Text Blob

In [2]:
```
#!pip install -U textblob
```

Dependencies

In [49]:
```
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer, PatternAnalyzer
```

In [18]:
```
text = 'Apple is a very succcessful company due to its visionary leadership team.
```

Default Method

In [19]:
```
blob = TextBlob(text)
blob.sentiment
```

Out[19]:
```
Sentiment(polarity=0.24625000000000002, subjectivity=0.45)
```

Naive Bayes

In [22]:
```
blob = TextBlob(text, analyzer=NaiveBayesAnalyzer())
blob.sentiment
```

Out[22]:
```
Sentiment(classification='pos', p_pos=0.8980227173699891, p_neg=0.1019772826300
1006)
```

Another example

In [179]:
```
text = 'There is a lockdown in Delhi due to covid19 pandemic'
```

In [180]:
```
blob = TextBlob(text)
blob.sentiment
```

Out[180]:
```
Sentiment(polarity=-0.125, subjectivity=0.375)
```

Naive Bayes is trained on Movie reviews

In [25]:
```
blob = TextBlob(text, analyzer=NaiveBayesAnalyzer())
blob.sentiment
```

Out[25]:
```
Sentiment(classification='pos', p_pos=0.6190774158665897, p_neg=0.3809225841334
109)
```

Default method is PatternAnalyzer

In [29]:
```
blob = TextBlob(text, analyzer=PatternAnalyzer())
blob.sentiment
```

Out[29]:
```
Sentiment(polarity=-0.125, subjectivity=0.375)
```

2. Word-Dictionary based

In [37]:
```
pos_words = []
neg_words = []

#Ignoring neutral, both, and other polarities
with open('data/subjectivity_clues_hltemnlp05/subjclueslen1-HLTEMNLP05.tff') as f
    for line in file:
        line_attrib = line.split()
        word = line_attrib[2].split('=')[1] #2nd column in the file
        polarity = line_attrib[-1].split('=')[1] #last column in the file
        if polarity =='positive':
            pos_words.append(word)
        elif polarity=='negative':
            neg_words.append(word)

print('Total positive words found: ',len(pos_words))
print('Total negative words found: ',len(neg_words))

#Write results to file for future use
with open('pos_words.txt', mode='wt', encoding='utf-8') as myfile:
    myfile.write('\n'.join(pos_words))
with open('neg_words.txt', mode='wt', encoding='utf-8') as myfile:
    myfile.write('\n'.join(neg_words))
```

```
Total positive words found:  2718
Total negative words found:  4911
```

Append your own domain specific words to positive or negative words

In [59]:
```
pos_word_add = ['lifted']
for term in pos_word_add:
    pos_words.append(term)

neg_word_add = ['coronavirus','covid','corona','covid19','pandemic','lockdown']
for term in neg_word_add:
    neg_words.append(term)

print('Total positive words found: ',len(pos_words))
print('Total negative words found: ',len(neg_words))
```

```
Total positive words found:  2720
Total negative words found:  4929
```

Function to calculate sentiment based on word-dictionary

In [63]:
```
import nltk
def calc_sentiment_based_on_word_dict(text):
    sentiment_score = 0
    words = nltk.word_tokenize(text)
    for word in words:
        if word in pos_words:
            print('pos:',word)
            sentiment_score=sentiment_score+1
        if word in neg_words:
            print('neg:',word)
            sentiment_score=sentiment_score-1
    return sentiment_score/len(words)
```

Example 1

In [39]:
```
text = 'Apple is a very succcessful company due to its visionary leadership team.
sentiment = calc_sentiment_based_on_word_dict(text)
print('The sentiment score of this text is: {:.2f}'.format(sentiment) )
```

```
The sentiment score of this text is: 0.11
```

Example2

In [42]:
```
text = 'There is a lockdown in Chicago due to covid19 pandemic'
sentiment = calc_sentiment_based_on_word_dict(text)
print('The sentiment score of this text is: {:.2f}'.format(sentiment) )
```

```
The sentiment score of this text is: -0.30
```

