# Information Extraction

## Libraries

# 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 -:--:--  
 - -------------------------------------- 0.4/12.8 MB 7.4 MB/s eta 0:00:02  
 -- ------------------------------------- 0.9/12.8 MB 9.2 MB/s eta 0:00:02  
 ---- ----------------------------------- 1.4/12.8 MB 9.7 MB/s eta 0:00:02  
 ----- ---------------------------------- 1.9/12.8 MB 10.1 MB/s eta 0:00:02  
 ------- -------------------------------- 2.4/12.8 MB 10.4 MB/s eta 0:00:01  
 --------- ------------------------------ 2.9/12.8 MB 11.0 MB/s eta 0:00:01  
 ---------- ----------------------------- 3.5/12.8 MB 11.1 MB/s eta 0:00:01  
 ------------ --------------------------- 4.0/12.8 MB 11.1 MB/s eta 0:00:01  
 -------------- ------------------------- 4.5/12.8 MB 11.1 MB/s eta 0:00:01  
 --------------- ------------------------ 5.0/12.8 MB 11.1 MB/s eta 0:00:01  
 ----------------- ---------------------- 5.6/12.8 MB 11.1 MB/s eta 0:00:01  
 ------------------- -------------------- 6.1/12.8 MB 11.1 MB/s eta 0:00:01  
 -------------------- ------------------- 6.6/12.8 MB 11.1 MB/s eta 0:00:01  
 ---------------------- ----------------- 7.1/12.8 MB 11.1 MB/s eta 0:00:01  
 ----------------------- ---------------- 7.6/12.8 MB 11.1 MB/s eta 0:00:01  
 ------------------------- -------------- 8.2/12.8 MB 11.3 MB/s eta 0:00:01  
 --------------------------- ------------ 8.7/12.8 MB 11.3 MB/s eta 0:00:01  
 ---------------------------- ----------- 9.2/12.8 MB 11.3 MB/s eta 0:00:01  
 ------------------------------ --------- 9.7/12.8 MB 11.3 MB/s eta 0:00:01  
 ------------------------------- ------- 10.2/12.8 MB 11.3 MB/s eta 0:00:01  
 -------------------------------- ------ 10.8/12.8 MB 11.5 MB/s eta 0:00:01  
 ---------------------------------- ---- 11.3/12.8 MB 11.5 MB/s eta 0:00:01  
 ------------------------------------ -- 11.8/12.8 MB 11.5 MB/s eta 0:00:01  
 ------------------------------------- - 12.4/12.8 MB 11.5 MB/s eta 0:00:01  
 -------------------------------------- 12.8/12.8 MB 11.5 MB/s eta 0:00:01  
 --------------------------------------- 12.8/12.8 MB 11.1 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')

## Import Text

# creating a pdf file object  
# pdfFileObj = open('The\_Shadow\_Over\_Innsmouth.pdf', 'rb')  
pdfFileObj = open('2024.pdf', 'rb')  
   
# creating a pdf reader object  
pdfReader = PyPDF2.PdfReader(pdfFileObj)  
  
# how many pages  
len(pdfReader.pages)  
  
print(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()

18

## Convert to Sentences and Pandas

* ^ means start with
* [0-9] means any of these digits
* [a-zA-Z] means any alpha latin character lower or upper case
* $ ends with
* . mean any character
* o means zero or more of the previous character (so .\* means zero or more of any character)

# create a place to save the text  
saved\_words = []  
  
# Joining the extracted text pages into a single string  
book = ' '.join(text)  
  
# 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

sentences  
0 Data science interview questions and answers f...  
1 We encourage you to go through the curated lis...  
2 Apply as data scientists I 'm hiring developer...  
3 As a vast field , with plenty of demand , both...  
4 Hence , to prepare both parties , we have cura...

## Part of Speech Tagging

* Tag your data with spacy’s part of speech tagger.
* Convert this data into a Pandas DataFrame.

# 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"]  
)

* Use the dataframe to calculate the most common parts of speech.

