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!

In [1]:
```
from bs4 import BeautifulSoup
import requests
import re
import smartquote
import pandas as pd
import spacy
nlp = spacy.load("en_core_web_lg")
```

In [2]:
```
site = "https://www.foxsports.com/stories/soccer/christian-pulisic-scores-biggest-usa-
res = requests.get(site)
soup = BeautifulSoup(res.content, "html.parser")
```

In [3]:
```
content = soup.find(class_ = 'story-content').text

article = re.findall('DOHA, Qatar.+pain of failure.', content)[0]
article = re.sub('Deandre Yedlin .+ \(Photo by Claudio Villa/Getty Images\)', '', arti
article = re.sub('USA\'s .+ in 38\'', '', article)
article = smartquote.substitute(article)
article = re.sub('\.(?=\w)', '. ', article)
article = re.sub('\s+', ' ', article)
```

In [4]:
```
with open("article.txt", "w") as text_file:
    text_file.write(article)
```

1. Show the most common words in the article

In [5]:
```
doc = nlp(article)
tokens = [token for token in doc]
```

In [6]:
```
tokens_text = [token.text for token in tokens
          if not token.is_punct
          and not token.is_stop
          and not token.is_digit
          ]
```

In [7]:
```
words = pd.Series(tokens_text).value_counts(ascending = False)
```

In [8]:
```
words[0:9]
```

Out[8]:
```
Iran         9
goal         7
face         5
Pulisic      5
ball         5
Americans    4
game         4
winning      3
team         3
dtype: int64
```

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

In [9]:
```
tokens_pos = [token.pos_ for token in tokens
          if not token.is_punct
          and not token.is_stop
          and not token.is_digit
          ]
```

In [10]:
```
tokens_df = pd.DataFrame(zip(tokens_text, tokens_pos), columns = ['word', 'pos'])
```

In [11]:
```
with pd.option_context('display.max_rows', None
                       ):
    print(tokens_df.groupby(by='pos')['word'].value_counts(ascending = False))
```

pos	word	
ADJ	American	2
	defensive	2
	free	2
	abdominal	1
	better	1
	busier	1
	calm	1
	cautious	1
	different	1
	emotional	1
	excellent	1
	fierce	1
	final	1
	finer	1
	heroic	1
	important	1
	little	1
	national	1
	new	1
	old	1
	paramount	1
	past	1
	political	1
	potential	1
	ready	1
	rife	1
	set	1
	sizable	1
	speedy	1
	total	1
	ultra	1
	unflinching	1
	unmarked	1
	winded	1
	young	1
ADP	alongside	1
ADV	forward	2
	longer	2
	Earlier	1
	Late	1
	Suddenly	1
	barely	1
	exclusively	1
	forever	1
	gradually	1
	heavily	1
	home	1
	ill	1
	immediately	1
	lively	1
	near	1
	offside	1
	presumably	1
	right	1
	seemingly	1
	wide	1
NOUN	goal	7
	ball	5
	face	5

game	4
break	3
minutes	3
team	3
bar	2
failure	2
half	2
midfield	2
moments	2
need	2
pain	2
price	2
time	2
way	2
yards	2
Ghoddos	1
action	1
anthem	1
area	1
bench	1
body	1
boost	1
boot	1
box	1
breakaways	1
buildup	1
chances	1
chest	1
clash	1
coach	1
contest	1
country	1
days	1
dedication	1
defender	1
defense	1
defiance	1
deficiency	1
devil	1
effectiveness	1
elimination	1
emotion	1
end	1
equation	1
fans	1
flag	1
flank	1
folly	1
foot	1
forwards	1
frustration	1
gain	1
goalkeeper	1
goals	1
halftime	1
hand	1
head	1
header	1
history	1
hospital	1

