NLP

Find your favorite news source and grab the article text.

1. Show the most common words in the article.
2. Show the most common words under a part of speech. (i.e. NOUN: {'Bob':12, 'Alice':4,})
3. Find a subject/object relationship through the dependency parser in any sentence.
4. Show the most common Entities and their types.
5. Find Entites and their dependency (hint: entity.root.head)
6. Find the most similar words in the article

Note: Yes, the notebook from the video is not provided, I leave it to you to make your own :) it's your final assignment for the semester. Enjoy!

0. Importing Text

A. importing a part of an article from a webpage using 'requests' and 'BeautifulSoup' libraries

- Article : Who is the British royal family willing to protect? https://www.vox.com/culture/24099969/kate-middleton-missing-controversy-meghan-markle-british-royal-family (https://www.vox.com/culture/24099969/kate-middleton-missing-controversy-meghan-markle-british-royal-family)
- requests library : https://pypi.org/project/requests/ (https://pypi.org/project/requests/)
- BeautifulSoup library : https://beautiful-soup-4.readthedocs.io/en/latest/#quick-start (https://beautiful-soup-4.readthedocs.io/en/latest/#quick-start)

In [1]:

```
import requests

#Set Url
url = 'https://www.vox.com/culture/24099969/kate-middleton-missing-controversy-meghan-markle-br

# get html text form the Url
article = requests.get(url)
html = article.text
```

In [2]:

```
#Install and import BeatifulSoup
!pip install beautifulsoup4

from bs4 import BeautifulSoup
```

```
Requirement already satisfied: beautifulsoup4 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages
(4.12.2)
Requirement already satisfied: soupsieve>1.2 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (fr
om beautifulsoup4) (2.4)
```

In [3]:
```
# scrap text from html using BeautifulSoup
soup = BeautifulSoup (html, 'html.parser')
text = soup.get_text()
print(text)
```

```




What happened to Kate Middleton? - Vox
```

B. Load spacy library and save text

In [4]:
```
#Load spacy
import spacy
```

In [5]:
```
#Download medium(which includes vectors)-sized english model
!python -m spacy download en_core_web_md
```

```
Requirement already satisfied: charset-normalizer<4,>=2 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (2023.11.17)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (0.1.4)
Requirement already satisfied: colorama in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from tqdm<5.0.0,>=4.38.0->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (0.4.6)
Requirement already satisfied: click<9.0.0,>=7.1.1 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from typer<0.10.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (8.0.4)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in c:₩users₩kcosm₩anaconda3₩lib₩site-packages (from weasel<0.4.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-web-md==3.7.1) (0.1
```

In [6]:
```
#Load English Dataset, following the guideline (https://spacy.io/models)
nlp = spacy.load("en_core_web_md")
import en_core_web_md
nlp = en_core_web_md.load()
```

In [7]:
```
# save text as 'doc'
doc = nlp(text)
doc
```

Out[7]:
```
What happened to Kate Middleton? - Vox
```

C. Tokenizing the text and convert informations into pandas DataFrame

In [8]:
```
#Create a dataframe to contain token informations

import pandas as pd

df = pd.DataFrame({'text' : ' ' ,
                   'pos' : ' ',
                   'lemma' : ' ',
                   'entity' : ' ',
                    'dependency' : ' '}, index = [0])
df
```

Out[8]:

	text	pos	lemma	entity	dependency
0					

In [9]:
```
n = 1

#Text tokenization
for sentence in doc.sents:
    for token in sentence:

        #Create each row of dataframe containing informations of each token
        i = pd.DataFrame({'text' : token.text,
                          'pos' : token.pos_,
                          'lemma' : token.lemma_,
                          'entity' : token.ent_type_,
                          'dependency' : token.dep_}, index = [n])

        #Append each row of dataframe (There was no 'append' attribution in pandas dataframe so
        df = pd.concat([df, i])
        n += 1

df
```

Out[9]:

	text	pos	lemma	entity	dependency
0					
1	\n\n\n\n	SPACE	\n\n\n\n		dep
2	What	PRON	what		nsubj
3	happened	VERB	happen		ROOT
4	to	ADP	to		prep
...	...	...	...	...	...
2242	.	PUNCT	.		punct
2243	All	DET	all		det
2244	Rights	PROPN	Rights		compound
2245	Reserved	PROPN	Reserved		ROOT
2246	\n \n \n\n\n\n\n\n\n\n\n\n	SPACE	\n \n \n\n\n\n\n\n\n\n\n\n		dep

2247 rows × 5 columns

In [10]:
```
#drop rows with '₩n' values
df = df[~df.text.str.contains("₩n")]

#drop the first row
df = df.drop([0])

df
```

Out[10]:

	text	pos	lemma	entity	dependency
2	What	PRON	what		nsubj
3	happened	VERB	happen		ROOT
4	to	ADP	to		prep
5	Kate	PROPN	Kate	PERSON	compound
6	Middleton	PROPN	Middleton	PERSON	pobj
...	...	...	...	...	...
2241	LLC	PROPN	LLC	ORG	appos
2242	.	PUNCT	.		punct
2243	All	DET	all		det
2244	Rights	PROPN	Rights		compound
2245	Reserved	PROPN	Reserved		ROOT

2029 rows × 5 columns

1. Show the most common words in the article

In [11]:
```
#Count Values of 'text' column using value_counts()
#Using 'to_string()'' attribute to see all values without truncation

print((df['text'].value_counts()).to_string())
```

```
text
,                       85
.                       84
the                     72
to                      54
and                     45
a                       33
of                      31
’ s                      30
Kate                    26
for                     24
is                      21
in                      21
that                    19
with                    15
Meghan                  14
her                     12
more                    12
Vox                     12
```