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

upos  
NOUN 1223  
PUNCT 526  
VERB 514  
DET 466  
ADP 428  
ADJ 340  
AUX 301  
PROPN 229  
PRON 168  
CCONJ 137  
PART 92  
ADV 72  
SCONJ 68  
NUM 55  
SPACE 45  
X 16  
SYM 9  
INTJ 8  
Name: count, dtype: int64

* Use the dataframe to calculate if words are considered more than one part of speech (crosstabs or groupby).

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]

upos ADJ ADP ADV AUX CCONJ DET INTJ NOUN NUM PART PRON PROPN \  
token   
+ 0 0 0 0 0 0 0 0 1 0 0 1   
- 0 0 0 0 0 0 0 0 0 0 0 1   
-based 1 0 0 0 0 0 0 0 0 0 0 0   
-means 0 0 0 0 0 0 0 2 0 0 0 3   
-square 1 0 0 0 0 0 0 1 0 0 0 0   
... ... ... ... ... ... ... ... ... ... ... ... ...   
to 0 25 0 0 0 0 0 0 0 79 0 0   
training 0 0 0 0 0 0 0 3 0 0 0 0   
use 0 0 0 0 0 0 0 1 0 0 0 0   
which 0 0 0 0 0 1 0 0 0 0 5 0   
• 0 1 0 0 0 0 0 0 19 0 2 0   
  
upos PUNCT SCONJ SPACE SYM VERB X total   
token   
+ 0 0 0 0 0 0 2   
- 6 0 0 0 0 0 2   
-based 0 0 0 0 1 0 2   
-means 0 0 0 0 0 0 2   
-square 0 0 0 0 0 0 2   
... ... ... ... ... ... .. ...   
to 0 0 0 0 0 0 2   
training 0 0 0 0 4 0 2   
use 0 0 0 0 10 0 2   
which 0 0 0 0 0 0 2   
• 0 0 0 0 2 5 5   
  
[97 rows x 19 columns]

* What is the most common part of speech? ANSWER THIS IN YOUR TEXT
* Do you see words that are multiple parts of speech? ANSWER THIS IN YOUR TEXT

## KPE

* Use textacy to find the key phrases in your text.
* o in the r window for r people
* o library(reticulate)
* o py\_install("networkx < 3.0", pip = T)

# 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))

['content Basic datum science interview question', 'datum science technical interview question', 'intermediate datum science interview question', 'advanced datum science interview question', 'datum science job']  
[{'content Basic datum science interview question'}, {'intermediate datum science interview question'}, {'datum science technical interview question'}, {'advanced datum science interview question'}, {'exploratory datum analysis'}, {'datum science application'}, {'datum science life cycle'}, {'different datum type'}, {'datum science job'}, {'sample datum'}]

• Use summa to find the key phrases in your text.

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

[('data science interview questions', 0.22221301412918812), ('sampling', 0.14684813577795505), ('sample', 0.14684813577795505), ('samplings', 0.14684813577795505), ('samples', 0.14684813577795505), ('variables', 0.14487282640726087), ('variable', 0.14487282640726087), ('values', 0.1339291369649484), ('value', 0.1339291369649484), ('methods', 0.12859857435603242)]

* What differences do you see in their outputs? COMMENT ON HOW SLOW!
* Using textacy utilities, combine like key phrases. SEE ABOVE
* Do the outputs make sense given your text? ANSWER THIS QUESTION

## NER + Snorkel

* Use spacy to extract named entities.
* Create a summary of your named entities.

