NLP Assignment #5

Luke Schwenke

Assignment Goals

1. Identify what is this company name, by looking at the entity distributions across both tweets and news articles
2. Identify what other companies are most frequently mentioned along with your primary company

- Analyze what companies are most frequently mentioned within the same document (tweet and news article)
- While analyzing news articles, extract separate entities from titles and texts

1. Identify most frequent locations of events, by extracting appropriate named entities

- Locations may include countries, states, cities, regions, etc.

In [ ]:
```
import pandas as pd
import nltk
import spacy
from collections import Counter
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize, word_tokenize


pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', 500)
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
import multiprocessing
num_processors = multiprocessing.cpu_count()
print(f'Available CPUs: {num_processors}')
```

```
Available CPUs: 8
```

In [ ]:
```
from pandarallel import pandarallel
pandarallel.initialize(nb_workers=num_processors-1, use_memory_fs=False, progre
```

```
INFO: Pandarallel will run on 7 workers.
INFO: Pandarallel will use standard multiprocessing data transfer (pipe) to tr
ansfer data between the main process and workers.
```

Read news data

In [ ]:
```
news_path = 'https://storage.googleapis.com/msca-bdp-data-open/news/nlp_a_5_new
news_df = pd.read_json(news_path, orient='records', lines=True)
```

```
print(f'Sample contains {news_df.shape[0]:,.0f} news articles')
news_df.head(2)
```

```
Sample contains 10,012 news articles
```

Out[ ]:

	url	date	language	title	
0	http://kokomoperspective.com/obituaries/jon-w-horton/article_b6ba8e1e-cb9c-11eb-9868-fb11b88b9778.html	2021-06-13	en	Jon W. Horton | Obituaries | kokomoperspective.com	U
1	https://auto.economictimes.indiatimes.com/news/auto-components/birla-precision-to-ramp-up-capacity-to-tap-emerging-opportunities-in-india/81254902	2021-02-28	en	Birla Precision to ramp up capacity to tap emerging opportunities in India, Auto News, ET Auto	

Read Tweets data

In [ ]:
```
tweets_path = 'https://storage.googleapis.com/msca-bdp-data-open/tweets/nlp_a_
tweets_df = pd.read_json(tweets_path, orient='records', lines=True)
print(f'Sample contains {tweets_df.shape[0]:,.0f} tweets')
tweets_df.head(2)
```

```
Sample contains 10,105 tweets
```

Out[ ]:

	id	lang	date	name	retweeted	
0	1534565117614084096	en	2022-06-08	Low Orbit Tourist 🌍📷		Body &amp; Assembly - H United King 53.3504,-2.8352296,402m\n\n Body &amp; Assembly is a Ja Rover factory in Halewood, Eng forms the major part of the complex which is shared with manufacture transmissions a [Wikipedia] https://t.co/LP
1	1534565743429394439	en	2022-06-08	CompleteCar.ie	RT	Land Rover Ireland has annou the new Range Rover Spor €114,15 @completecar:\n\nhttps://t.co/Tj https://t.co/Q

Discard non-English results & Apply appropriate text cleaning methods

In [ ]:
```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

nltk.download('words')
from nltk.corpus import words

english_words = set(words.words())
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/lmschwenke/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     /Users/lmschwenke/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

In [ ]:
```
import re

def clean_text(text):
    # Remove mentions
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)
    # Remove hashtags (but keep the text after #)
    text = re.sub(r'#', '', text)
    # Remove RT (retweet symbol)
    text = re.sub(r'RT[\s]+', '', text)
    # Remove hyperlinks
    text = re.sub(r'https?:\/\/\S+', '', text)
    # Remove newline characters
    text = re.sub(r'\n', ' ', text)
    # Remove carriage return characters
    text = re.sub(r'\r', '', text)
    # Remove "&amp;"
    text = re.sub(r'&amp;', '', text)
    # Remove other special characters and numbers
    text = re.sub(r'[^A-Za-z\s]', '', text)
    # Convert multiple spaces to a single space
    text = re.sub(r'\s+', ' ', text)
    # Optionally, convert to lowercase
    # text = text.lower()
    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in stop_words])
    # Remove non-English words
    text = ' '.join([word for word in text.split() if word.lower() in english_w