Example3

In [65]:
```
text = 'The lockdown is lifted.'
sentiment = calc_sentiment_based_on_word_dict(text)
print('The sentiment score of this text is: {:.2f}'.format(sentiment) )
```

```
neg: lockdown
pos: lifted
The sentiment score of this text is: 0.00
```

In [66]:
```
blob = TextBlob(text, analyzer=PatternAnalyzer())
blob.sentiment
```

Out[66]:
```
Sentiment(polarity=0.0, subjectivity=0.0)
```

3. Custom Trained Classifier

Read annotated data file

In [154]:
```
import pandas as pd
pd.set_option('display.max_colwidth', -1)
news_df = pd.read_csv('data/POS_NEG_NEWS.csv')
print(news_df.shape)
news_df.head(3)
```

```
(30, 4)
```

Out[154]:

	id	text	class	
0	1	Target stores announced yesterday that it will permanently raise its starting wage for U.S. team members to $15 per hour. Additionally, the company will give a one-time recognition bonus of $200.00 to its frontline store and distribution center hourly workers for their efforts during the coronavirus pandemic.\n\nAdditionally, starting this week, Target is also offering a new healthcare benefit that provides access to free virtual healthcare visits through the end of the year, regardless of whether team members are on a Target health insurance plan. The company also announced additional extensions of a 30-day paid leave for vulnerable team members, as well as free backup care for family members.\n\n“In the best of times, our team brings incredible energy and empathy to our work, and in harder times they bring those qualities plus extraordinary resilience and agility to keep,” said Target CEO Brian Cornell in a statement. “Everything we aspire to do and be as a company builds on the central role our team members play.”	POS	http
1	2	He grew up in a gentrified neighborhood with only a few African-American residents, but after confessing that he’s always felt uneasy walking alone on its streets, his faith in humanity was restored by dozens of neighbors who walked with him in solidarity.\n\nThe 29-year-old has lived in the 12 South neighborhood of Nashville his whole life, after his family moved there 54 years ago. But over the course of his life, Shawn Dromgoole has seen the neighborhood change dramatically.\n\nRising home values priced out many of the black families who lived there. As they moved out to find more affordable housing, new families moved in, most of them white and more financially well-off—leaving Shawn feeling out of place in his hometown.	POS	http
2	3	This teenage entrepreneur is responsible for inventing a simple, yet brilliant way to help curb coronavirus infections.\n\n15-year-old Max Melia designed and developed a wearable wristband which alerts users whenever they are about to touch their face\n\nHe first came up with the brilliant idea two years ago as a means of reducing the spread of the cold and flu. After both of Max’s parents contracted COVID-19 four months ago, however, he threw all his efforts into developing a working prototype.\n“Watching this pandemic unfold on the news, it was clear the devastating effect it was having on people lives’ across the world,” says Max. “However, it wasn’t until I saw the severity of the virus first-hand, when my parents both contracted COVID-19, did I truly appreciate just what we were dealing with.”\n\nThe tech-savvy English teen has since produced the VybPro, which warns users whenever they are about to subconsciously touch their face.\n\nThe gesture has been recognized as one of the key ways that coronavirus can be transmitted, making it a potentially live-saving invention.	POS	htt

Dependencies

In [71]:
```
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_sc
from sklearn import metrics
from sklearn.feature_extraction.text import CountVectorizer, HashingVectorizer,
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import precision_recall_fscore_support as score
```

Prepare Predictor and Response variables

In [155]:
```
X = news_df['text']
y = news_df['class'].map({'POS':0, 'NEG':1})
print(X.shape)
print(y.shape)
```

```
(30,)
(30,)
```

Create train/test split with stratification

In [96]:
```
# split X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.33, random_s
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(20,)
(10,)
(20,)
(10,)
```

Vectorize data using CountVectorizer

In [98]:
```
vectorizer = CountVectorizer(analyzer='word',
                             max_df=0.9,
                             max_features=1000,
                             ngram_range=(1, 3))

X_train_dtm = vectorizer.fit_transform(X_train)
X_test_dtm = vectorizer.transform(X_test)
```

