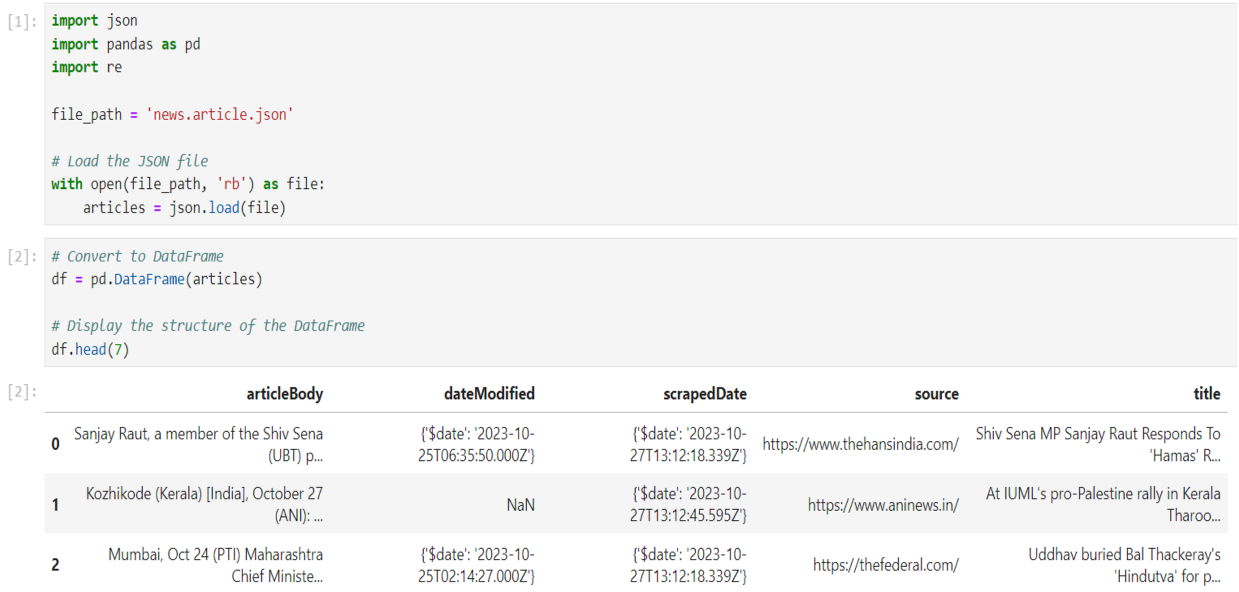
Data Preprocessing.ipynb



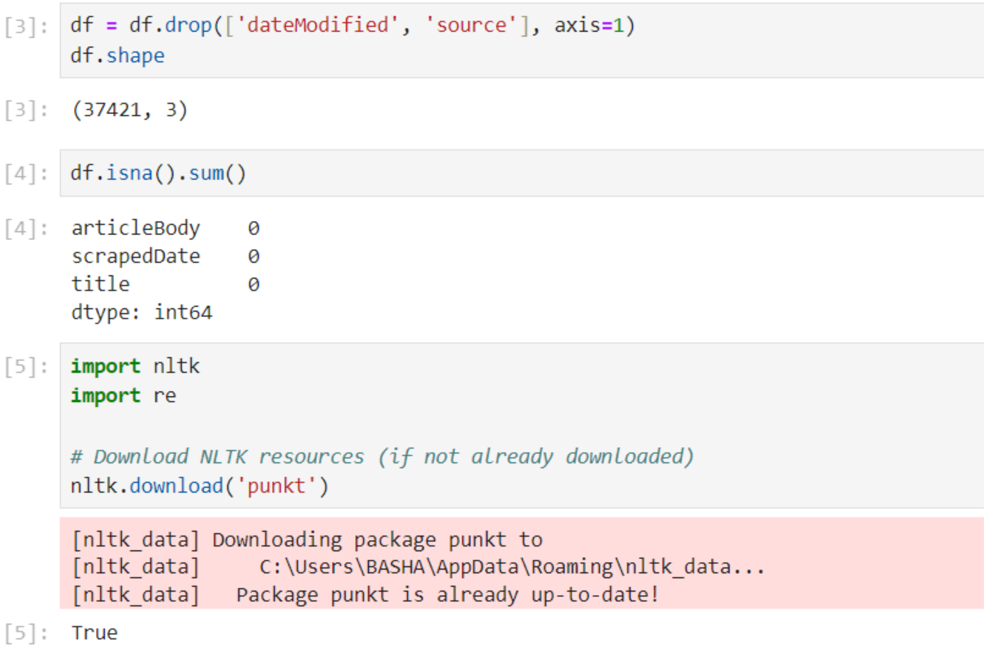
```
[1]: import json
     import pandas as pd
     import re

     file_path = 'news.article.json'

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

```
[2]: # Convert to DataFrame
     df = pd.DataFrame(articles)