    return text.strip()
```

In [ ]:
```
tweets_df['tweets_clean'] = tweets_df['text'].parallel_apply(clean_text)
news_df['text_clean'] = news_df['text'].parallel_apply(clean_text)
news_df['title_clean'] = news_df['title'].parallel_apply(clean_text)
```

In [ ]:
```
tweets_df[['tweets_clean']].head(3)
```

Out[ ]:

	tweets_clean
0	Body Assembly United Kingdom Body Assembly Jaguar Land Rover factory major part complex Ford manufacture site
1	Land Rover new Range Rover Sport
2	New Land Rover Range Rover Top Speed With Ease On Autobahn

In [ ]:
```
news_df[['text_clean']].head(3)
```

Out[ ]:

	text_clean
0	W permission edit article Up Log In Dashboard Account Dashboard Profile Saved latest local news IN A stray shower thunderstorm possible Low F light A stray shower thunderstorm possible Low F light variable June Full latest local news COVID Support Local Journalism Now ever world needs trustworthy good journalism free Please support us making contribution Contribute W W June Health Ball Memorial He born August On married Donna The couple blessed two marriage worked Camp Quaker After worked St...
1	Precision ramp capacity tap Auto News Auto We privacy policy Click Continue accept continue privacy user Auto privacy policy align new data Union Please review accept continue see privacy policy policy We use ensure best experience choose ignore message well assume happy receive track site origin track statistics consent state current serve content relevant identify Fingerprinting uniquely identify client Daily daily list important information industry read accepted Retail News Health News N...
2	Global Hydrogen Station Current State Future Prognosis Passenger Hydrogen Fuel Cell HOME MAIL NEWS SPORTS FINANCE CELEBRITY STYLE WEATHER MOBILE Yahoo Sports Sign Mail Sign view mail Sports Home Sports Home Fantasy Fantasy Fantasy Football Fantasy Hockey Fantasy Basketball Fantasy Auto Auto Racing Fantasy Golf Fantasy Baseball Home Yahoo Sports Sports Junior Hockey Home Home More Football Home Soccer Soccer Soccer Home Premier League League A La World Cup Over Over Everything Yahoo Hockey Ho...

In [ ]:

```
news_df[['title_clean']].head(3)
```

Out[ ]:

	title_clean
0	W
1	Precision ramp capacity tap Auto News Auto
2	Global Hydrogen Station Current State Future Prognosis Passenger Hydrogen Fuel Cell

Named Entity Recognition (NER) - Using NLTK for Organizations

In [ ]:

```
# Initialize a counter to keep track of organization frequencies
organization_counter = Counter()

# Define a function to extract organizations using NLTK
def nltk_extract_organizations(text):
    entities = []
    for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(text)), binary=
        if hasattr(chunk, 'label') and chunk.label() == 'ORGANIZATION':
            entities.append(' '.join(c[0] for c in chunk))
    return entities
```

In [ ]:

```
tweets_df['organizations'] = tweets_df['tweets_clean'].parallel_apply(nltk_ext
news_df['organizations_from_text'] = news_df['text_clean'].parallel_apply(nltk
news_df['organizations_from_title'] = news_df['title_clean'].parallel_apply(nl
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['organizations'],
                  "News Text": news_df['organizations_from_text'],
                  "News Title": news_df['organizations_from_title']}

def get_top_orgs(text_body):
    top_organizations_dict = {}

    for key, text in text_body.items():
        all_organizations = [org for org_list in text for org in org_list]

        # Update the organization frequencies
        organization_counter.clear()
        organization_counter.update(all_organizations)

        # Find the most commonly mentioned organizations
        most_common_organizations = organization_counter.most_common(20)

        top_organizations_dict[key] = most_common_organizations

    return top_organizations_dict
```

In [ ]:
```
top_organizations = get_top_orgs(text_body_dict)

print("NLTK Outputs: \n")
for key, organizations in top_organizations.items():
    print(f"{key}:")
    for org, count in organizations:
        print(f"{org}: {count}")
    print()
```

```
NLTK Outputs:

Tweets:
Jaguar Land Rover: 575
Land Rover: 553
Jaguar Land Rover General: 265
Land Rover Defender: 228
Ford: 82
SHAMELESS Health Board: 64
Rover: 61
Land Rover Range Rover: 53
Jaguar Jeep Land Rover: 51
Range Rover: 49
Nestle Jaguar Land Rover: 47
Jaguar: 43
Grenadier Land Rover Defender Business Daily: 43
Health Board: 41
Jaguar Land: 38
Jaguar Land Rover Driving Challenge: 37
Subterranean Challenge: 36
Honda: 29
EU: 22
Gravity Business Park: 20

News Text:
COVID: 9724
Princess: 5707
VERY: 5644
LA: 4825
US: 2932
Duchess: 2501
THE: 2498
House: 2149
NOT: 2127
Mail Media: 1905
ALL: 1433
FIRST: 1280
RELATED: 1277
Lipa: 1228
Republican: 1152
MILLION: 1133
City: 1131
Vanity Fair: 1008
Associated: 963
Land Rover: 961

News Title:
Star News: 177
Ford: 111
Automotive News: 94
Business Live: 59
News: 44
News Driven: 43
Mail: 42
COVID: 40
Fast Lane Car: 33
Land: 31
Ford Escape: 31
RAM: 27
Auto News: 26
Auto News Auto: 24
Express Star: 22
AWD: 21
RAM Sale: 19
Car Expert: 19
AWD Sale: 16
Senate: 15
```

Apply Sentence Segmentation with NLTK for extracting Organizations

In [ ]:
```
def nltk_extract_organizations_sentences(text):
    entities = []

    sentences = sent_tokenize(text)

    for sentence in sentences:
        # Tokenize each sentence into words and perform organization extractio
        for chunk in nltk.ne_chunk(nltk.pos_tag(word_tokenize(sentence)), bina
            if hasattr(chunk, 'label') and chunk.label() == 'ORGANIZATION':
                entities.append(' '.join(c[0] for c in chunk))
    return entities
```

In [ ]:
```
tweets_df['organizations_sentences'] = tweets_df['tweets_clean'].parallel_apply
news_df['organizations_from_text_sentences'] = news_df['text_clean'].parallel_a
news_df['organizations_from_title_sentences'] = news_df['title_clean'].paralle
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['organizations_sentences'],
                  "News Text": news_df['organizations_from_text_sentences'],
                  "News Title": news_df['organizations_from_title_sentences']}

top_organizations = get_top_orgs(text_body_dict)

print("NLTK Outputs - Sentence Segmentation: \n")
for key, organizations in top_organizations.items():
    print(f"{key}:")
    for org, count in organizations:
        print(f"{org}: {count}")
    print()
```

```
NLTK Outputs - Sentence Segmentation:

Tweets:
Land: 925
Land Rover: 694
LAND: 188
ROVER: 128
Duke Duchess: 96
SHAMELESS: 93
Land Rover Discovery: 87
SHAMELESS Health Board Zimbabwe: 64
UPDATE: 53
Jaguar Land: 46
Hospital: 36
BaT Land Rover: 32
FRANCHISE: 32
Duke Duchess Jaguar Land Rover: 28
Rover: 23
BAE Hawk: 20
Defender: 19
NEW: 19
Duke: 18
Land Rover Which: 17

News Text:
COVID: 9724
Princess: 5707
VERY: 5644
LA: 4825
US: 2932
Duchess: 2501
THE: 2498
House: 2149
NOT: 2127
Mail Media: 1905
ALL: 1433
FIRST: 1280
RELATED: 1277
Lipa: 1228
Republican: 1152
MILLION: 1133
City: 1131
Vanity Fair: 1008
Associated: 963
Land Rover: 961