Function to fit three different ML models

In [99]:
```
models = [MultinomialNB,LogisticRegression,SGDClassifier]

def fit_models(X_train,y_train,X_test,y_test,model):
    model = model()
    model.fit(X_train,y_train)
    return model
```

Model 1 - Naive Bayes

In [132]:
```
#Naive Bayes
%time model = fit_models(X_train_dtm,y_train,X_test_dtm,y_test,models[0])
# make class predictions for X_test_dtm
y_pred_class = model.predict(X_test_dtm)

# calculate accuracy of class predictions
print('Accuracy Score:',metrics.accuracy_score(y_test, y_pred_class))


# calculate precision and recall
print(classification_report(y_test, y_pred_class))


# calculate the confusion matrix
print(metrics.confusion_matrix(y_test, y_pred_class))
```

```
Wall time: 1 ms
Accuracy Score: 0.9
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       1.00      0.80      0.89         5

    accuracy                           0.90        10
   macro avg       0.92      0.90      0.90        10
weighted avg       0.92      0.90      0.90        10

[[5 0]
 [1 4]]
```

Model 2 - Logistic Regression

In [126]:
```
#Logistic Regression Bayes
%time model = fit_models(X_train_dtm,y_train,X_test_dtm,y_test,models[1])
# make class predictions for X_test_dtm
y_pred_class = model.predict(X_test_dtm)

# calculate accuracy of class predictions
print('Accuracy Score:',metrics.accuracy_score(y_test, y_pred_class))

# calculate precision and recall
print(classification_report(y_test, y_pred_class))

# calculate the confusion matrix
print(metrics.confusion_matrix(y_test, y_pred_class))
```

```
Wall time: 2 ms
Accuracy Score: 0.9
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       1.00      0.80      0.89         5

    accuracy                           0.90        10
   macro avg       0.92      0.90      0.90        10
weighted avg       0.92      0.90      0.90        10

[[5 0]
 [1 4]]
```

Model 3 - SVM

In [127]:
```
#SVM
%time model = fit_models(X_train_dtm,y_train,X_test_dtm,y_test,models[2])
# make class predictions for X_test_dtm
y_pred_class = model.predict(X_test_dtm)

# calculate accuracy of class predictions
print('Accuracy Score:',metrics.accuracy_score(y_test, y_pred_class))

# calculate precision and recall
print(classification_report(y_test, y_pred_class))

# calculate the confusion matrix
print(metrics.confusion_matrix(y_test, y_pred_class))
countVect_SVM_CM = metrics.confusion_matrix(y_test, y_pred_class)
```

```
Wall time: 1e+03 µs
Accuracy Score: 0.8
              precision    recall  f1-score   support

           0       1.00      0.60      0.75         5
           1       0.71      1.00      0.83         5

    accuracy                           0.80        10
   macro avg       0.86      0.80      0.79        10
weighted avg       0.86      0.80      0.79        10

[[3 2]
 [0 5]]
```

Unseen Test example 1

In [133]:
```
text = ["Today was a grim day for New Delhi where there happend communal riots ar
val_vect = vectorizer.transform(text)
%time y_pred_class = model.predict(val_vect)
print(y_pred_class)
```

```
Wall time: 1e+03 µs
[1]
```

Unseen Test example 2

In [135]:
```
text = ["I am happy that university has given me scholarship for my Masters degre
val_vect = vectorizer.transform(text)
%time y_pred_class = model.predict(val_vect)
print(y_pred_class)
```

```
Wall time: 0 ns
[0]
```

Unseen Article 1

In [136]:
```
#https://www.mirror.co.uk/news/uk-news/teen-accused-pc-andrew-harper-22247350
text =["""A teenager accused of murdering PC Andrew Harper said he "didn't give a
a court heard.

Henry Long, 19, allegedly claimed he had been watching DVDs, including Fast and t

He allegedly said "I don't give a f*** about any of this", as he was charged with

The trio allegedly got rid of their mobile phones by the time police arrived at t

Long remarked "look at me, do I look like a murderer?", as officers detained him,
horrific killing. He later admitted manslaughter ahead of the murder trial.

Mr Laidlaw said: "Police began, in numbers, to enter the Four Houses Corner site

"Henry Long along with Albert Bowers was found in a Fendt Diamond caravan in Plot

"Neither man was found in possession of a mobile phone, and the police could not

"What becomes clear is that both young men had managed to get rid of their teleph
travelling community who were not arrested."""]
```