# 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]

entity  
ORG 38  
CARDINAL 30  
PERSON 14  
PRODUCT 10  
GPE 9  
NORP 3  
WORK\_OF\_ART 3  
LOC 2  
ORDINAL 2  
DATE 2  
EVENT 1  
Name: count, dtype: int64  
entity CARDINAL DATE EVENT \  
token   
AI 0 0 0   
ANOVA 0 0 0   
Algorithm 0 0 0   
Apply Now INTERMEDIATE DATA SCIENCE INTERVIEW Q... 0 0 0   
Apply Now WRAPPING UP 0 0 1   
... ... ... ...   
• Non 0 0 0   
• Recursive 0 0 0   
• Repeat 0 0 0   
• Select 0 0 0   
• Wrapper 0 0 0   
  
entity GPE LOC NORP ORDINAL \  
token   
AI 0 0 0 0   
ANOVA 0 0 0 0   
Algorithm 0 0 0 0   
Apply Now INTERMEDIATE DATA SCIENCE INTERVIEW Q... 0 0 0 0   
Apply Now WRAPPING UP 0 0 0 0   
... ... ... ... ...   
• Non 0 0 0 0   
• Recursive 0 0 0 0   
• Repeat 0 0 0 0   
• Select 0 0 0 0   
• Wrapper 0 0 0 0   
  
entity ORG PERSON PRODUCT \  
token   
AI 1 0 0   
ANOVA 2 0 0   
Algorithm 0 0 1   
Apply Now INTERMEDIATE DATA SCIENCE INTERVIEW Q... 0 0 1   
Apply Now WRAPPING UP 0 0 0   
... ... ... ...   
• Non 1 0 0   
• Recursive 0 0 1   
• Repeat 1 0 0   
• Select 0 0 1   
• Wrapper 0 0 1   
  
entity WORK\_OF\_ART   
token   
AI 0   
ANOVA 0   
Algorithm 0   
Apply Now INTERMEDIATE DATA SCIENCE INTERVIEW Q... 0   
Apply Now WRAPPING UP 0   
... ...   
• Non 0   
• Recursive 0   
• Repeat 0   
• Select 0   
• Wrapper 0   
  
[67 rows x 11 columns]

entity CARDINAL DATE EVENT GPE LOC NORP ORDINAL ORG PERSON \  
token   
Data Science 0 0 0 0 0 0 0 1 1   
  
entity PRODUCT WORK\_OF\_ART total   
token   
Data Science 0 0 2

* Apply Snorkel to your data to show any relationship between names.

## get the data into a good format

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"])

## figure out where to look (between and to the left)

# 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

## figure out what to look for

# 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)

100%|██████████| 49/49 [00:00<00:00, 669.47it/s]

L\_dev

array([[0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [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, 1, 0, 0],  
 [0, 0, 0, 0],  
 [1, 0, 0, 0],  
 [0, 0, 0, 0],  
 [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, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [1, 1, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [1, 1, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [1, 1, 0, 0],  
 [0, 0, 0, 0],  
 [1, 1, 0, 0],  
 [1, 1, 0, 0],  
 [0, 0, 0, 0],  
 [1, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [1, 0, 0, 0],  
 [1, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [0, 0, 0, 0],  
 [1, 0, 0, 0],  
 [1, 0, 0, 0]])

DF\_combined = pd.concat([DF\_dev, pd.DataFrame(L\_dev, columns = ["location1", "location2", "family1", "family2"])], axis = 1)  
DF\_combined