News Title:
Star News: 177
Ford: 111
Automotive News: 94
Business Live: 59
News: 44
News Driven: 43
Mail: 42
COVID: 40
Fast Lane Car: 33
Land: 31
Ford Escape: 31
RAM: 27
Auto News: 26
Auto News Auto: 24
Express Star: 22
AWD: 21
RAM Sale: 19
Car Expert: 19
AWD Sale: 16
Senate: 15
```

Named Entity Recognition (NER) - Using SpaCy for Organizations

In [ ]:
```
import spacy
from spacy import displacy
spacy.prefer_gpu()
# spacy.require_gpu()

print(spacy.__version__)
```

```
3.7.2
```

SpaCy Models:

- en_core_web_sm: English multi-task CNN trained on OntoNotes. Size – 11 MB
- en_core_web_md: English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl. Size – 91 MB
- en_core_web_lg: English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl. Size – 789 MB
- en_core_web_trf: English transformer pipeline (roberta-base). Components: transformer, tagger, parser, ner, attribute_ruler, lemmatizer. Size - 438 MB

In [ ]:
```
if not spacy.util.is_package("en_core_web_lg"):
    # If not, download and install it
    spacy.cli.download("en_core_web_lg")
```

In [ ]:
```
# Load SpaCy model
# nlp = spacy.load("en_core_web_sm")
# nlp = spacy.load("en_core_web_md")
nlp = spacy.load("en_core_web_lg")
# nlp = spacy.load("en_core_web_trf")
```

In [ ]:
```
# Checking active pipeline components
nlp.pipe_names
```

Out[ ]:
```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
```

In [ ]:
```
def spacy_extract_organizations(text):
    doc = nlp(text)
    organizations = [ent.text for ent in doc.ents if ent.label_ == "ORG"]
    return organizations
```

In [ ]:
```
# Apply the extract_organizations function to the "tweets_clean" column
tweets_df['organizations_spacy'] = tweets_df['tweets_clean'].parallel_apply(spa
news_df['organizations_from_text_spacy'] = news_df['text_clean'].parallel_apply
news_df['organizations_from_title_spacy'] = news_df['title_clean'].parallel_app
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['organizations_spacy'],
                  "News Text": news_df['organizations_from_text_spacy'],
                  "News Title": news_df['organizations_from_title_spacy']}
```

In [ ]:
```
top_organizations = get_top_orgs(text_body_dict)

print("SpaCy Outputs: \n")
for key, organizations in top_organizations.items():
    print(f"{key}:")
    for org, count in organizations:
        print(f"{org}: {count}")
    print()
```

```
SpaCy Outputs:

Tweets:
Jaguar Land Rover: 575
Land Rover: 553
Jaguar Land Rover General: 265
Land Rover Defender: 228
Ford: 82
SHAMELESS Health Board: 64
Rover: 61
Land Rover Range Rover: 53
Jaguar Jeep Land Rover: 51
Range Rover: 49
Nestle Jaguar Land Rover: 47
Jaguar: 43
Grenadier Land Rover Defender Business Daily: 43
Health Board: 41
Jaguar Land: 38
Jaguar Land Rover Driving Challenge: 37
Subterranean Challenge: 36
Honda: 29
EU: 22
Gravity Business Park: 20

News Text:
Ford: 4895
House: 3352
Honda: 2509
White House: 2445
Vanity Fair: 2034
Mail Media: 1905
Duchess: 1874
Apple: 1481
Palace: 1480
United: 1479
Vogue: 1476
Jeep: 1076
Royal Family: 1044
Shop: 1041
Land Rover: 972
Dodge: 946
Range Rover: 920
Genesis: 842
Royal: 806
Jaguar: 796

News Title:
Daily Mail: 1484
Ford: 264
Star News: 156
Honda: 127
Autocar: 113
Automotive News: 97
Express Star: 84
Car Dealer Magazine: 77
Daily Times News: 70
Jeep: 46
Dodge: 45
Auto News Auto: 41
Daily Record: 36
```

```
The China Post: 35
Mirror: 33
Jaguar: 27
Star: 27
Jaguar Land Rover: 25
EU: 20
Times: 18
```

Apply Sentence Segmentation with SpaCy for extracting Organizations

In [ ]:
```
def spacy_extract_organizations_sentences(text):
    sentences = sent_tokenize(text)
    organizations = []
    for sentence in sentences:
        doc = nlp(sentence)
        sentence_organizations = [ent.text for ent in doc.ents if ent.label_ ==
        organizations.extend(sentence_organizations)

