sentiment_analysis

July 26, 2019

1 Crisis Sentiment Analysis

This activity is a mini-project where students will create a data visualization dashboard, they have to analyze sentiment and tone about the news related to the financial crisis of 2008 that where published along the last month. Students will retrieve the news articles from the News API; by default, the developer account gives access to news articles up to a month old.

In this activity, students will use their new sentiment analysis skills, in combination to some of the skills they already master such as: Pandas, Pyviz, Plotly Express and PyViz Panel.

This Jupyter notebook is a sandbox where students will conduct the sentiment analysis tasks and charts creation before assembling the dashboard.

[1]:
```
# Initial imports
import os
from path import Path
import pandas as pd
import numpy as np
import hvplot.pandas
import nltk
from wordcloud import WordCloud
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from newsapi import NewsApiClient
from ibm_watson import ToneAnalyzerV3
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
import plotly.express as px
import matplotlib.pyplot as plt
import matplotlib as mpl
import panel as pn

plt.style.use("seaborn-whitegrid")
pn.extension("plotly")
```

1.1 Instructions

1.1.1 Fetching the Latests News Metions About the Crisis of 2008

Using the News API, get all the news in English about the financial crisis of 2008 using the keywords `"financial AND crisis AND 2008"` in the q parameter. Define a `page_size=100` to have at least 100 news articles to analyze.

[2]:
```
# Retrieve the News API key
news_api = os.getenv("news_api")
```

[3]:
```
# Create the newsapi client
newsapi = NewsApiClient(api_key=news_api)
```

[4]:
```
# Fetch the news articles about the financial crisis on 2008 in English
crisis_news_en = newsapi.get_everything(
    q="financial AND crisis AND 2008", language="en", page_size=100
)

# Show the total number of news
crisis_news_en["totalResults"]
```

[4]:
```
2153
```

1.1.2 Creating a VADER Sentiment Scoring Function

Use the VADER sentiment scoring function from `NLTK` to score the sentiment polarity of the 100 news you fetched. Just for convenience, start downloading the `vader_lexicon` in order to initialize the VADER sentiment analyzer.

[5]:
```
# Download/Update the VADER Lexicon
nltk.download("vader_lexicon")
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/josearturomorasoto/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

[5]:
```
True
```

[6]:
```
# Initialize the VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()
```

In order to score the VADER sentiment, create a function named `get_sentiment_scores(text, date, source, url)` that will receive four parameters.

- `text` is the text whose sentiment will be scored.
- `date` the date the news article was published using the format `YYYY-MM-DD`.
- `source` is the name of the news article's source.
- `url` is the URL that points to the article.

The `get_sentiment_score()` function should return a Python dictionary with the scoring results. This dictionary is going to be used in the next section to create a DataFrame; the structure of the dictionary is the following:

- `date` the date passed as parameter to the function.
- `text` the text passed a parameter to the function.
- `source` the source passed as parameter to the function.

- `url` the URL passed as parameter to the function.
- `compound` the compound score from the VADER sentiment analyzer.
- `pos` the positive score from the VADER sentiment analyzer.
- `neu` the neutral score from the VADER sentiment analyzer.
- `neg` the negative score from the VADER sentiment analyzer.
- `normalized` the normalized scored based on the `compound` results. Its value should be `1` for positive sentiment, `-1` for negative sentiment, and `0` for neutral sentiment.

This is an example of the function's return value:

```
{'date': '2019-06-24',
'text': '\nMore than a decade since the global economic meltdown of 2008
    devastated lives across the world, no one who caused the crisis has
    been held responsible.\n\n"The 2008 financial crisis displayed what
    the world now identifies as financial contagion," says Philip J Baker,
    the former managing partner of a US-based \nhedge fund that collapsed
    during the financial crisis.\n\nDespite this, "zero Wall Street chief
    executives have been to prison, even though there is today absolutely
    no doubt that Wall Street executives and politicians \nwere complicit
    in creating the crisis," he says. \n\nBaker was among the few
    relatively smaller players imprisoned for the part they played.\n\n
    In July 2009, he was arrested in  Germany and extradited to the
    United States where he faced federal court on charges of fraud and
    financial crimes.\n\nHe pled guilty and was sentenced to 20 years
    in prison for costing some 900 investors about $294mn worldwide.
    He served eight years in jail and is now on \nparole and advocates
    against financial crime.\n',
'source': 'aljazeera',
'url': 'https://www.aljazeera.com/programmes/specialseries/2019/06/men-stole-world-2008-financ
'compound': -0.9911,
'pos': 0.048,
'neu': 0.699,
'neg': 0.254,
'normalized': -1}
```

[7]:
```python
# Define a function to get the sentiment scores
def get_sentiment_scores(text, date, source, url):
    sentiment_scores = {}