In [137]:
```
val_vect = vectorizer.transform(text)
%time y_pred_class = model.predict_proba(val_vect)
print(y_pred_class)
```

```
Wall time: 1 ms
[[8.05630867e-13 1.00000000e+00]]
```

Unseen Article 2

In [138]:
```
#https://www.ndtv.com/world-news/dubai-supermarket-delivers-mango-in-lamborghini-
text=["""A supermarket in Dubai is delivering Mangoes at the doorsteps of its cus
"The king should travel like a king," says Mohammad Jehanzeb, the managing direct

Mohammad Jehanzeb delivers the pulpy fruit himself and also takes the customers c

In order to avail the offer rolled out on the Facebook page of the famous superma

"The idea is to put a smile on people's faces and make them feel special," says M

The videos of delighted mango lovers, taking a joy ride in the car, are doing rou
A supermarket in Dubai is delivering Mangoes at the doorsteps of its customers ir

"Each order takes about an hour. We do about 7-8 home deliveries a day but are hc

Arshad Khan, who hails from India's Lucknow, said that his children were excited

"For someone who hails from Lucknow -- the land of the famous dussheri and landga
experience of seeing them come to my place in a luxury supercar," Gulf News quote

"It was quite exhilarating and I must confess that the mangoes were as delicious
```

In [139]:
```
val_vect = vectorizer.transform(text)
%time y_pred_class = model.predict_proba(val_vect)
print(y_pred_class)
```

```
Wall time: 1e+03 µs
[[0.99533839 0.00466161]]
```

Use Snorkel to automate text labelling

4. Named Entity based Sentiment Analysis (Targetted)

Spacy based Named Entity recognition

In [158]:
```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_sm")

def spacy_ner(text):
    text = text.replace('\n', ' ')
    doc = nlp(text)
    entities = []
    labels = []
    position_start = []
    position_end = []

    for ent in doc.ents:
        if ent.label_ in ['PERSON','ORG','GPE']:
            entities.append(ent)
            labels.append(ent.label_)
    return entities,labels

def fit_ner(df):
    """The dataframe should have a column named 'text'"""
    print('Fitting Spacy NER model...')
    ner = df['text'].apply(spacy_ner)
    ner_org = {}
    ner_per = {}
    ner_gpe = {}

    for x in ner:
        #print(list(x))
        for entity, label in zip(x[0],x[1]):
            #print(type(entity.text))
            if label =='ORG':
                ner_org[entity.text] = ner_org.get(entity.text,0) + 1
            elif label=='PERSON':
                ner_per[entity.text] = ner_per.get(entity.text,0) + 1
            else:
                ner_gpe[entity.text] = ner_gpe.get(entity.text,0) + 1

    return {'ORG':ner_org,'PER':ner_per,'GPE':ner_gpe}
```

In [159]:
```
named_entities = fit_ner(news_df)
```

```
Fitting Spacy NER model...
```

Organization Named Entities

In [163]:
```
named_entities['ORG']
```

```
 'Sooraj': 1,
 'Kollam Rural Police': 1,
 'Duke University': 2,
 'the National Climate Assessment': 1,
 'Complications': 1,
 'Nasdaq Composite': 1,
 'Nasdaq': 2,
 'Dow': 1,
 'MUFG': 1,
 'United Airlines': 1,
 'Delta': 1,
 'Bleakley Advisory Group': 1,
 'Carnival': 1,
 'Royal Caribbean': 1,
 'Disney': 1,
 'SFPD': 1,
 'The Santa Fe Police Department': 1,


 'SFR': 1,
```