    return organizations
```

In [ ]:
```
# Apply the extract_organizations function to the "tweets_clean" column
tweets_df['organizations_spacy_sentences'] = tweets_df['tweets_clean'].paralle
news_df['organizations_from_text_spacy_sentences'] = news_df['text_clean'].para
news_df['organizations_from_title_spacy_sentences'] = news_df['title_clean'].pa
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['organizations_spacy_sentences'],
                  "News Text": news_df['organizations_from_text_spacy_sentence
                  "News Title": news_df['organizations_from_title_spacy_senten
```

In [ ]:
```
top_organizations = get_top_orgs(text_body_dict)

print("SpaCy Outputs - Sentences: \n")
for key, organizations in top_organizations.items():
    print(f"{key}:")
    for org, count in organizations:
        print(f"{org}: {count}")
    print()
```

```
SpaCy Outputs - Sentences:

Tweets:
Jaguar Land Rover: 575
Land Rover: 553
Jaguar Land Rover General: 265
Land Rover Defender: 228
Ford: 82
SHAMELESS Health Board: 64
Rover: 61
Land Rover Range Rover: 53
Jaguar Jeep Land Rover: 51
Range Rover: 49
Nestle Jaguar Land Rover: 47
Jaguar: 43
Grenadier Land Rover Defender Business Daily: 43
Health Board: 41
Jaguar Land: 38
Jaguar Land Rover Driving Challenge: 37
Subterranean Challenge: 36
Honda: 29
EU: 22
Gravity Business Park: 20

News Text:
Ford: 4895
House: 3352
Honda: 2509
White House: 2445
Vanity Fair: 2034
Mail Media: 1905
Duchess: 1874
Apple: 1481
Palace: 1480
United: 1479
Vogue: 1476
Jeep: 1076
Royal Family: 1044
Shop: 1041
Land Rover: 972
Dodge: 946
Range Rover: 920
Genesis: 842
Royal: 806
Jaguar: 796

News Title:
Daily Mail: 1484
Ford: 264
Star News: 156
Honda: 127
Autocar: 113
Automotive News: 97
Express Star: 84
Car Dealer Magazine: 77
Daily Times News: 70
Jeep: 46
Dodge: 45
Auto News Auto: 41
Daily Record: 36
```

```
The China Post: 35
Mirror: 33
Jaguar: 27
Star: 27
Jaguar Land Rover: 25
EU: 20
Times: 18
```

Named Entity Recognition (NER) - Using SpaCy for Locations

In [ ]:
```
def spacy_extract_locations(text):
    doc = nlp(text)
    organizations = [ent.text for ent in doc.ents if ent.label_ == "GPE"]
    return organizations
```

In [ ]:
```
tweets_df['locations_spacy'] = tweets_df['tweets_clean'].parallel_apply(spacy_e
news_df['locations_from_text_spacy'] = news_df['text_clean'].parallel_apply(spa
news_df['locations_from_title_spacy'] = news_df['title_clean'].parallel_apply(s
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['locations_spacy'],
                  "News Text": news_df['locations_from_text_spacy'],
                  "News Title": news_df['locations_from_title_spacy']}
```

In [ ]:
```
top_locations = get_top_orgs(text_body_dict)

print("SpaCy Outputs: \n")
for key, locations in top_locations.items():
    print(f"{key}:")
    for loc, count in locations:
        print(f"{loc}: {count}")
    print()
```

```
SpaCy Outputs:

Tweets:
Russia: 464
Zimbabwe: 87
BaT: 42
Brunswick: 40
US: 36
China: 29
Somerset: 28
Mungofa: 20
Cayman: 16
North West: 15
Jordan: 13
Park: 11
st: 9
Japan: 8
LA: 5
Derby: 4
Tableau: 3
Canada: 3
Iceland: 3
Arusha: 2

News Text:
LA: 14597
US: 12184
New York City: 6926
New York: 4093
China: 2422
Las: 1572
Russia: 1504
Canada: 960
Trump: 824
Turkey: 787
Japan: 778
San: 635
Brazil: 623
Michigan: 571
TOWN: 533
Jordan: 514
Boston: 487
New Jersey: 417
Colorado: 380
Orange County: 374