	injury	1
	ins	1
	kick	1
	kicks	1
	knockout	1
	lead	1
	line	1
	man	1
	matchup	1
	men	1
	nerve	1
	net	1
	night	1
	onslaught	1
	option	1
	pair	1
	pass	1
	pieces	1
	players	1
	possession	1
	pre	1
	pressure	1
	priorities	1
	race	1
	reach	1
	resolve	1
	role	1
	round	1
	rounds	1
	scans	1
	shin	1
	shot	1
	shoulder	1
	speeding	1
	spirit	1
	squad	1
	stage	1
	striker	1
	superstar	1
	surprise	1
	tactics	1
	talk	1
	task	1
	teams	1
	teen	1
	threat	1
	throw	1
	tournament	1
	turf	1
	underdog	1
	undertones	1
	wall	1
	win	1
	winner	1
	year	1
	years	1
PROPN	Iran	9
	Pulisic	5
	Americans	4
	Dest	3

	Taremi	3
	Turner	3
	Berhalter	2
	Carter	2
	McKennie	2
	Mehdi	2
	USA	2
	Vickers	2
	Weah	2
	Zimmerman	2
	Aaronson	1
	Al	1
	Alireza	1
	American	1
	Azmoun	1
	Beiranvand	1
	Brenden	1
	Cameron	1
	Carlos	1
	Champions	1
	Chelsea	1
	Christian	1
	Cup	1
	DOHA	1
	England	1
	Ghoddos	1
	Gregg	1
	Josh	1
	League	1
	Matt	1
	Morteza	1
	Netherlands	1
	Pouraliganji	1
	Qatar	1
	Queiroz	1
	Ream	1
	Saman	1
	Sardar	1
	Sargent	1
	Saturday	1
	Sergiño	1
	Stadium	1
	States	1
	Thumama	1
	Tim	1
	Timothy	1
	Tuesday	1
	United	1
	Walker	1
	Weston	1
	World	1
PUNCT	"Onward	1
SCONJ	despite	2
VERB	winning	3
	advanced	1
	advised	1
	announced	1
	beat	1
	belted	1
	broke	1

brought	1
care	1
changed	1
collected	1
collided	1
commanding	1
connect	1
consider	1
contorted	1
creating	1
crunch	1
curled	1
decided	1
destined	1
emerge	1
equalizing	1
etched	1
failed	1
fell	1
figurehead	1
floated	1
gained	1
got	1
headed	1
hold	1
introducing	1
know	1
knowing	1
lay	1
liked	1
lost	1
march	1
muscling	1
nearing	1
needed	1
needs	1
nodded	1
paid	1
play	1
pounce	1
prepared	1
pressed	1
prevent	1
proven	1
received	1
relegating	1
replaced	1
required	1
rubbed	1
ruled	1
saw	1
score	1
scored	1
scoring	1
secure	1
seeing	1
showed	1
sit	1
sitting	1
sought	1

```
spotted          1
sprung           1
steer            1
steered          1
struggled        1
surviving        1
taken            1
taking           1
took             1
underestimate    1
whipped          1
won              1
wracking         1
write            1
Name: word, dtype: int64
```