Person Named Entities

In [164]:
```
named_entities['PER']
```

Out[164]:
```
{'Brian Cornell': 1,
 'Shawn Dromgoole': 1,
 'Shawn': 1,
 'Max Melia': 1,
 'Max': 2,
 'Rahman Daiyan': 1,
 'Emma Lovell': 1,
 'Rose Amal': 1,
 'Wolfgang Palme': 1,
 '-11º Celsius': 1,
 'Kamryn Johnson': 1,
 'Ron Johnson': 1,
 'George Floyd': 3,
 'Libertarian Justin Amash': 1,
 'Tim Scott': 1,
 'Scott': 1,
 'Madisen Hallberg': 1,
 'Emmanuel Henreid': 1,
 'Hallberg': 1,
 'GNN': 1,
 'Onry': 1,
 'Peter Horby': 1,
 'Martin Landry': 1,
 'Cher': 1,
 'Kaavan': 1,
 'Jody McDowell': 1,
 'McDowell': 2,
 'Tamil Nadu': 2,
 'Jayaraj': 6,
 'Pennis': 4,
 'Arun Balagopalan': 2,
 'Jessie D. Gregory': 1,
 'Stephen Nolan': 1,
 'Kevin Deboni': 1,
 'Deboni': 6,
 'Caleb White': 1,
 'Irving Federman': 1,
 'Federman': 1,
 'David Brown': 1,
 'Brown': 1,
 'Stroger': 1,
 'Amaria J. Jones': 1,
 'E Palaniswami': 1,
 'Kanimozhi': 1,
 'Henri Tiphagne': 1,
 'Jeff Beamish': 2,
 'Wayne Deutsch': 1,
 'Deutsch': 3,
 'W. Illinois Highway': 1,
 'Jasean Francis': 1,
 'Charles Riley': 1,
 'Puma': 1,
 'Subhas': 1,
 'Lalita Dhandhania': 1,
```

```
 'Amit': 3,
 'Shilpi': 1,
 'Bengaluru': 1,
 'Kolkata': 1,
 'Mrs Shilpi Agarwal': 1,
 'Shilpi Agarwal': 1,
 'Prima': 1,
 'Uthra': 2,
 'Sooraj': 1,
 'Hari Sankar': 1,
 'Heat': 1,
 'Chris Rupkey': 1,
 'Peter Boockvar': 1,
 'Retailer Gap': 1,
 'Paul Joye': 1,
 'Joye': 1,
 'Bajit Singh': 1}
```

Geopolitical Named Entities

In [165]:
```
named_entities['GPE']
```

Out[165]:
```
{'U.S.': 7,
 'Nashville': 1,
 'Lego': 1,
 'Palme': 2,
 'Austria': 1,
 'Minneapolis': 3,
 'Minnesota': 1,
 'Colorado': 1,
 'Kentucky': 1,
 'Oregon': 1,
 'UK': 2,
 'Horby': 1,
 'Islamabad': 1,
 'Kaavan': 1,
 'Sri Lanka': 1,
 'Pakistan': 1,
 'America': 4,
 'Yellowstone': 2,
 'improv': 1,
 'Quebec': 1,
 'Tennessee': 1,
 'Baltimore': 1,
 'Maryland': 1,
 'Tuticorin': 4,
 'Chicago': 9,
 'St. Bernard’s': 1,
 'Bridgeport': 1,
 'Schaumburg': 2,
 'Gresham': 1,
 'Cook County': 1,
 'Nolan': 11,
 'Las Vegas': 1,
 'Austin': 2,
 'North LeClaire Avenue': 1,
 'the United States': 2,
 'the Southern United States': 2,
 'Florida': 6,
 'Texas': 9,
 'Sonoma': 2,
 'Louisiana': 2,
 'Little Village': 1,
 'New Lenox': 1,
 'Will County': 2,
 'Orland Park': 1,
 'South Chicago': 1,
 'Bengaluru': 2,
 'Kolkata': 3,
 'Whitefield': 2,
 'India': 1,
 'Sooraj': 1,
 'Kerala': 2,
 'California': 1,
 'New York': 2,
 'New Jersey': 2,
```

```
 'Connecticut': 2,
 'Arizona': 1,
 'Orlando': 1,
 'Plaza': 1}
```

