sentiment_dashboard

July 26, 2019

1 Sentiment Analysis Dashboard

In this notebook, you will compile the visualizations from the previous analysis into functions that can be used for a Panel dashboard.

[1]:
```
# Initial imports
import os
from path import Path
import pandas as pd
import hvplot.pandas
from wordcloud import WordCloud
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 Loading Data

In this section, you will load the CSV files you created on the analysis notebook:

- The `news_vader.csv` file with all the news articles and the VADER sentiment scores.
- The `tone_data.csv` file with the IBM Watson Tone Analyzer results.
- The `top_words_data.cvs` file with the top 20 words for creating the word cloud based on the bag-of-words method.
- The `top_words_tfidf_data.cvs` file with the top 20 words for creating the word cloud based on TF-IDF.

[2]:
```
crisis_news_df = pd.read_csv(Path("Data/news_vader.csv"), index_col="date")
tone_df = pd.read_csv(Path("Data/tone_data.csv"))
top_words = pd.read_csv(Path("Data/top_words_data.csv"))
top_words_tfidf = pd.read_csv(Path("Data/top_wors_tfidf_data.csv"))
```

1.2 Plots Functions

In this section, you will copy the code for each plot type from your analysis notebook and place it into separate functions that Panel can use to create panes for the dashboard. These functions will convert the plot object to a Panel pane.

Be sure to include any DataFrame transformation/manipulation code required along with the plotting code. Return a Panel pane object from each function that can be used to build the dashboard.

Note: Remove any `.show()` lines from the code. We want to return the plots instead of showing them. The Panel dashboard will then display the plots.

[3]:
```
def avg_sent_plot():
    """
    Creates the averange sentiment line plot (compound and normalized scores)
    """
    avg_sent_data = (
        crisis_news_df[["compound", "normalized"]]
        .groupby(by=crisis_news_df.index)
        .mean()
    )
    avg_sent_plot = avg_sent_data.hvplot(
        title="Average Sentiment About the Economic Crisis of 2008 Last Month",␣
 ↪rot=90
    )

    return avg_sent_plot


def sentiment_bar_chart():
    """
    Creates the average sentiment distribution bar chart
    """
    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_bar_chart = sentiment_chart_df.hvplot.bar(
        xlabel="Sentiment",
        ylabel="Number of News",
        title="Overall Sentiment Distribution",
        color="text",
    )

    return sentiment_bar_chart
```

```
def pos_news_table():
    """
    Create a table with the top 10 positive news articles
    """
    pos_news = crisis_news_df.sort_values(by="compound", ascending=False)
    pos_news = pos_news.head(10)
    pos_news_table = pos_news.hvplot.table(
        columns=["date", "source", "text", "url"], width=500
    )

    return pos_news_table


def neg_news_table():
    """
    Create a table with the top 10 negative news articles
    """
    neg_news = crisis_news_df.sort_values(by="compound", ascending=True)
    neg_news = neg_news.head(10)
    neg_news_table = neg_news.hvplot.table(
        columns=["date", "source", "text", "url"], width=500
    )

    return neg_news_table


def source_sentiment_chart():
    """
    Creates the sentiment per news source bar chart
    """
    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 = 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,
    )
```

```
    return source_sentiment_chart


def bow_wordcloud():
    """
    Creates the wordcloud based on the bag-of-word method
    """
    terms_list = str(top_words["Word"].tolist())
    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)
    plt.close(fig_bow_cloud)

    return pn.pane.Matplotlib(fig_bow_cloud, tight=True, width=500)


def tfidf_wordcloud():
    """
    Creates the wordcloud based on TF-IDF
    """
    terms_list_tfidf = str(top_words_tfidf["Word"].tolist())
    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)
    plt.close(fig_tfidf_cloud)

    return pn.pane.Matplotlib(fig_tfidf_cloud, tight=True, width=500)


def tone_radar():
    """
    Creates the tone analysis radar plot
    """
    tone_radar = px.bar_polar(
        tone_df,
        r="score",
        theta="tone_name",
        color="score",
        hover_data=["url"],
        title="News Articles Tone Analysis",
    )
```

```
    return pn.pane.Plotly(tone_radar)
```

1.3 Dashboard Creation

In this section, you will combine all of the plots into a single dashboard view using Panel. Be creative with your dashboard design!

[4]:
```
# Define dashboard title
title = pn.pane.Markdown(
    """
# Sentiment Analysis Dashboard from News Articles About the 2008 Crisis
""",
    width=900,
)
```

[5]:
```
# Define dashboard welcome text
welcome = pn.pane.Markdown(
    """
This dashboard presents a visual analysis of the sentiments about the financial␣
 ↪crisis of 2008
from the news articles in the last month that mentioned this historical fact.

News articles are retrieved using the [News API service](https://newsapi.org).
"""
)
```

[6]:
```
# Create a tab layout for the dashboard
tabs = pn.Tabs(
    ("Welcome", pn.Row(welcome, pn.Column(avg_sent_plot(),␣
 ↪sentiment_bar_chart()))),
    (
        "News Articles Analysis",
        pn.Column(
            pn.Row(
                pn.Column("## Top 10 Positive News", pos_news_table()),
                pn.Column("## Top 10 Negative News", neg_news_table()),
            ),
            source_sentiment_chart(),
        ),
    ),
    ("Wordcloud Analysis", pn.Row(bow_wordcloud(), tfidf_wordcloud())),
    ("Radar News Plot", pn.Column(tone_radar())),
)

panel = pn.Column(pn.Row(title), tabs, width=900)
```

1.4 Serve the Panel Dashboard

[7]:
```
panel.servable()
```

[7]:
```
Column(width=900)
    [0] Row
        [0] Markdown(str, width=900)
    [1] Tabs
        [0] Row
            [0] Markdown(str)
            [1] Column
                [0] HoloViews(NdOverlay)
                [1] HoloViews(Bars)
        [1] Column
            [0] Row
                [0] Column
                    [0] Markdown(str)
                    [1] HoloViews(Table)
                [1] Column
                    [0] Markdown(str)
                    [1] HoloViews(Table)
            [1] HoloViews(Bars)
        [2] Row
            [0] Matplotlib(Figure, tight=True, width=500)
            [1] Matplotlib(Figure, tight=True, width=500)
        [3] Column
            [0] Plotly(Figure)
```