2. Show the most common words under a part of speech. (i.e. NOUN: {'Bob':12, 'Alice':4,})

In [12]:
```
#Generate array containing unique values of pos(part of speech)

pos = df['pos'].unique()

pos
```

Out[12]:
```
array(['PRON', 'VERB', 'ADP', 'PROPN', 'PUNCT', 'ADJ', 'NOUN', 'ADV',
       'INTJ', 'NUM', 'AUX', 'DET', 'PART', 'SPACE', 'CCONJ', 'SYM',
       'SCONJ'], dtype=object)
```

In [13]:
```
#Create for loop to print out the most common words for each pos

for p in pos:

    # create dataframe only containing rows with each pos value
    df2 = df[df['pos'].isin([p])]

    # count words for each pos value
    count = df2['text'].value_counts()

    # print value_counts for three most common words
    print ("three most common words in", p)
    print (count[0:3])
    print ('--------------------------------')
```

```
three most common words in PRON
text
her      12
she      10
they      9
Name: count, dtype: int64
--------------------------------
three most common words in VERB
text
’s          13
protect     5
signing     4
Name: count, dtype: int64
--------------------------------
three most common words in ADP
text
of     31
for    23
in     21
Name: count, dtype: int64
```

3. Find a subject/object relationship through the dependency parser in any sentence.

In [14]:
```
# extract two sentences from the article

sent = nlp("""Have you heard the news? Princess Catherine of Wales, formerly Kate Middleton, se
```

In [15]:

```
# define pr_tree as done in lecture

def pr_tree(word, level):
    if word.is_punct:
        return
    for child in word.lefts:
        pr_tree(child, level+1)
    print('          '*level + word.text + '-' + word.dep_)
    for child in word.rights:
        pr_tree(child, level+1)
```

In [16]:

```
#run for loops for each sentence
for sentence in sent.sents:
    pr_tree(sentence.root, 0)
    print('--------------------------------------------------------')
```

```
          Have-aux
          you-nsubj
heard-ROOT
                    the-det
          news-dobj
--------------------------------------------------------
                    Princess-compound
          Catherine-nsubj
                    of-prep
                              Wales-pobj
                              formerly-advmod
                              Kate-compound
                    Middleton-appos
seems-ROOT
                    to-aux
          be-xcomp
                    missing-acomp
--------------------------------------------------------
```

4. Show the most common Entities and their types.

In [17]:

```
#Count Values of 'entity' column using 'value_counts()'

print(df['entity'].value_counts())
```

```
entity
               1721
PERSON           96
DATE             72
ORG              52
LAW              24
WORK_OF_ART      17
NORP             11
GPE               8
CARDINAL          8
FAC               6
TIME              5
PRODUCT           3
ORDINAL           3
MONEY             2
LOC               1
Name: count, dtype: int64
```

5. Find Entites and their dependency (hint: entity.root.head)

In [18]:
```
entities = df['entity'].unique()

entities
```

Out[18]:
```
array(['', 'PERSON', 'ORG', 'NORP', 'DATE', 'LOC', 'TIME', 'PRODUCT',
       'GPE', 'FAC', 'ORDINAL', 'CARDINAL', 'WORK_OF_ART', 'LAW', 'MONEY'],
      dtype=object)
```

In [19]:
```
#Create for loop to print out dependency of each entity type

for entity in entities:

    # create dataframe only containing rows with each pos value
    df3 = df[df['entity'].isin([entity])]

    # count words for each pos value
    count = df3['dependency'].value_counts()

    # print value_counts for three most common words
    print ("dependency in", entity)
    print (count)
    print ('--------------------------------')
```

```
dependency in
dependency
punct       224
prep        179
det         135
pobj        118
nsubj       111
compound    109
ROOT        101
advmod       81
dobj         75
amod         75
aux          70
conj         63
cc           54
poss         34
ccomp        31
xcomp        31
mark         26
acomp        24
```

6. Find the most similar words in the article

In [20]:
```
#define 'similarity' as an array
similarity = []
```

In [21]:

```
#add similarity of each tokens in 'similarity' array using for-loop
for token1 in doc:
    for token2 in doc:
        if token1.is_alpha and token2.is_alpha and token1.text != token2.text:
            similarity.append((token1.text, token2.text, token1.similarity(token2)))
```

```
C:\Users\kcosm\AppData\Local\Temp\ipykernel_55208\1321941786.py:4: UserWarning: [W008] Evaluat
ing Token.similarity based on empty vectors.
  similarity.append((token1.text, token2.text, token1.similarity(token2)))
```

In [22]:

```
#Sort with similarity
similarity.sort(key=lambda x: x[2], reverse=True)
```

In [23]:

```
similarity
```

Out[23]:

```
[('Meghan', 'Markle', 1.0000001192092896),
 ('Meghan', 'Sussexes', 1.0000001192092896),
 ('photo', 'photoshoot', 1.0000001192092896),
 ('photo', 'photoshoot', 1.0000001192092896),
 ('photo', 'photoshoot', 1.0000001192092896),
 ('photo', 'photoshoot', 1.0000001192092896),
 ('photo', 'photoshoot', 1.0000001192092896),
 ('photo', 'photoshoot', 1.0000001192092896),
 ('Meghan', 'Markle', 1.0000001192092896),
 ('Meghan', 'Sussexes', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
 ('Markle', 'Sussexes', 1.0000001192092896),
 ('Markle', 'Meghan', 1.0000001192092896),
```

Words with the highest similarity

- 'Meghan' and 'Markle' and 'Sussexes')
- 'photo' and 'photoshoot'