Sentiment Analysis based on top named entities

Targetted by finding sentences containing the named entities and performing sentiment analysis only on those sentences one by one

In [166]:
```
from textblob import TextBlob
import nltk

def get_keyword_sentences(text,keyword):
    """Extract sentences containing the Named Entity/Keyword"""
    text = text.replace('\n', ' ')
    sentences = nltk.sent_tokenize(text)

    sent_of_interest = []
    for sent in sentences:
        if keyword in sent.lower():
            sent_of_interest.append(sent)
    return sent_of_interest if len(sent_of_interest)>0 else False


def get_sentiment(list_of_sent):
    """ Extract sentiment from a sentence using TextBlob Pattern Analyzer"""
    sentiment = 0
    count = 0

    if list_of_sent !=False:
        for sent in list_of_sent:
            blob = TextBlob(sent)
            sentiment += blob.sentiment.polarity
            count += len(sent)
    return sentiment/count if count!=0 else 0

def extract_sentiment(df,keywords,top=10):
    """df: data for the cluster, keywords: ner for that cluster"""
    print('Extracting Sentiments using TextBlob...')
    keywords = sorted(keywords.items(),key=lambda x: x[1],reverse=True)
    sentiment_dict = {}
    for keyword,count in keywords[:top]:
        df['sentences'] = df['text'].apply(get_keyword_sentences,keyword=keyword.
        keyword_sentiment = df['sentences'].apply(get_sentiment).sum()
        sentiment_dict[keyword] = keyword_sentiment
    return sentiment_dict
```

Top 30 Organizations according to their sentiments

In [174]:
```
keywords = named_entities['ORG']
sentiment_result_org = extract_sentiment(news_df,keywords,top=30)
sentiment_result_org
sorted(sentiment_result_org.items(),key=lambda x:x[1], reverse=True)
```

```
Extracting Sentiments using TextBlob...
```

Out[174]:
```
[('the US House of Representatives', 0.0019907100199071),
 ('the Conservation Fund', 0.0019907100199071),
 ('Kamryn', 0.0018972048066875655),
 ('the National Parks Service', 0.0017921146953405018),
 ('the Land and Water Conservation Fund', 0.0013941535496305815),
 ('the Natural Resources Management Act', 0.0013941535496305815),
 ('Target', 0.0012131536833410769),
 ('the Great American Outdoors Act', 0.0011459492888064317),
 ('Capitol Hill', 0.000883481597767312),
 ('UNSW Sydney', 0.0008679023242130039),
 ('Chemical', 0.0008595478595478595),
 ('UNSW', 0.0007272727272727272),
 ('Senate', 0.0007037413380270522),
 ('Advanced Energy Materials', 0.0006195786864931846),
 ('School of Chemical Engineering', 0.0006195786864931846),
 ('FDA', 0.0005633802816901409),
 ('Whole Foods', 0.0004136029411764706),
 ('DUI', 0.0003423736143555005),
 ('Duke University', 0.0002017213555675094),
 ('the Forest Service', 0.0),
 ('the Bureau of Indian Education', 0.0),
 ('the Bureau of Land Management', 0.0),
 ('the Fish and Wildlife Service', 0.0),
 ('Persis', 0.0),
 ('Santhankulam', 0.0),
 ('FBI', -0.0003755868544600939),
 ('the National Oceanic & Atmospheric Administration', -0.0005025125628140704),
 ('Amit Agarwal', -0.0006304347826086958),
 ('DCP', -0.0010416666666666667),
 ('Nasdaq', -0.0042763157894736845)]
```

Top 30 Persons according to their sentiments

In [175]:
```
keywords = named_entities['PER']
sentiment_result_org = extract_sentiment(news_df,keywords,top=30)
sentiment_result_org
sorted(sentiment_result_org.items(),key=lambda x:x[1], reverse=True)
```

```
Extracting Sentiments using TextBlob...
```