    # Sentiment scoring with VADER
    text_sentiment = analyzer.polarity_scores(text)
    sentiment_scores["date"] = date
    sentiment_scores["text"] = text
    sentiment_scores["source"] = source
    sentiment_scores["url"] = url
    sentiment_scores["compound"] = text_sentiment["compound"]
    sentiment_scores["pos"] = text_sentiment["pos"]
    sentiment_scores["neu"] = text_sentiment["neu"]
    sentiment_scores["neg"] = text_sentiment["neg"]
    if text_sentiment["compound"] >= 0.05:  # Positive
        sentiment_scores["normalized"] = 1
    elif text_sentiment["compound"] <= -0.05:  # Negative
        sentiment_scores["normalized"] = -1
    else:
        sentiment_scores["normalized"] = 0  # Neutral

    return sentiment_scores
```

1.1.3 Creating the News Articles' Sentiments DataFrame

In this section you have to create a DataFrame that is going to be used to plot the sentiment analysis results. Using a `for-loop`, iterate across all the news articles you fetched to create the DataFrame structure; define an empty list to append the sentiment scoring results for each news article and create the DataFrame using the list as data source.

Once you create the DataFrame do the following:

- Sort the DataFrame rows by the `date` column.
- Define the `date` column as the DataFrame index.
- Save the DataFrame as a CSV file in order to use it on the sentiment analysis dashboard creation.

[8]:
```
# Empty list to store the DataFrame structure
sentiments_data = []

# Loop through all the news articles
for article in crisis_news_en["articles"]:
    try:
        # Get sentiment scoring using the get_sentiment_score() function
        sentiments_data.append(
            get_sentiment_scores(
                article["content"],
                article["publishedAt"][:10],
                article["source"]["name"],
                article["url"],
            )
        )

    except AttributeError:
        pass

# Create a DataFrame with the news articles' data and their sentiment scoring␣
 ↪results
crisis_news_df = pd.DataFrame(sentiments_data)

# Sort the DataFrame rows by date
crisis_news_df = crisis_news_df.sort_values(by="date")
```

```
# Define the date column as the DataFrame's index
crisis_news_df.set_index("date", inplace=True)
crisis_news_df.head()
```

[8]:
```
            compound    neg    neu  normalized    pos           source  \
date
2019-06-25   -0.1027  0.038  0.962          -1  0.000       Forbes.com
2019-06-25   -0.8625  0.206  0.794          -1  0.000         Politico
2019-06-25    0.1027  0.055  0.881           1  0.064        Slate.com
2019-06-26   -0.6705  0.111  0.889          -1  0.000  Fastcompany.com
2019-06-26    0.0000  0.000  1.000           0  0.000         Digg.com

                                                         text  \
date
2019-06-25  <ul><li>Share to facebook</li><li>Share to twi...
2019-06-25  Michael Kruse is a senior staff writer for Pol...
2019-06-25  Andrew Yang didnt set out to become a test pre...
2019-06-26  For the latter half of 2016, the Standing Rock...
2019-06-26  We're halfway into 2019 and so far this year w...

                                                          url
date
2019-06-25  https://www.forbes.com/sites/mayrarodriguezval...
2019-06-25  https://www.politico.com/magazine/story/2019/0...
2019-06-25  https://slate.com/news-and-politics/2019/06/an...
2019-06-26  https://www.fastcompany.com/90364616/public-ba...
2019-06-26  http://digg.com/2019/20-biggest-bankruptcies-c...
```

[9]:
```
# Save the news articles DataFrame with VADER Sentiment scoring as a CSV file
file_path = Path("Data/news_vader.csv")
crisis_news_df.to_csv(file_path)
```

1.1.4 Creating the Average Sentiment Chart

Use `hvPlot` to create a two lines chart that compares the average `compound` and `normalized` sentiment scores along the last month.

[10]:
```
avg_sent_data = (
    crisis_news_df[["compound", "normalized"]].groupby(by=crisis_news_df.index).
 ↪mean()
)
avg_sent_data.head()
```

[10]:
```
            compound  normalized
date
2019-06-25 -0.287500   -0.333333
2019-06-26 -0.223500   -0.333333
2019-06-27  0.424033    0.666667
```

