """  
 Returns tokens in the length 3 window to the left of the person mentions  
 """  
 # TODO: need to pass window as input params  
 window = 5  
  
 end = cand.person1\_word\_idx[0]  
 cand.person1\_left\_tokens = cand.tokens[0:end][-1 - window : -1]  
  
 end = cand.person2\_word\_idx[0]  
 cand.person2\_left\_tokens = cand.tokens[0:end][-1 - window : -1]  
 return cand

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[22], line 5  
 1 # live locate home road roads in at street (locations tied together)  
 2 # family terms for people   
 3   
 4 # get words between the data points   
----> 5 @preprocessor()  
 6 def get\_text\_between(cand: DataPoint) -> DataPoint:  
 7 """  
 8 Returns the text between the two person mentions in the sentence  
 9 """  
 10 start = cand.person1\_word\_idx[1] + 1  
  
NameError: name 'preprocessor' is not defined

# live locate home road roads in at street (locations tied together)  
# family terms for people   
  
found\_location = 1  
found\_family = -1  
ABSTAIN = 0  
  
location = {"live", "living", "locate", "located", "home", "road", "roads", "street", "streets", "in", "at", "of"}  
  
@labeling\_function(resources=dict(location=location), pre=[get\_text\_between])  
def between\_location(x, location):  
 return found\_location if len(location.intersection(set(x.between\_tokens))) > 0 else ABSTAIN  
  
@labeling\_function(resources=dict(location=location), pre=[get\_left\_tokens])  
def left\_location(x, location):  
 if len(set(location).intersection(set(x.person1\_left\_tokens))) > 0:  
 return found\_location  
 elif len(set(location).intersection(set(x.person2\_left\_tokens))) > 0:  
 return found\_location  
 else:  
 return ABSTAIN  
  
family = {"spouse", "wife", "husband", "ex-wife", "ex-husband", "marry",   
 "married", "father", "mother", "sister", "brother", "son", "daughter",   
 "grandfather", "grandmother", "uncle", "aunt", "cousin",   
 "boyfriend", "girlfriend"}  
  
@labeling\_function(resources=dict(family=family), pre=[get\_text\_between])  
def between\_family(x, family):  
 return found\_family if len(family.intersection(set(x.between\_tokens))) > 0 else ABSTAIN  
  
@labeling\_function(resources=dict(family=family), pre=[get\_left\_tokens])  
def left\_family(x, family):  
 if len(set(family).intersection(set(x.person1\_left\_tokens))) > 0:  
 return found\_family  
 elif len(set(family).intersection(set(x.person2\_left\_tokens))) > 0:  
 return found\_family  
 else:  
 return ABSTAIN  
  
# create a list of functions to run   
lfs = [  
 between\_location,  
 left\_location,  
 between\_family,  
 left\_family  
]  
# build the applier function   
applier = PandasLFApplier(lfs)  
# run it on the dataset   
L\_dev = applier.apply(DF\_dev)  
L\_dev  
DF\_combined = pd.concat([DF\_dev, pd.DataFrame(L\_dev, columns = ["location1", "location2", "family1", "family2"])], axis = 1)  
DF\_combined  
  
DF\_combined['location\_yes'] = DF\_combined['location1'] + DF\_combined["location2"]  
DF\_combined['family\_yes'] = DF\_combined['family1'] + DF\_combined["family2"]  
  
print(DF\_combined['location\_yes'].value\_counts())  
print(DF\_combined['family\_yes'].value\_counts())

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[23], line 10  
 6 ABSTAIN = 0  
 8 location = {"live", "living", "locate", "located", "home", "road", "roads", "street", "streets", "in", "at", "of"}  
---> 10 @labeling\_function(resources=dict(location=location), pre=[get\_text\_between])  
 11 def between\_location(x, location):  
 12 return found\_location if len(location.intersection(set(x.between\_tokens))) > 0 else ABSTAIN  
 14 @labeling\_function(resources=dict(location=location), pre=[get\_left\_tokens])  
 15 def left\_location(x, location):  
  
NameError: name 'labeling\_function' is not defined

# locations only  
DF\_loc = DF\_combined[DF\_combined['location\_yes'] > 0]  
DF\_loc = DF\_loc[['person1', 'person2']].reset\_index(drop = True)  
  
cooc\_loc = DF\_loc.groupby(by=["person1", "person2"], as\_index=False).size()  
  
# family only  
DF\_fam = DF\_combined[DF\_combined['family\_yes'] > 0]  
DF\_fam = DF\_fam[['person1', 'person2']].reset\_index(drop = True)  
  
cooc\_fam = DF\_fam.groupby(by=["person1", "person2"], as\_index=False).size()  
  
# take out issues where entity 1 == entity 2  
cooc\_loc = cooc\_loc[cooc\_loc['person1'] != cooc\_loc['person2']]  
cooc\_fam = cooc\_fam[cooc\_fam['person1'] != cooc\_fam['person2']]  
  
print(cooc\_loc.head())  
print(cooc\_fam.head())