Out[175]:
```
[('GNN', 0.004569909563803812),
 ('McDowell', 0.002396953405017921),
 ('Brian Cornell', 0.002359882005899705),
 ('Hallberg', 0.0014084507042253522),
 ('Ron Johnson', 0.0013736263736263737),
 ('Shawn', 0.001013856032443393),
 ('Rose Amal', 0.0009461358313817331),
 ('Max Melia', 0.0006799163179916319),
 ('Rahman Daiyan', 0.0006195786864931846),
 ('Emma Lovell', 0.0006195786864931846),
 ('George Floyd', 0.0005177020708082832),
 ('Tim Scott', 0.0003012048192771084),
 ('Scott', 0.00023961661341853033),
 ('Wolfgang Palme', 5.8685446009389656e-05),
 ('Deutsch', 5.835667600373482e-05),
 ('Jayaraj', 0.0),
 ('Arun Balagopalan', 0.0),
 ('Jeff Beamish', 0.0),
 ('Shawn Dromgoole', 0.0),
 ('Madisen Hallberg', 0.0),
 ('Emmanuel Henreid', 0.0),
 ('Pennis', -0.00021367521367521362),
 ('Uthra', -0.0003479236812570146),
 ('Deboni', -0.0003628628628628629),
 ('Max', -0.0005604288499025341),
 ('Libertarian Justin Amash', -0.0005889281507656066),
 ('Amit', -0.001074475065616798),
 ('-11º Celsius', -0.001097560975609756),
 ('Tamil Nadu', -0.002578268876611418),
 ('Kamryn Johnson', -0.00641025641025641)]
```

Top 30 Geopolitical Entities according to their sentiments

In [176]:
```
keywords = named_entities['GPE']
sentiment_result_org = extract_sentiment(news_df,keywords,top=30)
sentiment_result_org
sorted(sentiment_result_org.items(),key=lambda x:x[1], reverse=True)
```

```
Extracting Sentiments using TextBlob...
```

Out[176]:
```
[('U.S.', 0.007197756333989664),
 ('Lego', 0.003282828282828283),
 ('Florida', 0.0019480167692380369),
 ('Yellowstone', 0.0017921146953405018),
 ('Colorado', 0.0017676767676767676),
 ('Nashville', 0.0015873015873015873),
 ('America', 0.0013372595005477282),
 ('New York', 0.00115112160566706),
 ('New Jersey', 0.00115112160566706),
 ('Connecticut', 0.00115112160566706),
 ('Will County', 0.0008675534991324465),
 ('Texas', 0.0008449765331576672),
 ('Schaumburg', 0.0007514450867052024),
 ('Kerala', 0.0006862745098039215),
 ('Minnesota', 0.0004947916666666666),
 ('Austria', 0.00040345019476905957),
 ('Nolan', 0.00020554291972032683),
 ('UK', 6.731275341697178e-05),
 ('Sonoma', 0.0),
 ('Louisiana', 0.0),
 ('Chicago', -0.00018155486893953014),
 ('Palme', -0.00034435261707988976),
 ('Austin', -0.0003947368421052632),
 ('the United States', -0.0005025125628140704),
 ('the Southern United States', -0.0005025125628140704),
 ('Kolkata', -0.001114900314795383),
 ('Minneapolis', -0.0011658363526570051),
 ('Whitefield', -0.0013840830449826991),
 ('Bengaluru', -0.0017738359201773836),
 ('Tuticorin', -0.0020131938736589897)]
```

Application: Analyzing Sentiment in a book

In [181]:
```
book = open('data/book_cab_and_goose.txt',encoding="utf8").read()
```

In [182]:
```
blob = TextBlob(book, analyzer=NaiveBayesAnalyzer())
blob.sentiment
```

Out[182]:
```
Sentiment(classification='pos', p_pos=1.0, p_neg=6.5736053249909995e-257)
```

In [183]:
```
blob = TextBlob(book)
blob.sentiment
```

Out[183]:
```
Sentiment(polarity=0.10141585513512091, subjectivity=0.4815873085413998)
```