```
2019-06-28 -0.131675   -0.250000
2019-06-29  0.483250    1.000000
```

[11]:
```
avg_sent_plot = avg_sent_data.hvplot(
    title="Average Sentiment About the Economic Crisis of 2008 Last Month",␣
 ↪rot=90
)

avg_sent_plot
```

[11]:
```
:NdOverlay   [Variable]
   :Curve   [date]   (value)
```

1.1.5 Creating the Sentiment Distribution Chart

Based on the `normalized` sentiment score, create a bar chart using `hvPlot` that shows the number of negative, neutral and positive news articles. This chart represents the overall sentiment distribution.

[12]:
```
sentiment_chart_df = (
    crisis_news_df[["normalized", "text"]].groupby("normalized").count()
)
sentiment_chart_df.rename(
    index={-1: "Negative", 0: "Neutral", 1: "Positive"}, inplace=True
)
sentiment_chart_df
```

[12]:
```
            text
normalized
Negative      43
Neutral       11
Positive      46
```

[13]:
```
sentiment_bar_chart = sentiment_chart_df.hvplot.bar(
    xlabel="Sentiment",
    ylabel="Number of News",
    title="Overall Sentiment Distribution",
    color="text",
)

sentiment_bar_chart
```

[13]:
```
:Bars   [normalized]   (text)
```

1.1.6 Getting the Top 10 Positive and Negative News Articles

In this section you have to create two DataFrames, one with the top 10 positive news according to the `compound` score, and other with the top 10 negative news. Refer to the `hvplot.table()` documentation to create two tables presenting the following columns of these news articles:

- Date

- Source
- Text
- URL

[14]:
```
# Getting Top 10 positive news articles
pos_news = crisis_news_df.sort_values(by="compound", ascending=False)
pos_news = pos_news.head(10)
pos_news
```

[14]:
```
            compound    neg    neu  normalized    pos             source  \
date
2019-07-13    0.8831  0.073  0.689           1  0.238  The New York Times
2019-07-01    0.8481  0.027  0.769           1  0.204     Business Insider
2019-07-08    0.8360  0.000  0.809           1  0.191             Ozy.com
2019-07-08    0.8360  0.000  0.809           1  0.191             Ozy.com
2019-07-01    0.8240  0.000  0.807           1  0.193      Marketwatch.com
2019-07-08    0.7650  0.049  0.751           1  0.200      Marketwatch.com
2019-06-30    0.7184  0.000  0.878           1  0.122             Reuters
2019-07-18    0.7050  0.027  0.868           1  0.105          Daily Mail
2019-07-20    0.6956  0.149  0.595           1  0.256            The Hill
2019-07-24    0.6908  0.044  0.822           1  0.134          TechCrunch

                                                         text  \
date
2019-07-13  Let me start with what might seem like a trivi...
2019-07-01  The drafters of the Declaration of Independenc...
2019-07-08  This is an OZY Special Briefing, an extension ...
2019-07-08  This is an OZY Special Briefing, an extension ...
2019-07-01  LONDON (Project Syndicate) The U.S. economy ha...
2019-07-08  The hunt for yield is making parts of the U.S...
2019-06-30  LONDON (Reuters) - Deutsche Bank plans to hire...
2019-07-18  After years of stability, Britain is facing pr...
2019-07-20  How many nations have already embraced sociali...
2019-07-24  Its the absolute best economy the United State...

                                                          url
date
2019-07-13  https://www.nytimes.com/2019/07/13/opinion/gol...
2019-07-01  https://www.businessinsider.com/50-maps-that-e...
2019-07-08  https://www.ozy.com/need-to-know/special-brief...
2019-07-08  https://www.ozy.com/need-to-know/deutsche-bank...
2019-07-01  https://www.marketwatch.com/story/old-age-does...
2019-07-08  https://www.marketwatch.com/story/yield-gap-be...
2019-06-30  https://www.reuters.com/article/us-deutsche-ba...
2019-07-18  https://www.dailymail.co.uk/news/article-72627...
2019-07-20  https://thehill.com/opinion/finance/453913-has...
2019-07-24  http://techcrunch.com/2019/07/24/why-do-media-...
```

[15]:
```
# Create a table with hvplot
pos_news_table = pos_news.hvplot.table(
    columns=["date", "source", "text", "url"], width=500
)
pos_news_table
```

[15]:
```
:Table   [date,source,text,url]
```

[16]:
```
# Getting Top 10 negative news articles
neg_news = crisis_news_df.sort_values(by="compound", ascending=True)
neg_news = neg_news.head(10)
neg_news
```

[16]:
```
            compound    neg    neu  normalized    pos                 source  \
date
2019-07-22   -0.9343  0.293  0.707          -1  0.000             Medium.com
2019-07-10   -0.8783  0.198  0.802          -1  0.000        Marketwatch.com
2019-07-12   -0.8779  0.206  0.794          -1  0.000        Marketwatch.com
2019-07-13   -0.8658  0.251  0.749          -1  0.000           Hypebeast.com
2019-06-25   -0.8625  0.206  0.794          -1  0.000               Politico
2019-07-22   -0.8573  0.170  0.830          -1  0.000  Washingtonexaminer.com
2019-07-22   -0.8442  0.214  0.786          -1  0.000        National Review
2019-07-09   -0.8402  0.191  0.809          -1  0.000        Dailysignal.com
2019-07-08   -0.7351  0.137  0.863          -1  0.000             TechCrunch
2019-07-21   -0.6705  0.247  0.636          -1  0.117            Youtube.com