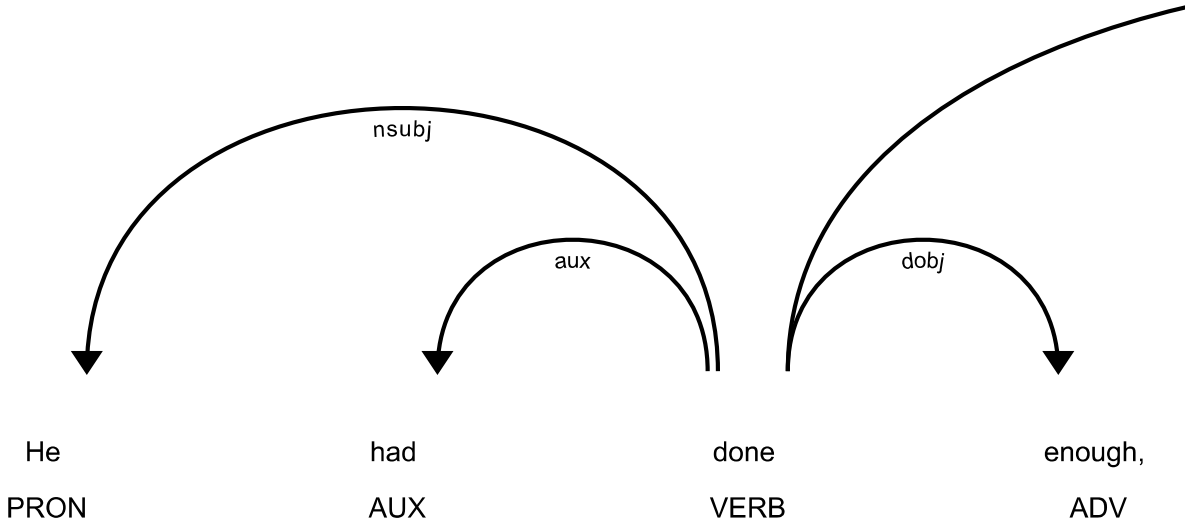
1. Find a subject/object relationship through the dependency parser in any sentence.

In [12]:
```
def pr_tree(word, level):
    if word.is_punct:
        return
    for child in word.lefts:
        pr_tree(child, level + 1)
    print('\t' * level + word.text + ' - ' + word.dep_)
    for child in word.rights:
        pr_tree(child, level + 1)
```

In [13]:
```
sel_sent = 11
i = 0
for sent in doc.sents:
    if i == sel_sent:
        print(sent)
        sel_sent_text = sent
        pr_tree(sent.root, 0)
        print('-' * 100)
    i+=1
```

```
He had done enough, if you consider scoring the most immediately important goal by an
American man in 12 years to be "enough.
        He - nsubj
        had - aux
done - ROOT
        enough - dobj
                if - mark
                you - nsubj
        consider - advcl
                scoring - xcomp
                        the - det
                                most - advmod
                        immediately - advmod
                        important - amod
                goal - dobj
                        by - prep
                                an - det
                                American - amod
                        man - pobj
                in - prep
                        12 - nummod
                        years - pobj
                to - aux
                be - xcomp
                        enough - acomp
--------------------------------------------------------------------------------
---------------
```

In [14]:
```
spacy.displacy.render(sel_sent_text, style = 'dep')
```



1. Show the most common Entities and their types.

In [15]:
```
ent_text = []
ent_type = []
for token in doc.ents:
    ent_text.append(token.text)
    ent_type.append(token.label_)
```

In [16]:
```
ent_df = pd.DataFrame(
    zip(ent_text, ent_type),
    columns = ['text', 'type']
)
```

In [17]:
```
ent_df.groupby(by='type')['text'].value_counts(ascending=False)
```

Out[17]:

type	text	
CARDINAL	0	1
	1	1
	10	1
	16	1
	one	1
DATE	12 years	1
	24-year-old	1
	Saturday	1
	the first half	1
	the past days	1
EVENT	the World Cup	1
FAC	Al Thumama Stadium	1
GPE	Iran	9
	USA	2
	England	1
	Netherlands	1
	Qatar	1
	the United States	1
NORP	Americans	4
	American	3
ORDINAL	first	3
ORG	Dest	2
	Pulisic	2
	Turner	2
	Carter-Vickers	1
	Champions League	1
	Chelsea	1
	McKennie	1
	Sergiño Dest	1
	Taremi	1
	flank	1
PERSON	Mehdi Taremi	2
	Pulisic	2
	Alireza Beiranvand	1
	Berhalter	1
	Brenden Aaronson	1
	Cameron Carter-Vickers	1
	Carlos Queiroz	1
	Christian Pulisic	1
	DOHA	1
	Ghoddos	1
	Gregg Berhalter's	1
	Josh Sargent	1
	Matt Turner's	1
	Morteza Pouraliganji	1
	Saman Ghoddos	1
	Sardar Azmoun	1
	Tim Ream	1
	Timothy Weah	1
	Walker Zimmerman	1
	Weah	1
	Weston McKennie	1
	Zimmerman	1
QUANTITY	a few yards	1
	six yards	1
TIME	38 minutes	1
	52 minutes	1
	Tuesday night	1

```
          several minutes           1
Name: text, dtype: int64
```