Sentiment per sentence

In [184]:
```
blob = TextBlob(book)

polarity = []
subjectivity = []
sentences = []
PatternAnalyzerDF = pd.DataFrame(columns=['sentence', 'polarity', 'subjectivity']

for sentence in blob.sentences:
    polarity.append(sentence.sentiment.polarity)
    subjectivity.append(sentence.sentiment.subjectivity)
    sentences.append(str(sentence.raw))

PatternAnalyzerDF['sentence'] = sentences
PatternAnalyzerDF['polarity'] = polarity
PatternAnalyzerDF['subjectivity'] = subjectivity

PatternAnalyzerDF['sentence'] = PatternAnalyzerDF['sentence'].str.replace('\n',
```

In [185]:
```
PatternAnalyzerDF.head(10)
```

Out[185]:

	sentence	polarity	subjectivity
0	The Project Gutenberg eBook, Cab and Caboose, by Kirk Munroe This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever.	0.000000	0.000000
1	You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org Title: Cab and Caboose The Story of a Railroad Boy Author: Kirk Munroe Release Date: September 4, 2007 [eBook #22497] Language: English ***START OF THE PROJECT GUTENBERG EBOOK CAB AND CABOOSE*** E-text prepared by Mark C. Orton, Linda McKeown, Anne Storer, and the Project Gutenberg Online Distributed Proofreading Team (http://www.pgdp.net) Note: Project Gutenberg also has an HTML version of this file which includes the original illustrations.	0.187500	0.375000
2	See 22497-h.htm or 22497-h.zip: (http://www.gutenberg.net/dirs/2/2/4/9/22497/22497-h/22497-h.htm) or (http://www.gutenberg.net/dirs/2/2/4/9/22497/22497-h.zip) CAB AND CABOOSE The Story of a Railroad Boy by KIRK MUNROE OFFICERS OF THE NATIONAL COUNCIL Honorary President, THE HON.	-0.750000	1.000000
3	WOODROW WILSON Honorary Vice-President, HON.	0.000000	0.000000
4	WILLIAM H. TAFT Honorary Vice-President, COLONEL THEODORE ROOSEVELT President, COLIN H. LIVINGSTONE, Washington, D. C. Vice-President, B. L. DULANEY, Bristol, Tenn.	0.000000	0.000000
5	Vice-President, MILTON A. McRAE, Detroit.	0.000000	0.000000
6	Mich.	0.000000	0.000000
7	Vice-President, DAVID STARR JORDAN, Stanford University, Cal.	0.000000	0.000000
8	Vice-President, F. L. SEELY, Asheville, N. C. Vice-President, A. STAMFORD WHITE, Chicago, Ill. Chief Scout, ERNEST THOMPSON SETON, Greenwich, Connecticut National Scout Commissioner, DANIEL CARTER BEARD, Flushing, N. Y.	0.000000	0.000000
9	NATIONAL HEADQUARTERS BOY SCOUTS OF AMERICA THE FIFTH AVENUE BUILDING, 200 FIFTH AVENUE TELEPHONE GRAMERCY 545 NEW YORK CITY FINANCE COMMITTEE John Sherman Hoyt, Chairman August Belmont George D. Pratt Mortimer L. Schiff H. Rogers Winthrop GEORGE D. PRATT, Treasurer JAMES E. WEST, Chief Scout Executive ADDITIONAL MEMBERS OF THE EXECUTIVE BOARD Ernest P. Bicknell Robert Garrett Lee F. Hanmer John Sherman Hoyt Charles C. Jackson Prof. Jeremiah W. Jenks William D. Murray Dr. Charles P. Neill George D. Porter Frank Presbrey Edgar M. Robinson Mortimer L. Schiff Lorillard Spencer Seth Sprague Terry July 31st, 1913.	0.136364	0.454545

Plotting sentiment variation in the book as the story is unveiled

In [188]:
```
import matplotlib.pyplot as plt
PatternAnalyzerDF.sort_index(inplace=True)
sentiment_top_df = PatternAnalyzerDF.head(n=200)
pd.set_option('display.max_colwidth', 200)
```

In [194]:
```
plt.figure().set_size_inches(20, 10)
plt.plot(sentiment_top_df['polarity'])
plt.xlabel('Sentence')
plt.ylabel('Polarity')
plt.show()
```