                                                         text  \
date
2019-07-22  I warned about an economic crash years before ...
2019-07-10  Bianca, a flight attendant living in Denver, h...
2019-07-12  A stock-market index of small caps is at its w...
2019-07-13  R. Kelly was arrested on several charges last ...
2019-06-25  Michael Kruse is a senior staff writer for Pol...
2019-07-22  Have you heard the news? Presidential candidat...
2019-07-22  Senator Elizabeth Warren (D., Mass.) at a town...
2019-07-09  Most\r\nof the public was outraged when, durin...
2019-07-08  Technology has been used to manage regulatory ...
2019-07-21  AbstractSpreadsheets are one of the most widel...

                                                          url
date
2019-07-22  https://medium.com/@teamwarren/the-coming-econ...
2019-07-10  https://www.marketwatch.com/story/this-flight-...
2019-07-12  https://www.marketwatch.com/story/this-stock-m...
2019-07-13  https://hypebeast.com/2019/7/r-kelly-federal-r...
2019-06-25  https://www.politico.com/magazine/story/2019/0...
2019-07-22  https://www.washingtonexaminer.com/opinion/eli...
2019-07-22  https://www.nationalreview.com/news/elizabeth-...
```

```
2019-07-09  https://www.dailysignal.com/2019/07/09/senate-...
2019-07-08  http://techcrunch.com/2019/07/08/the-startups-...
2019-07-21        https://www.youtube.com/watch?v=GyWKxFxyyrQ
```

[17]:
```
# Create a table with hvplot
neg_news_table = neg_news.hvplot.table(
    columns=["date", "source", "text", "url"], width=500
)
neg_news_table
```

[17]:
```
:Table   [date,source,text,url]
```

1.1.7 Creating the Sentiment Distribution by News Article's Source

In this section, use `hvPlot` to create a bar chart that presents the distribution of negative, neutral and positive news according to the `normalized` score; the results should be grouped by `source`.

[18]:
```
source_sentiment_chart_df = (
    crisis_news_df[["normalized", "source", "text"]]
    .groupby(["normalized", "source"])
    .count()
)
source_sentiment_chart_df.rename(
    index={-1: "Negative", 0: "Neutral", 1: "Positive"}, inplace=True
)
source_sentiment_chart_df
```

[18]:
```
                                   text
normalized source
Negative   Aol.com                    1
           BBC News                   1
           Boingboing.net             1
           Business Insider           6
           Dailysignal.com            1
           Fastcompany.com            2
           Forbes.com                 3
           Gothamist.com              1
           Huffpost.com               1
           Hypebeast.com              1
           Indianexpress.com          1
           Marketwatch.com            6
           Medium.com                 1
           Moneycontrol.com           1
           Mywallst.com               1
           National Review            1
           Phys.org                   1
           Politico                   2
           Project-syndicate.org      3
           Salon.com                  1
```

```
                Seattletimes.com           1
                TechCrunch                 1
                Thinkprogress.org          1
                Typepad.com                1
                Washingtonexaminer.com     2
                Youtube.com                1
Neutral         BBC News                   3
                Digg.com                   1
                Forbes.com                 1
                Indianexpress.com          1
...                                      ...
                The Economist              1
                Theatlantic.com            1
Positive        Autoblog.comhttps          1
                Business Insider           6
                CNN                        1
                Daily Mail                 1
                Fool.com                   2
                Forbes.com                 3
                Hbr.org                    1
                Huffpost.com               1
                Jalopnik.com               1
                Marketwatch.com            6
                Npr.org                    1
                Ozy.com                    2
                Politico                   1
                Project-syndicate.org      2
                Psfk.com                   1
                Qz.com                     1
                Reuters                    2
                Ritholtz.com               1
                Slate.com                  1
                TechCrunch                 1
                The Globe And Mail         2
                The Hill                   1
                The New York Times         1
                Theatlantic.com            1
                USA Today                  1
                Vice News                  1
                Vox.com                    1
                Yahoo.com                  2

[62 rows x 1 columns]
```

[19]:
```
source_sentiment_chart = source_sentiment_chart_df.hvplot.bar(
    xlabel="Sentiment",
    ylabel="Number of News",
    title="Sentiment Distribution by News Article's Source",
    height=450,
    width=1000,
    rot=90,
)
source_sentiment_chart
```

[19]:
```
:Bars   [normalized,source]   (text)
```

1.1.8 Creating the Word Clouds

In this section you will create two word clouds, one using the bag-of-words method and other using TF-IDF.

Bag-of-Words' Word Cloud Use the `CountVectorizer` module from `sklearn` to create a word cloud with the top 20 words with the highest counting. Save the DataFrame with the top 20 words as a CSV file named `top_words_data.csv` for future use on the dashboard creation.

[20]:
```
# Creating the CountVectorizer instance defining the stop words in English to␣
 ↪be ignored
vectorizer = CountVectorizer(stop_words="english")
```

[21]:
```
# Getting the tokenization and occurrence counting
X_words = vectorizer.fit_transform(crisis_news_df["text"])
```

[22]:
```
# Retrieve unique words list
words = vectorizer.get_feature_names()