sentence \  
0 The two main sampling techniques used as per s...   
1 Structured , semi -structured , and unstructur...   
2 Volume , Velocity , Variety , Veracity , and V...   
3 NLP stands for Natural Language Processing , w...   
4 Correlation and covariance are two measures of...   
5 Correlation is a measure of how two variables ...   
6 The Hamming distance and the Levenshtein dista...   
7 The Hamming distance and the Levenshtein dista...   
8 The Hamming distance and the Levenshtein dista...   
9 The number of bits that differ between two str...   
10 The number of edit operations ( insert , delet...   
11 Nunique function is an aggregation function in...   
12 .Is SVM a classification Algorithm ?   
13 Univariate analysis has one variable , whereas...   
14 A Pandas Index is a mutable , ordered set that...   
15 .List some benefits of using TensorFlow There ...   
16 Wrapper method includes : • Forward selection ...   
17 Filter method includes : • Chi-square • ANOVA ...   
18 Below are the steps for building a decision tr...   
19 Below are the steps for building a decision tr...   
20 Below are the steps for building a decision tr...   
21 Below are the steps for building a decision tr...   
22 Below are the steps for building a decision tr...   
23 Below are the steps for building a decision tr...   
24 Below are the steps for building a decision tr...   
25 Below are the steps for building a decision tr...   
26 Below are the steps for building a decision tr...   
27 Below are the steps for building a decision tr...   
28 Below are the steps for building a decision tr...   
29 Below are the steps for building a decision tr...   
30 Below are the steps for building a decision tr...   
31 Below are the steps for building a decision tr...   
32 Below are the steps for building a decision tr...   
33 .Give some drawbacks of linear regression mode...   
34 Bivariate analysis is the study of two variabl...   
35 X [ : , np.newaxis ] + Y .Solve the below code...   
36 X [ : , np.newaxis ] + Y .Solve the below code...   
37 X [ : , np.newaxis ] + Y .Solve the below code...   
38 X [ : , np.newaxis ] + Y .Solve the below code...   
39 X [ : , np.newaxis ] + Y .Solve the below code...   
40 X [ : , np.newaxis ] + Y .Solve the below code...   
41 X [ : , np.newaxis ] + Y .Solve the below code...   
42 X [ : , np.newaxis ] + Y .Solve the below code...   
43 X [ : , np.newaxis ] + Y .Solve the below code...   
44 X [ : , np.newaxis ] + Y .Solve the below code...   
45 The probability is / .Using the Euclidean dist...   
46 For : ( X , Y ) = ( , ) ( X , Y ) = ( , ) ...   
47 For : ( X , Y ) = ( , ) ( X , Y ) = ( , ) ...   
48 For : ( X , Y ) = ( , ) ( X , Y ) = ( , ) ...   
  
 tokens \  
0 [The, two, main, sampling, techniques, used, a...   
1 [Structured, ,, semi, -structured, ,, and, uns...   
2 [Volume, ,, Velocity, ,, Variety, ,, Veracity,...   
3 [NLP, stands, for, Natural, Language, Processi...   
4 [Correlation, and, covariance, are, two, measu...   
5 [Correlation, is, a, measure, of, how, two, va...   
6 [The, Hamming, distance, and, the, Levenshtein...   
7 [The, Hamming, distance, and, the, Levenshtein...   
8 [The, Hamming, distance, and, the, Levenshtein...   
9 [The, number, of, bits, that, differ, between,...   
10 [The, number, of, edit, operations, (, insert,...   
11 [Nunique, function, is, an, aggregation, funct...   
12 [.Is, SVM, a, classification, Algorithm, ?]   
13 [Univariate, analysis, has, one, variable, ,, ...   
14 [A, Pandas, Index, is, a, mutable, ,, ordered,...   
15 [.List, some, benefits, of, using, TensorFlow,...   
16 [Wrapper, method, includes, :, •, Forward, sel...   
17 [Filter, method, includes, :, •, Chi, -, squar...   
18 [Below, are, the, steps, for, building, a, dec...   
19 [Below, are, the, steps, for, building, a, dec...   
20 [Below, are, the, steps, for, building, a, dec...   
21 [Below, are, the, steps, for, building, a, dec...   
22 [Below, are, the, steps, for, building, a, dec...   
23 [Below, are, the, steps, for, building, a, dec...   
24 [Below, are, the, steps, for, building, a, dec...   
25 [Below, are, the, steps, for, building, a, dec...   
26 [Below, are, the, steps, for, building, a, dec...   
27 [Below, are, the, steps, for, building, a, dec...   
28 [Below, are, the, steps, for, building, a, dec...   
29 [Below, are, the, steps, for, building, a, dec...   
30 [Below, are, the, steps, for, building, a, dec...   
31 [Below, are, the, steps, for, building, a, dec...   
32 [Below, are, the, steps, for, building, a, dec...   
33 [.Give, some, drawbacks, of, linear, regressio...   
34 [Bivariate, analysis, is, the, study, of, two,...   
35 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
36 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
37 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
38 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
39 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
40 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
41 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
42 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
43 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
44 [X, [, :, ,, np.newaxis, ], +, Y, .Solve, the,...   
45 [The, probability, is, /, .Using, the, Euclide...   
46 [For, :, (, X, ,, Y, ), =, (, , ,, , ), (, X...   
47 [For, :, (, X, ,, Y, ), =, (, , ,, , ), (, X...   
48 [For, :, (, X, ,, Y, ), =, (, , ,, , ), (, X...   
  