News Title:
US: 132
North York: 72
China: 33
Russia: 30
Scotia: 29
Saskatoon: 26
Midland: 18
Tilbury: 18
Japan: 13
Somerset: 11
Commonwealth: 11
New York: 11
LA: 9
```

```
Canada: 9
Cobourg: 8
Michigan: 7
Guinea: 6
Colorado: 6
Brazil: 5
Turkey: 4
```

Apply Sentence Segmentation with SpaCy for extracting Locations

In [ ]:
```
def spacy_extract_locations_sentences(text):
    sentences = sent_tokenize(text)
    organizations = []
    for sentence in sentences:
        doc = nlp(sentence)
        sentence_organizations = [ent.text for ent in doc.ents if ent.label_ =
        organizations.extend(sentence_organizations)

    return organizations
```

In [ ]:
```
# Apply the extract_organizations function to the "tweets_clean" column
tweets_df['locations_spacy_sentences'] = tweets_df['tweets_clean'].parallel_ap
news_df['locations_from_text_spacy_sentences'] = news_df['text_clean'].paralle
news_df['locations_from_title_spacy_sentences'] = news_df['title_clean'].paral
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['locations_spacy_sentences'],
                  "News Text": news_df['locations_from_text_spacy_sentences'],
                  "News Title": news_df['locations_from_title_spacy_sentences'
```

In [ ]:
```
top_locations = get_top_orgs(text_body_dict)

print("SpaCy Outputs - Sentences: \n")
for key, locations in top_locations.items():
    print(f"{key}:")
    for loc, count in locations:
        print(f"{loc}: {count}")
    print()
```

```
SpaCy Outputs - Sentences:

Tweets:
Russia: 464
Zimbabwe: 87
BaT: 42
Brunswick: 40
US: 36
China: 29
Somerset: 28
Mungofa: 20
Cayman: 16
North West: 15
Jordan: 13
Park: 11
st: 9
Japan: 8
LA: 5
Derby: 4
Tableau: 3
Canada: 3
Iceland: 3
Arusha: 2

News Text:
LA: 14597
US: 12184
New York City: 6926
New York: 4093
China: 2422
Las: 1572
Russia: 1504
Canada: 960
Trump: 824
Turkey: 787
Japan: 778
San: 635
Brazil: 623
Michigan: 571
TOWN: 533
Jordan: 514
Boston: 487
New Jersey: 417
Colorado: 380
Orange County: 374

News Title:
US: 132
North York: 72
China: 33
Russia: 30
Scotia: 29
Saskatoon: 26
Midland: 18
Tilbury: 18
Japan: 13
Somerset: 11
Commonwealth: 11
New York: 11
LA: 9
```

```
Canada: 9
Cobourg: 8
Michigan: 7
Guinea: 6
Colorado: 6
Brazil: 5
Turkey: 4
```

Named Entity Recognition (NER) - Using NLTK for Locations

In [ ]:
```
# Initialize a counter to keep track of organization frequencies
organization_counter = Counter()

# Define a function to extract organizations using NLTK
def nltk_extract_locations(text):
    entities = []
    for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(text)), binary=
        if hasattr(chunk, 'label') and chunk.label() == 'GPE':
            entities.append(' '.join(c[0] for c in chunk))
    return entities
```

In [ ]:
```
tweets_df['locations_nltk'] = tweets_df['tweets_clean'].parallel_apply(nltk_ex
news_df['locations_from_text_nltk'] = news_df['text_clean'].parallel_apply(nlt
news_df['locations_from_title_nltk'] = news_df['title_clean'].parallel_apply(n
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['locations_nltk'],
                  "News Text": news_df['locations_from_text_nltk'],
                  "News Title": news_df['locations_from_title_nltk']}
```

In [ ]:
```
top_locations = get_top_orgs(text_body_dict)

print("NLTK Outputs: \n")
for key, locations in top_locations.items():
    print(f"{key}:")
    for loc, count in locations:
        print(f"{loc}: {count}")
    print()
```

```
NLTK Outputs:

Tweets:
Land: 1851
Russia: 454
New: 121
LAND: 117
Prince: 86
Car: 64
Meet: 41
New Land: 29
Great: 28
China: 25
South: 19
EU: 19
Latest: 17
Hi: 16
Check: 16
Mission: 16
Good: 16
Whilst: 15
Please: 13
Boss: 13