# Get the last 100 word (just as a sample)
print(words[-100:])
```

```
['underdogs', 'underestimate', 'unemployment', 'unfolded', 'unfortunately',
'uninterrupted', 'union', 'united', 'university', 'unleashed', 'unless',
'unlocking', 'unnoticed', 'unsettling', 'unsplash', 'unthinkable', 'unveiled',
'upright', 'urban', 'usa', 'use', 'used', 'users', 'valley', 'various', 'vast',
've', 'venezuela', 'verge', 'versus', 'vessels', 'veteran', 'vice', 'victims',
'video', 'view', 'viewers', 'views', 'visit', 'vladimir', 'volatile', 'vriend',
'vucci', 'wage', 'wager', 'wall', 'wants', 'warned', 'warning', 'warren',
'warrenhas', 'washington', 'wasn', 'wasnt', 'way', 'weak', 'weakest', 'wealth',
'week', 'weekly', 'wel', 'wh', 'whats', 'white', 'widely', 'widespread',
'williamson', 'win', 'window', 'wing', 'winning', 'winters', 'wisdom', 'wise',
'witnesses', 'wobble', 'wonder', 'wondering', 'wont', 'words', 'work',
'working', 'world', 'wouldnt', 'writer', 'writers', 'written', 'yang', 'year',
'years', 'yield', 'yielded', 'yielding', 'yields', 'york', 'youtuber', 'zombie',
'zombies', 'zone', 'zurich']
```

[23]:
```
# Getting the bag of words as DataFrame
words_df = pd.DataFrame(
    list(zip(words, np.ravel(X_words.sum(axis=0)))), columns=["Word",␣
 ↪"Word_Count"]
)

# Sorting words by 'Word_Count' in descending order
words_df.sort_values(by="Word_Count", ascending=False, inplace=True)
```

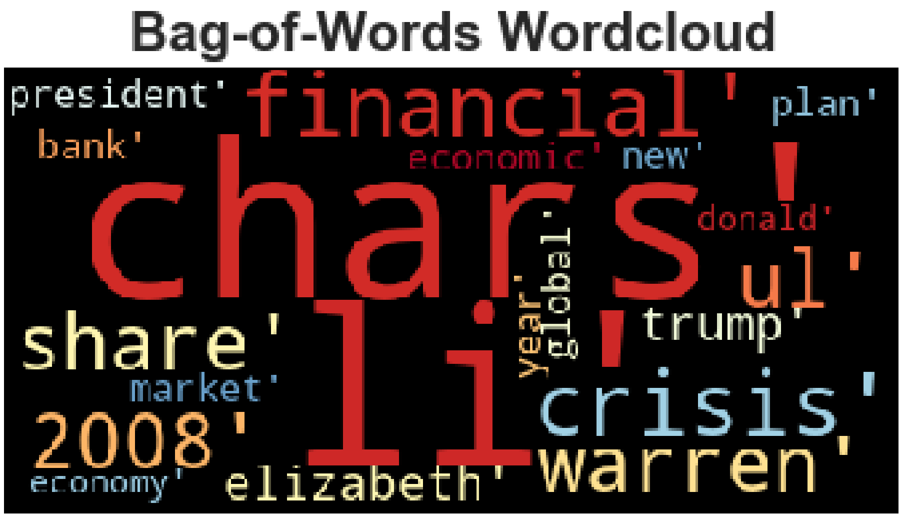
[24]:
```
# Get top 20 words with the highest counting
top_words = words_df.head(20)
top_words
```

[24]:
```
           Word  Word_Count
343       chars          99
850          li          57
601   financial          26
424      crisis          21
1494     warren          19
35         2008          18
1283      share          18
1442         ul          17
527   elizabeth          17
1431      trump          16
1100  president          13
232        bank          12
976         new          12
515    economic          12
1054       plan          11
1533       year          11
650      global          11
905      market          10
520     economy          10
499      donald          10
```

[25]:
```
# Save the top words DataFrame
file_path = Path("Data/top_words_data.csv")
top_words.to_csv(file_path)
```

[26]:
```
# Create a string list of terms to generate the bag-of-words word cloud
terms_list = str(top_words["Word"].tolist())
```

[27]:
```
# Create the bag-of-words word cloud
wordcloud = WordCloud(colormap="RdYlBu").generate(terms_list)
fig_bow_cloud = plt.figure()
plot_bow_cloud = plt.imshow(wordcloud)
plot_bow_cloud = plt.axis("off")
fontdict = {"fontsize": 20, "fontweight": "bold"}
plot_bow_cloud = plt.title("Bag-of-Words Wordcloud", fontdict=fontdict)
plot_bow_cloud = plt.show()
plt.close(fig_bow_cloud)
```



TF-IDF Wordcloud Use the `TfidfVectorizer` module from `sklearn` to create a word cloud with the top 20 words with the highest frequency. Save the DataFrame with the top 20 words as a CSV file named `top_wors_tfidf_data.csv` for future use on the dashboard creation.

[28]:
```
# Getting the TF-IDF
tfidf_vectorizer = TfidfVectorizer(stop_words="english")
X_tfidf = tfidf_vectorizer.fit_transform(crisis_news_df["text"])
```

[29]:
```
# Retrieve words list from corpous
words_tfidf = tfidf_vectorizer.get_feature_names()