 person1 \  
0 two   
1 Structured   
2 Value   
3 NLP   
4 two   
5 two   
6 Levenshtein   
7 Levenshtein   
8 two   
9 two   
10 one   
11 Nunique   
12 SVM   
13 one   
14 Pandas   
15 TensorFlow   
16 One   
17 • Chi-square   
18 • Calculate   
19 • Calculate   
20 • Calculate   
21 • Calculate   
22 • Calculate   
23 • Calculate   
24 • Calculate   
25 • Calculate   
26 • Calculate   
27 • Calculate   
28 • Calculate   
29 • Calculate   
30 • Select   
31 • Select   
32 • Repeat   
33 linear   
34 Bivariate   
35 Biotech   
36 Biotech   
37 Biotech   
38 Biotech   
39 years   
40 years   
41 years   
42 SELECT Name FROM StudentData WHERE Department ...   
43 SELECT Name FROM StudentData WHERE Department ...   
44 three   
45 Euclidean   
46 ROC   
47 ROC   
48 Build Looking   
  
 person2 person1\_word\_idx \  
0 • Non (1, 1)   
1 three (0, 0)   
2 five (9, 9)   
3 Natural Language Processing (0, 0)   
4 two (4, 4)   
5 two (6, 6)   
6 two (5, 5)   
7 two (5, 5)   
8 two (8, 8)   
9 Hamming (7, 7)   
10 Levenshtein (17, 17)   
11 PostgreSQL (0, 0)   
12 Algorithm (1, 1)   
13 two (3, 3)   
14 Pandas DataFrame (1, 1)   
15 TensorFlow (5, 5)   
16 • Backward (8, 8)   
17 • Filter (4, 5)   
18 • Calculate (18, 19)   
19 • Calculate (18, 19)   
20 • Select (18, 19)   
21 • Repeat (18, 19)   
22 INR (18, 19)   
23 • Calculate (18, 19)   
24 • Select (18, 19)   
25 • Repeat (18, 19)   
26 INR (18, 19)   
27 • Select (18, 19)   
28 • Repeat (18, 19)   
29 INR (18, 19)   
30 • Repeat (18, 29)   
31 INR (18, 29)   
32 INR (18, 29)   
33 linear (4, 4)   
34 two (0, 0)   
35 years (71, 71)   
36 SELECT Name FROM StudentData WHERE Department ... (71, 71)   
37 three (71, 71)   
38 two (71, 71)   
39 SELECT Name FROM StudentData WHERE Department ... (76, 76)   
40 three (76, 76)   
41 two (76, 76)   
42 three (12, 40)   
43 two (12, 40)   
44 two (100, 100)   
45 P (6, 6)   
46 Build Looking (29, 29)   
47 US (29, 29)   
48 US (33, 34)   
  