News Text:
New York City: 6932
New York: 4612
New: 4580
Prince: 4387
China: 2525
Palace: 2492
Amelia: 2071
South: 1736
United: 1720
Crown: 1519
North: 1385
West: 1289
Swift: 1238
Russia: 1203
San: 1097
Moss: 1052
Land: 920
Jordan: 842
Grand: 771
US: 709

News Title:
New: 143
Prince: 140
China: 92
Land: 82
Russia: 30
North York: 28
Jaguar: 25
Latest: 25
South: 24
US: 21
Covid: 21
EU: 20
Grenadier: 18
```

```
Electric: 18
Car: 17
Japan: 15
German: 15
Enjoy: 14
Queen: 14
Best: 13
```

Apply Sentence Segmentation with NLTK for extracting Locations

In [ ]:
```
def nltk_extract_locations_sentences(text):
    entities = []

    sentences = sent_tokenize(text)

    for sentence in sentences:
        # Tokenize each sentence into words and perform organization extractio
        for chunk in nltk.ne_chunk(nltk.pos_tag(word_tokenize(sentence)), bina
            if hasattr(chunk, 'label') and chunk.label() == 'GPE':
                entities.append(' '.join(c[0] for c in chunk))
    return entities
```

In [ ]:
```
tweets_df['organizations_nltk_sentences'] = tweets_df['tweets_clean'].parallel_
news_df['organizations_from_text_nltk_sentences'] = news_df['text_clean'].para
news_df['organizations_from_title_nltk_sentences'] = news_df['title_clean'].pa
```

```
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
44), Label(value='0 / 1444'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
VBox(children=(HBox(children=(IntProgress(value=0, description='0.00%', max=14
31), Label(value='0 / 1431'))), …
```

In [ ]:
```
text_body_dict = {"Tweets": tweets_df['organizations_nltk_sentences'],
                  "News Text": news_df['organizations_from_text_nltk_sentences
                  "News Title": news_df['organizations_from_title_nltk_sentenc
```

In [ ]:
```
top_locations = get_top_orgs(text_body_dict)

print("NLTK Outputs: \n")
for key, locations in top_locations.items():
    print(f"{key}:")
    for loc, count in locations:
        print(f"{loc}: {count}")
    print()
```

```
NLTK Outputs:

Tweets:
Land: 1851
Russia: 454
New: 121
LAND: 117
Prince: 86
Car: 64
Meet: 41
New Land: 29
Great: 28
China: 25
South: 19
EU: 19
Latest: 17
Hi: 16
Check: 16
Mission: 16
Good: 16
Whilst: 15
Please: 13
Boss: 13

News Text:
New York City: 6932
New York: 4612
New: 4580
Prince: 4387
China: 2525
Palace: 2492
Amelia: 2071
South: 1736
United: 1720
Crown: 1519
North: 1385
West: 1289
Swift: 1238
Russia: 1203
San: 1097
Moss: 1052
Land: 920
Jordan: 842
Grand: 771
US: 709

News Title:
New: 143
Prince: 140
China: 92
Land: 82
Russia: 30
North York: 28
Jaguar: 25
Latest: 25
South: 24
US: 21
Covid: 21
EU: 20
Grenadier: 18
```

```
Electric: 18
Car: 17
Japan: 15
German: 15
Enjoy: 14
Queen: 14
Best: 13
```

Conclusions:

Part 1: Main Organization/Company:

After extractinging entities separately from Tweets, Article Titles, and Article Texts, the top organization mentioned in the Tweets data is **Jaguar Land Rover** and top organization mentioned in the Articles is **Ford**. Both sentence and non-sentence segmentation across the NLTK and SpaCy packages returned similar results.

Part 2: Other Organizations/Companies:

Other companies recognized are Ford, Honda, Appled, Daily Mail, Jeep, and more

Part 3: Location of Events:

After extractinging entities separately from Tweets, Article Titles, and Article Texts, the most frequently mentioned locations (countries, states, cities, regions, etc.) are Russia, China, New York City, New York, the United States (US), LA, Turkey, Japan, Zimbabwe, Brunswick, and more.