# Get the last 100 word (just as a sample)
print(words_tfidf[-100:])
```

```
['underdogs', 'underestimate', 'unemployment', 'unfolded', 'unfortunately',
'uninterrupted', 'union', 'united', 'university', 'unleashed', 'unless',
'unlocking', 'unnoticed', 'unsettling', 'unsplash', 'unthinkable', 'unveiled',
'upright', 'urban', 'usa', 'use', 'used', 'users', 'valley', 'various', 'vast',
've', 'venezuela', 'verge', 'versus', 'vessels', 'veteran', 'vice', 'victims',
'video', 'view', 'viewers', 'views', 'visit', 'vladimir', 'volatile', 'vriend',
'vucci', 'wage', 'wager', 'wall', 'wants', 'warned', 'warning', 'warren',
'warrenhas', 'washington', 'wasn', 'wasnt', 'way', 'weak', 'weakest', 'wealth',
'week', 'weekly', 'wel', 'wh', 'whats', 'white', 'widely', 'widespread',
'williamson', 'win', 'window', 'wing', 'winning', 'winters', 'wisdom', 'wise',
'witnesses', 'wobble', 'wonder', 'wondering', 'wont', 'words', 'work',
'working', 'world', 'wouldnt', 'writer', 'writers', 'written', 'yang', 'year',
'years', 'yield', 'yielded', 'yielding', 'yields', 'york', 'youtuber', 'zombie',
'zombies', 'zone', 'zurich']
```

[30]:
```
# Creating a DataFrame Representation of the TF-IDF results
words_tfidf_df = pd.DataFrame(
    list(zip(words_tfidf, np.ravel(X_tfidf.sum(axis=0)))), columns=["Word",␣
 ↪"Frequency"]
)

