Israel-Hamas War Timeline Analysis

(Niranjan Rathor)(niranjanrathor888@gmail.com)

Problem Statement

The task is to analyze a dataset of news articles, extract relevant events related to the Israel-Hamas conflict, and visualize these events on a timeline. The goal is to understand the frequency and nature of the reported events over time.

Approach and Methodology

Step 1: Data Preparation

1. **Load the JSON file**: The first step involves loading the JSON file containing the news articles. This is done using Python's built-in `json` module.
2. **Preprocess the articles**: The text of the articles is cleaned to remove noise such as punctuation, special characters, and irrelevant symbols using regular expressions.
3. **Filter relevant articles**: We filter the articles to retain only those that are relevant to the Israel-Hamas conflict. This is achieved using keyword-based filtering, where we search for specific terms related to the conflict (e.g., "Israel", "Hamas", "Gaza", "Palestine", "war").

Step 2: Event Extraction

1. **Identify date mentions**: We use NLP techniques, specifically the spaCy library, to extract dates mentioned in the text of the articles.
2. **Identify event descriptions**: We extract sentences or paragraphs that describe events related to the conflict. This is done by splitting the text into sentences and retaining those that mention key terms.

Step 3: Summarization

1. **Cluster events by date**: Events are grouped based on their extracted dates. This involves parsing and normalizing the dates to ensure consistency.
2. **Summarize clustered events**: For each date, we create concise summaries by taking the first few sentences of the events described on that date.

Step 4: Visualization

Create a timeline plot: We use the `matplotlib` library to visualize the events on a timeline. Each event is annotated with its date and a brief description to provide context.

Code Explanation

Data Preparation

```python
# Step 1: Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Step 2: Load the JSON file
import json
import pandas as pd

# Path to the JSON file in Google Drive
file_path = '/content/drive/My Drive/news.article.json'

# Load the JSON file
with open(file_path, 'r') as file:
    articles = json.load(file)

# Step 3: Convert to DataFrame for easier manipulation
df = pd.DataFrame(articles)
```

```
Mounted at /content/drive
```

Text Cleaning and Filtering

```python
import re

def clean_text(text):
    text = re.sub(r'\s+', ' ', text)  # Remove extra whitespace
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    return text

df['cleaned_text'] = df['articleBody'].apply(clean_text)
```

```python
def filter_relevant_articles(text):
    keywords = ['Israel', 'Hamas', 'Gaza', 'Palestine', 'war']
    return any(keyword.lower() in text.lower() for keyword in keywords)

df['is_relevant'] = df['cleaned_text'].apply(filter_relevant_articles)
relevant_articles = df[df['is_relevant']]
```

```python
relevant_articles.head()
```

	articleBody	dateModified	scrapedDate	source	title	cleaned_text	is_relevant
0	Sanjay Raut, a member of the Shiv Sena (UBT) p...	{'$date': '2023-10-25T06:35:50.000Z'}	{'$date': '2023-10-27T13:12:18.339Z'}	https://www.thehansindia.com/	Shiv Sena MP Sanjay Raut Responds To 'Hamas' R...	Sanjay Raut a member of the Shiv Sena UBT part...	True
1	Kozhikode (Kerala) [India], October 27 (ANI): ...	NaN	{'$date': '2023-10-27T13:12:45.595Z'}	https://www.aninews.in/	At IUML's pro-Palestine rally in Kerala Tharoo...	Kozhikode Kerala India October 27 ANI Pointing...	True
2	Mumbai, Oct 24 (PTI) Maharashtra Chief Ministe...	{'$date': '2023-10-25T02:14:27.000Z'}	{'$date': '2023-10-27T13:12:18.339Z'}	https://thefederal.com/	Uddhav buried Bal Thackeray's 'Hindutva' for p...	Mumbai Oct 24 PTI Maharashtra Chief Minister E...	True

Event Extraction

```
[ ] import spacy
    from dateutil.parser import parse

    # Load spaCy model
    nlp = spacy.load("en_core_web_sm")

    def extract_dates(text):
        doc = nlp(text)
        dates = [ent.text for ent in doc.ents if ent.label_ == 'DATE']
        return dates

    relevant_articles['dates'] = relevant_articles['cleaned_text'].apply(extract_dates)
```

```
<ipython-input-6-e2651087d89f>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
  relevant_articles['dates'] = relevant_articles['cleaned_text'].apply(extract_dates)
```

```
def extract_events(text):
    sentences = text.split('.')
    return sentences

relevant_articles['events'] = relevant_articles['cleaned_text'].apply(extract_events)
```

Summarization:

```
from collections import defaultdict
from dateutil.parser import parse

event_dict = defaultdict(list)

for _, row in relevant_articles.iterrows():
    for date in row['dates']:
        try:
            parsed_date = parse(date, fuzzy=True)
            for event in row['events']:
                event_dict[parsed_date.date()].append(event)
        except Exception as e:
            print(f"Error parsing date: {e}")
            continue
```

```
Streaming output truncated to the last 5000 lines.
```

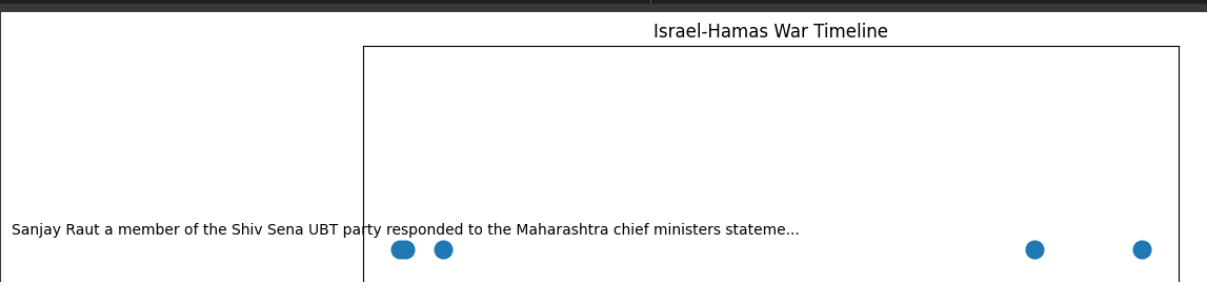
```
[ ] def summarize_events(events):
        return ' '.join(events[:5])  # Take the first 5 events as a simple summary

    summarized_events = {date: summarize_events(events) for date, events in event_dict.items()}
```

```
print(f"Number of dates: {len(dates)}")
print(f"Number of events: {len(events)}")
print("Sample dates:", dates[:5])
print("Sample events:", events[:5])
```

```
ganisations such as Hamas Hizbul Mujahideen LashkareToiba for selfish motives and chair Eknath Shinde should
```

Visualization

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Function to truncate long text
def truncate_text(text, max_length=100):
    return (text[:max_length] + '...') if len(text) > max_length else text

# Truncate the events
truncated_events = [truncate_text(event) for event in events]

# Plotting
plt.figure(figsize=(10, 5))
plt.plot_date(dates, [1] * len(dates), markersize=12)

for date, event in zip(dates, truncated_events):
    plt.annotate(event, (mdates.date2num(date), 1), textcoords="offset points", xytext=(0, 10), ha='center')

plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
plt.gca().yaxis.set_visible(False)

plt.title('Israel-Hamas War Timeline')
plt.show()
```



Assumptions

1. **Relevance of Articles**: We assume that articles containing specific keywords are relevant to the Israel-Hamas conflict.
2. **Date Extraction**: We assume that dates extracted by the spaCy model are accurate and relevant to the events described.
3. **Event Summarization**: We assume that the first few sentences of each event provide a meaningful summary.