 person2\_word\_idx location1 location2 family1 family2   
0 (12, 26, 27) 0 0 0 0   
1 (10, 10) 0 0 0 0   
2 (12, 12) 0 0 0 0   
3 (3, 4) 0 0 0 0   
4 (4, 4) 0 0 0 0   
5 (6, 6) 0 1 0 0   
6 (8, 8) 0 0 0 0   
7 (8, 8) 0 0 0 0   
8 (8, 8) 0 0 0 0   
9 (13, 13) 0 1 0 0   
10 (25, 25) 0 0 0 0   
11 (7, 7) 1 0 0 0   
12 (4, 4) 0 0 0 0   
13 (10, 10) 0 0 0 0   
14 (1, 18, 19) 0 0 0 0   
15 (5, 5) 0 1 0 0   
16 (4, 23, 24) 0 0 0 0   
17 (0, 4, 8, 10, 14, 15) 0 0 0 0   
18 (18, 19, 29, 30, 36, 37, 44, 56) 0 0 0 0   
19 (18, 19, 29, 30, 36, 37, 44, 56) 0 0 0 0   
20 (18, 29, 36, 44, 45, 56) 0 0 0 0   
21 (18, 29, 36, 44, 56, 57) 0 0 0 0   
22 (128, 128) 1 1 0 0   
23 (18, 19, 29, 30, 36, 37, 44, 56) 0 0 0 0   
24 (18, 29, 36, 44, 45, 56) 0 0 0 0   
25 (18, 29, 36, 44, 56, 57) 0 0 0 0   
26 (128, 128) 1 1 0 0   
27 (18, 29, 36, 44, 45, 56) 0 0 0 0   
28 (18, 29, 36, 44, 56, 57) 0 0 0 0   
29 (128, 128) 1 1 0 0   
30 (18, 29, 36, 44, 56, 57) 0 0 0 0   
31 (128, 128) 1 1 0 0   
32 (128, 128) 1 1 0 0   
33 (4, 4) 0 0 0 0   
34 (6, 6) 1 0 0 0   
35 (76, 76) 0 0 0 0   
36 (40, 41, 78, 79) 0 0 0 0   
37 (100, 100) 0 0 0 0   
38 (109, 109) 0 0 0 0   
39 (40, 41, 78, 79) 0 0 0 0   
40 (100, 100) 0 0 0 0   
41 (109, 109) 0 0 0 0   
42 (100, 100) 1 0 0 0   
43 (109, 109) 1 0 0 0   
44 (109, 109) 0 0 0 0   
45 (16, 16) 0 0 0 0   
46 (33, 34) 0 0 0 0   
47 (40, 40) 1 0 0 0   
48 (40, 40) 1 0 0 0

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())  
print(DF\_combined.head())  
DF\_combined.to\_csv('family\_check.csv', index=False)

location\_yes  
0 35  
1 9  
2 5  
Name: count, dtype: int64  
family\_yes  
0 49  
Name: count, dtype: int64  
 sentence \  
0 The two main sampling techniques used as per s...   
1 Structured , semi -structured , and unstructur...   
2 Volume , Velocity , Variety , Veracity , and V...   
3 NLP stands for Natural Language Processing , w...   
4 Correlation and covariance are two measures of...   
  
 tokens person1 \  
0 [The, two, main, sampling, techniques, used, a... two   
1 [Structured, ,, semi, -structured, ,, and, uns... Structured   
2 [Volume, ,, Velocity, ,, Variety, ,, Veracity,... Value   
3 [NLP, stands, for, Natural, Language, Processi... NLP   
4 [Correlation, and, covariance, are, two, measu... two   
  
 person2 person1\_word\_idx person2\_word\_idx location1 \  
0 • Non (1, 1) (12, 26, 27) 0   
1 three (0, 0) (10, 10) 0   
2 five (9, 9) (12, 12) 0   
3 Natural Language Processing (0, 0) (3, 4) 0   
4 two (4, 4) (4, 4) 0   
  
 location2 family1 family2 location\_yes family\_yes   
0 0 0 0 0 0   
1 0 0 0 0 0   
2 0 0 0 0 0   
3 0 0 0 0 0   
4 0 0 0 0 0

* What might you do to improve the default NER extraction?

## Knowledge Graphs

### Slides Version

* Based on the chosen text, add entities to a default spacy model.
* Add a norm\_entity, merge\_entity, and init\_coref pipelines.
* Update and add the alias lookup if necessary for the data.
* Add the name resolver pipeline.

### Or Use Your Snorkel Output

* Create a co-occurrence graph of the entities linked together in your text.