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[24], line 2  
 1 # locations only  
----> 2 DF\_loc = DF\_combined[DF\_combined['location\_yes'] > 0]  
 3 DF\_loc = DF\_loc[['person1', 'person2']].reset\_index(drop = True)  
 5 cooc\_loc = DF\_loc.groupby(by=["person1", "person2"], as\_index=False).size()  
  
NameError: name 'DF\_combined' is not defined

# start by plotting the whole thing for location   
cooc\_loc\_small = cooc\_loc[cooc\_loc['size']>1]  
graph = nx.from\_pandas\_edgelist(  
 cooc\_loc\_small[['person1', 'person2', 'size']] \  
 .rename(columns={'size': 'weight'}),  
 source='person1', target='person2', edge\_attr=True)  
  
pos = nx.kamada\_kawai\_layout(graph, weight='weight')  
  
\_ = plt.figure(figsize=(20, 20))  
nx.draw(graph, pos,   
 node\_size=1000,   
 node\_color='skyblue',  
 alpha=0.8,  
 with\_labels = True)  
plt.title('Graph Visualization', size=15)  
  
for (node1,node2,data) in graph.edges(data=True):  
 width = data['weight']   
 \_ = nx.draw\_networkx\_edges(graph,pos,  
 edgelist=[(node1, node2)],  
 width=width,  
 edge\_color='#505050',  
 alpha=0.5)  
  
plt.show()  
plt.close()  
# start by plotting the whole thing for location   
graph = nx.from\_pandas\_edgelist(  
 cooc\_fam[['person1', 'person2', 'size']] \  
 .rename(columns={'size': 'weight'}),  
 source='person1', target='person2', edge\_attr=True)  
  
pos = nx.kamada\_kawai\_layout(graph, weight='weight')  
  
\_ = plt.figure(figsize=(20, 20))  
nx.draw(graph, pos,   
 node\_size=1000,   
 node\_color='skyblue',  
 alpha=0.8,  
 with\_labels = True)  
plt.title('Graph Visualization', size=15)  
  
for (node1,node2,data) in graph.edges(data=True):  
 width = data['weight']   
 \_ = nx.draw\_networkx\_edges(graph,pos,  
 edgelist=[(node1, node2)],  
 width=width,  
 edge\_color='#505050',  
 alpha=0.5)  
  
plt.show()  
plt.close()

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[25], line 2  
 1 # start by plotting the whole thing for location   
----> 2 cooc\_loc\_small = cooc\_loc[cooc\_loc['size']>1]  
 3 graph = nx.from\_pandas\_edgelist(  
 4 cooc\_loc\_small[['person1', 'person2', 'size']] \  
 5 .rename(columns={'size': 'weight'}),  
 6 source='person1', target='person2', edge\_attr=True)  
 8 pos = nx.kamada\_kawai\_layout(graph, weight='weight')  
  
NameError: name 'cooc\_loc' is not defined

import pysrt  
import pandas as pd  
import re  
from sentence\_transformers import SentenceTransformer  
# install faiss-cpu  
import faiss  
import time  
subs = pysrt.open("bodies.srt")  
  
DF = pd.DataFrame([  
 {  
 "Text": sub.text  
} for sub in subs])  
  
DF  
def remove\_noise(text):  
 text = re.sub("<.\*>", " ", text)  
 text = re.sub("{.\*}", " ", text)  
 text = re.sub("\[.\*\]", " ", text)  
 text = text.strip()  
 return text  
  
DF['clean'] = DF['Text'].apply(remove\_noise)  
  
DF = DF[DF['clean'] != ""]  
  
DF

---------------------------------------------------------------------------  
ModuleNotFoundError Traceback (most recent call last)  
Cell In[26], line 1  
----> 1 import pysrt  
 2 import pandas as pd  
 3 import re  
  
ModuleNotFoundError: No module named 'pysrt'

# this is creating the embeddings   
model = SentenceTransformer('msmarco-MiniLM-L-12-v3')  
bodies\_text\_embds = model.encode(DF['clean'].to\_list())  
# Create an index using FAISS  
index = faiss.IndexFlatL2(bodies\_text\_embds.shape[1])  
index.add(bodies\_text\_embds)  
faiss.write\_index(index, 'index\_bodies')  
  
bodies\_text\_embds  
# define a search   
def search(query, k):  
   
 t=time.time()  
 query\_vector = model.encode([query])  
 top\_k = index.search(query\_vector, k)  
 print('totaltime: {}'.format(time.time()-t))  
 return [DF['clean'].to\_list()[\_id] for \_id in top\_k[1].tolist()[0]]

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
Cell In[27], line 2  
 1 # this is creating the embeddings   
----> 2 model = SentenceTransformer('msmarco-MiniLM-L-12-v3')  
 3 bodies\_text\_embds = model.encode(DF['clean'].to\_list())  
 4 # Create an index using FAISS  
  
NameError: name 'SentenceTransformer' is not defined

search("cop", 10)  
search("gun", 10)  
search("car", 10)