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

In [18]:
```
sel_ent = 8
i = 0
for ent in doc.ents:
    if i == 8:
        print(ent, "-" * 5, ent.label_)
        pr_tree(ent.root.head, 0)
        print('*' * 100)
    i+=1
```

```
Pulisic ----- ORG
        Pulisic - nsubj
                                the - det
                        superstar - nsubj
                figurehead - appos
                        for - prep
                                        this - det
                                        young - amod
                                        USA - compound
                                team - pobj
saw - ROOT
                that - det
        equation - dobj
        liked - conj
                it - dobj
                and - cc
                decided - conj
                                to - aux
                        go - xcomp
                                with - prep
                                                the - det
                                                winning - amod
                                        option - pobj
****************************************************************************************
***************
```

1. Find the most similar words in the article

In [19]:
```
doc = nlp(article.lower())
compare_list = []
for token1 in doc:
    for token2 in doc:
        sim = token1.similarity(token2)
        if 0.75 < sim < 1:
            if token1.text == token2.text or sim in compare_list:
                continue
            print(f'{token1.text}: {token2.text} {"-"*5} {token1.similarity(token2)}')
            compare_list.append(token1.similarity(token2))
```

```
C:\Users\gdlev\AppData\Local\Temp\ipykernel_5952\2918615760.py:5: UserWarning: [W008]
Evaluating Token.similarity based on empty vectors.
  sim = token1.similarity(token2)
```

scored: scoring ----- 0.8124553561210632
goal: goals ----- 0.80790114402771
that: because ----- 0.7600496411323547
that: however ----- 0.7528941631317139
tuesday: saturday ----- 0.8429607152938843
really: because ----- 0.772476315498352
really: just ----- 0.7586804628372192
anyone: anything ----- 0.7538626790046692
because: but ----- 0.8247512578964233
because: though ----- 0.8218691349029541
because: however ----- 0.7560612559318542
nothing: everything ----- 0.8324678540229797
nothing: anything ----- 0.8896462917327881
everything: anything ----- 0.8457676768302917
team: teams ----- 0.8131204843521118
too: so ----- 0.8098674416542053
but: though ----- 0.8362677097320557
but: however ----- 0.776674747467041
may: will ----- 0.7649534940719604
38: 52 ----- 0.910517692565918
defense: defensive ----- 0.789395809173584
midfield: goalkeeper ----- 0.7685624957084656
midfield: striker ----- 0.7810852527618408
forward: forwards ----- 0.8401443958282471
24: 12 ----- 0.790008544921875
24: 16 ----- 0.7861243486404419
champions: tournament ----- 0.7556649446487427
goalkeeper: striker ----- 0.8447802066802979
after: before ----- 0.8302974700927734
taken: taking ----- 0.7662090063095093
12: 16 ----- 0.9027277231216431
12: 10 ----- 0.8590026497840881
years: days ----- 0.7615094780921936
round: rounds ----- 0.7705825567245483
16: 10 ----- 0.7613131999969482
them: themselves ----- 0.78706955909729
though: however ----- 0.8572306632995605
emotional: emotion ----- 0.852793276309967
needed: need ----- 0.8177447319030762
effectiveness: potential ----- 0.7707822322845459
taking: take ----- 0.7769252061843872
kicks: kick ----- 0.8266697525978088
introducing: creating ----- 0.7714839577674866
defender: striker ----- 0.7669340968132019