# 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\_combined['family\_yes'] = DF\_combined['family\_yes'].abs()  
  
DF\_fam = DF\_combined[DF\_combined['family\_yes'] > 0]  
DF\_fam = DF\_fam[['person1', 'person2']].reset\_index(drop = True)  
  
#print(DF\_fam.head())  
  
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())

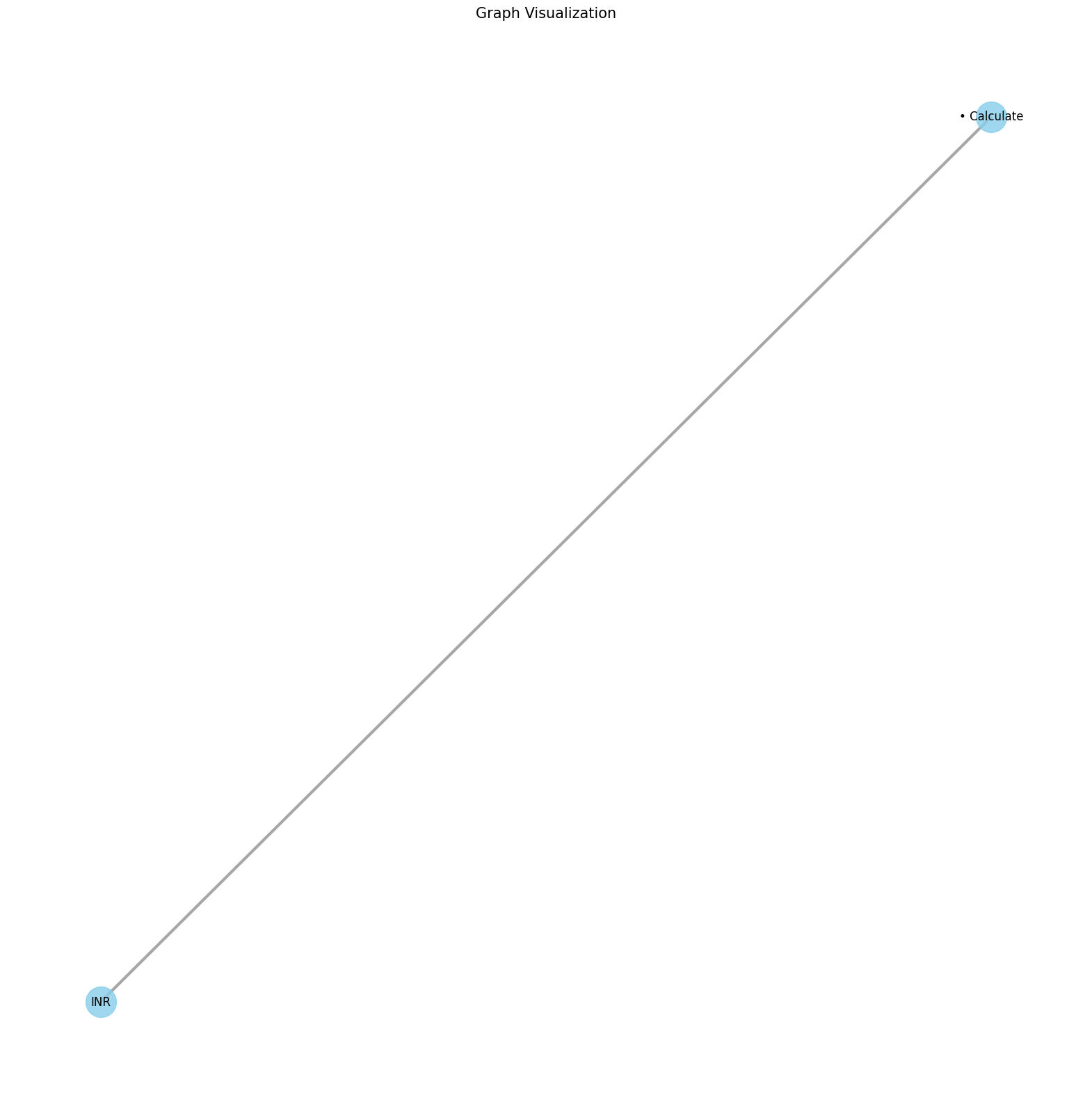
person1 person2 size  
0 Bivariate two 1  
1 Build Looking US 1  
2 Nunique PostgreSQL 1  
3 ROC US 1  
4 SELECT Name FROM StudentData WHERE Department ... three 1  
Empty DataFrame  
Columns: [person1, person2, size]  
Index: []