# Sorting words by 'Frequency' in descending order
words_tfidf_df = words_tfidf_df.sort_values(by=["Frequency"], ascending=False)
```

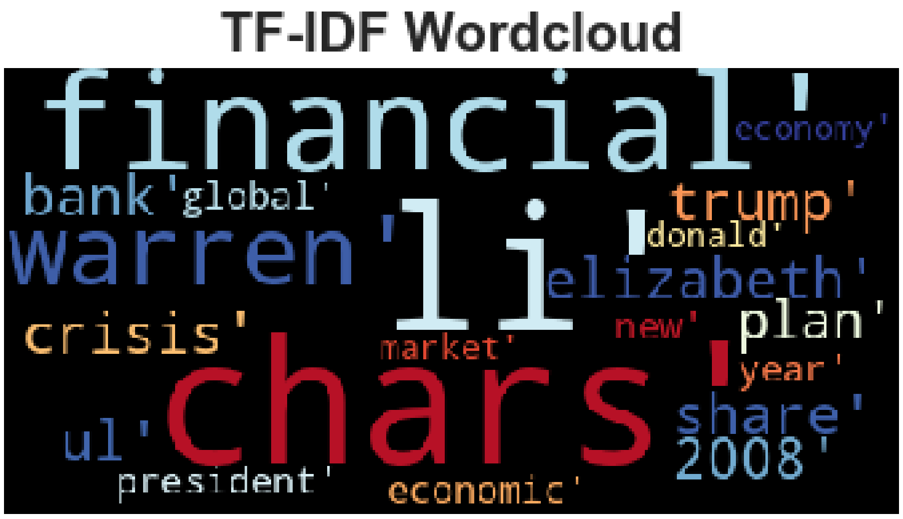
[31]:
```
# Get 20 top words
top_words_tfidf = words_tfidf_df.head(20)
top_words_tfidf
```

[31]:
```
           Word  Frequency
850          li   6.039592
343       chars   4.139945
601   financial   2.740431
1494     warren   2.601068
527   elizabeth   2.274471
424      crisis   2.269101
1283      share   2.160880
35         2008   2.060891
1431      trump   1.966178
1442         ul   1.859178
232        bank   1.697471
1054       plan   1.674686
1100  president   1.638754
1533       year   1.630204
515    economic   1.617877
650      global   1.617172
976         new   1.563557
520     economy   1.494965
905      market   1.402006
499      donald   1.352434
```

[32]:
```
# Save the top words TF-IDF DataFrame
file_path = Path("Data/top_wors_tfidf_data.csv")
top_words_tfidf.to_csv(file_path)
```

[33]:
```
# Create a string list of terms to generate the tf-idf word cloud
terms_list_tfidf = str(top_words_tfidf["Word"].tolist())
```

[34]:
```
# Create the tf-idf word cloud
wordcloud_tfidf = WordCloud(colormap="RdYlBu").generate(terms_list_tfidf)
fig_tfidf_cloud = plt.figure()
plot_tfidf_cloud = plt.imshow(wordcloud_tfidf)
plot_tfidf_cloud = plt.axis("off")
fontdict = {"fontsize": 20, "fontweight": "bold"}
plot_tfidf_cloud = plt.title("TF-IDF Wordcloud", fontdict=fontdict)
plot_tfidf_cloud = plt.show()
plt.close(fig_tfidf_cloud)
```



1.2 Challenge: Radar Chart with Tone Analysis

In this challenge section, you have to use Plotly Express and IBM Watson Tone Analyzer to create a radar chart presenting the tone of all the news articles that you retrieved.

Refer to the polar coordinates chart demo and the Plotly Express reference documentation to learn more about how to create this chart.

[35]:
```
# Get the Tone Analyzer API Key and URL
tone_api = os.getenv("tone_api")
tone_url = os.getenv("tone_url")
```

[36]:
```
# Initialize Tone Analyser Client
tone_analyzer = ToneAnalyzerV3(version="2017-09-21", iam_apikey=tone_api,␣
 ↪url=tone_url)
```

In order to create the radar chart, you need to score the tone of each article and retrieve the `document_tone`. Create a function named `get_tone(text,url)` that will receive two parameters and will get the tone score for a particular article.

- `text` the content of the article.
- `url` the URL pointing to the article.

The `get_tone()` function will use the `tone()` method from the `ToneAnalyzerV3` module to score the article's tone. Remember that for each document (or text), the `tone()` method of IBM Watson Tone Analyzer scores one or more overall document tones, you can also get and empty result if no tone were scored; this function should return a dictionary with the first document tone's score with the following structure:

- `score` refers to the first `tone` from the `document_tone`.
- `tone_id` refers to the `tone_id` from the first `tone`.
- `tone_name` refers to the `tone_name` from the first `tone`.
- `text` the text passed as parameter.
- `url` the URL passed as parameter.

This is an example of the function's return value:

```
{'score': 0.616581,
'tone_id': 'sadness',
'tone_name': 'Sadness',
'text': '\nMore than a decade since the global economic meltdown of 2008
    devastated lives across the world, no one who caused the crisis has
    been held responsible.\n\n"The 2008 financial crisis displayed what
    the world now identifies as financial contagion," says Philip J Baker,
    the former managing partner of a US-based \nhedge fund that collapsed
    during the financial crisis.\n\nDespite this, "zero Wall Street chief
    executives have been to prison, even though there is today absolutely
    no doubt that Wall Street executives and politicians \nwere complicit
    in creating the crisis," he says. \n\nBaker was among the few
    relatively smaller players imprisoned for the part they played.\n\n
    In July 2009, he was arrested in  Germany and extradited to the
    United States where he faced federal court on charges of fraud and
    financial crimes.\n\nHe pled guilty and was sentenced to 20 years
    in prison for costing some 900 investors about $294mn worldwide.
    He served eight years in jail and is now on \nparole and advocates
    against financial crime.\n',
'url': 'https://www.aljazeera.com/programmes/specialseries/2019/06/men-stole-world-2008-financ
```

[37]:
```
# Create a function to analyze the text's tone with the 'tone()' method of IBM␣
 ↪Watson Tone Analyzer.
def get_tone(text, url):
    try:
        tone_analysis = tone_analyzer.tone(
            {"text": text},
            content_type="application/json",
            sentences=False,
            content_language="en",
            accept_language="en",
        ).get_result()

        result = tone_analysis["document_tone"]["tones"][0]
        result["text"] = text
        result["url"] = url
        return result
    except:
        pass
```

Create a DataFrame with the tone scoring from all the news articles. Use an empty list to create a the DataFrame's structure and a `for-loop` to iterate across all the news to score their tone using the `get_tone()` function.

[38]:
```python
# Create an empty list to create the DataFrame's structure
articles_tone_data = []