* This creates a dataframe of node 1 and then node 2 (entity 1 to entity 2) and then frequency (size)

# 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()

# Filter the data  
cooc\_loc\_small = cooc\_loc[cooc\_loc['size'] > 1]  
  
# Create the graph from the filtered dataframe  
graph = nx.from\_pandas\_edgelist(  
 cooc\_loc\_small,  
 source='person1',  
 target='person2',  
 edge\_attr='size'  
)  
  
# Generate positions for each node using Kamada-Kawai layout considering the edge weights  
pos = nx.kamada\_kawai\_layout(graph, weight='size')  
  
# Plotting  
plt.figure(figsize=(20, 20))  
# Draw nodes  
nx.draw\_networkx\_nodes(graph, pos, node\_size=1000, node\_color='skyblue', alpha=0.8)  
# Draw labels  
nx.draw\_networkx\_labels(graph, pos)  
# Draw edges  
edges = nx.draw\_networkx\_edges(graph, pos, edge\_color='#505050', alpha=0.5,   
 width=[data['size'] for \_, \_, data in graph.edges(data=True)])  
  
plt.title('Graph Visualization', size=15)  
plt.axis('off') # Turn off the axis  
plt.show()



print(cooc\_fam.head())

Empty DataFrame  
Columns: [person1, person2, size]  
Index: []

# Filter the data  
cooc\_fam  
  
# Create the graph from the filtered dataframe  
graph = nx.from\_pandas\_edgelist(  
 cooc\_fam,  
 source='person1',  
 target='person2',  
 edge\_attr='size'  
)  
  
# Generate positions for each node using Kamada-Kawai layout considering the edge weights  
pos = nx.kamada\_kawai\_layout(graph, weight='size')  
  
# Plotting  
plt.figure(figsize=(20, 20))  
# Draw nodes  
nx.draw\_networkx\_nodes(graph, pos, node\_size=1000, node\_color='skyblue', alpha=0.8)  
# Draw labels  
nx.draw\_networkx\_labels(graph, pos)  
# Draw edges  
edges = nx.draw\_networkx\_edges(graph, pos, edge\_color='#505050', alpha=0.5,   
 width=[data['size'] for \_, \_, data in graph.edges(data=True)])  
  
plt.title('Graph Visualization', size=15)  
plt.axis('off') # Turn off the axis  
plt.show()



I had to make couple of changes in code.Some libraries were not working due to compatibility issues. graphs were not being plotted.Also, Family df was empty df hence converted -ve numbers to +ve.( we were using family\_yes > 0 and numbers were 0,-1,-2. We could have solved it by family\_yes < 0 but it could have caused issus futher due to -ve numbers hence made +ve)

The most common part of speech in the analyzed text is NOUN, with a total count of 10,799 occurrences.There are multiple parts of the speech.

The key phrase extraction outputs from Textacy and Summa show distinct characteristics. Textacy's output includes specific phrases like 'old man Marsh' and 'old Captain Obed Marsh', which seem directly extracted from the text. It also combines similar terms into sets, indicating an attempt to consolidate variations of key phrases.Summa's output, represented by the TR\_keywords, includes more granular terms like 'things' and 'streets', with associated scores indicating their relevance. The terms are more individualized and not grouped into phrases, suggesting Summa focuses on singular keywords rather than multi-word phrases.

The differences indicate that Textacy might be better suited for extracting and consolidating multi-word key phrases,while Summa appears to emphasize the importance of individual terms within the text. Given the context, Textacy's approach might provide more contextual insights into the text's themes, whereas Summa offers a breakdown of key terms by their significance.

To enhance default NER extraction, we can train the model with domain-specific data and incorporating contextual rules for better entity recognition.