# Iterate across all the news articles to score their tone.
print(f"Analyzing tone from {crisis_news_df.shape[0]} articles...")
for index, row in crisis_news_df.iterrows():
    try:
        print("*", end="")
        # Get news article's tone
        article_tone = get_tone(row["text"], row["url"])
        if article_tone:  # Validate if a tone where found
            articles_tone_data.append(article_tone)
    except:
        pass
print("\nDone :-)")
```

```
Analyzing tone from 100 articles...
********************************************************************************
********************
Done :-)
```

[39]:
```python
# Create the DateFrame containing the news articles and their tone scoring␣
↪results.
articles_tone_df = pd.DataFrame.from_dict(articles_tone_data)
articles_tone_df.score = articles_tone_df.score.round(2)
articles_tone_df.head()
```

[39]:
```
   score                                               text    tone_id  \
0   0.66  <ul><li>Share to facebook</li><li>Share to twi...  tentative
1   0.56  Michael Kruse is a senior staff writer for Pol...    sadness
2   0.50  Andrew Yang didnt set out to become a test pre...  tentative
3   0.53  A WELL-KNOWN stockmarket sell signal is a comp...  tentative
4   0.69  A new high for the U.S. stock market before a ...        joy

   tone_name                                                url
0  Tentative  https://www.forbes.com/sites/mayrarodriguezval...
1    Sadness  https://www.politico.com/magazine/story/2019/0...
2  Tentative  https://slate.com/news-and-politics/2019/06/an...
3  Tentative  https://www.economist.com/finance-and-economic...
4        Joy  https://www.marketwatch.com/story/a-china-trad...
```

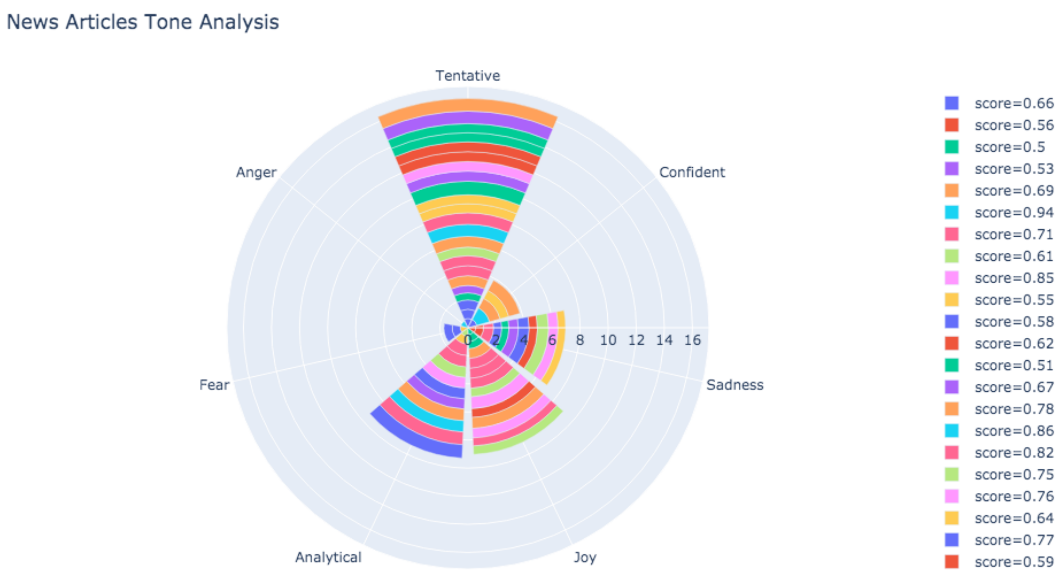
Save the DataFrame as a CSV file named `tone_data.csv` for further use on the dashboard creation.

[40]:
```python
articles_tone_df.to_csv(Path("Data/tone_data.csv"), index=False)
```

Create a radar chart using the `scatter_polar()` method from Plotly Express as follows:

- Use the `score` column for the `r` and `color` parameters.
- Use the `tone_name` column for the `theta` parameter.
- Use the `url` column for the `hover_data` parameter.
- Define a `title` for the chart.

[41]:
```python
tone_radar = px.bar_polar(
    articles_tone_df,
    r="score",
    theta="tone_name",
    color="score",
    hover_data=["url"],
    title="News Articles Tone Analysis",
)
tone_radar.show()
```


News Articles Tone Analysis
Tentative
Confident
Sadness
Joy
Analytical
Fear
Anger
score=0.66
score=0.56
score=0.5
score=0.53
score=0.69
score=0.94
score=0.71
score=0.61
score=0.85
score=0.55
score=0.58
score=0.62
score=0.51
score=0.67
score=0.78
score=0.86
score=0.82
score=0.75
score=0.76
score=0.64
score=0.77
score=0.59