     # Display the structure of the DataFrame
     df.head(7)
```

[2]:

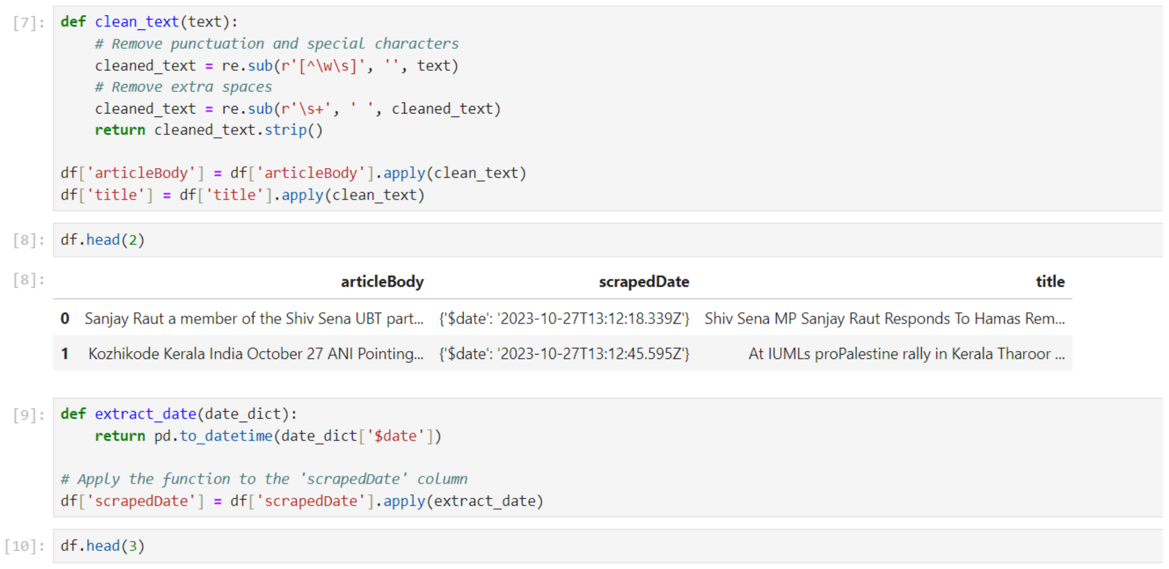
	articleBody	dateModified	scrapedDate	source	title
0	Sanjay Raut, a member of the Shiv Sena (UBT) p...	{'$date': '2023-10-25T06:35:50.000Z'}	{'$date': '2023-10-27T13:12:18.339Z'}	https://www.thehansindia.com/	Shiv Sena MP Sanjay Raut Responds To 'Hamas' R...
1	Kozhikode (Kerala) [India], October 27 (ANI):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

The notebook starts by importing necessary Python libraries, primarily Pandas for data manipulation. The dataset is read into a DataFrame using Pandas.



```
[3]: df = df.drop(['dateModified', 'source'], axis=1)
     df.shape

[3]: (37421, 3)

[4]: df.isna().sum()

[4]: articleBody    0
     scrapedDate    0
     title          0
     dtype: int64

[5]: import nltk
     import re

     # Download NLTK resources (if not already downloaded)
     nltk.download('punkt')

     [nltk_data] Downloading package punkt to
     [nltk_data]     C:\Users\BASHA\AppData\Roaming\nltk_data...
     [nltk_data]   Package punkt is already up-to-date!

[5]: True
```

Columns that are not necessary are removed from the dataframe and checked if any NA values are present in the dataframe.



```
[7]: def clean_text(text):
         # Remove punctuation and special characters
         cleaned_text = re.sub(r'[^\w\s]', '', text)
         # Remove extra spaces
         cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
         return cleaned_text.strip()

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

```
[8]: df.head(2)
```

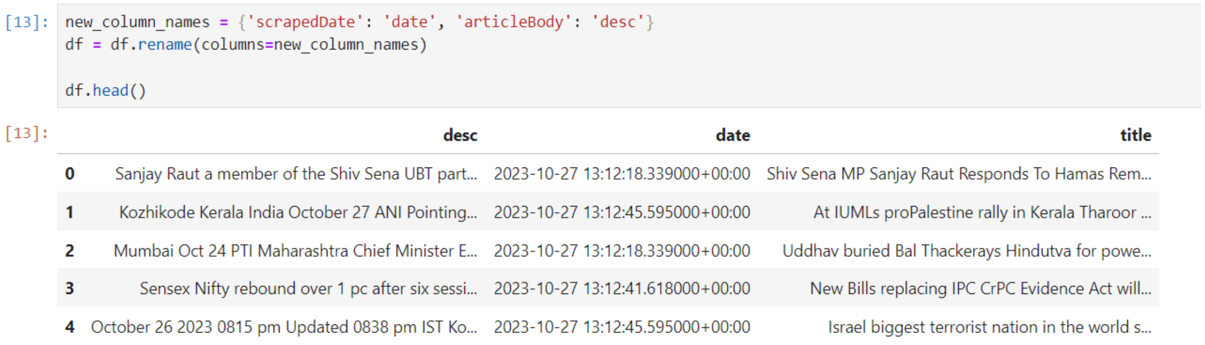
[8]:

	articleBody	scrapedDate	title
0	Sanjay Raut a member of the Shiv Sena UBT part...	{'$date': '2023-10-27T13:12:18.339Z'}	Shiv Sena MP Sanjay Raut Responds To Hamas Rem...
1	Kozhikode Kerala India October 27 ANI Pointing...	{'$date': '2023-10-27T13:12:45.595Z'}	At IUMLs proPalestine rally in Kerala Tharoor ...

```
[9]: def extract_date(date_dict):
         return pd.to_datetime(date_dict['$date'])

     # Apply the function to the 'scrapedDate' column
     df['scrapedDate'] = df['scrapedDate'].apply(extract_date)
```

```
[10]: df.head(3)
```

A function clean_text is defined to clean the text data. This function removes punctuation and extra spaces. The clean_text function is applied to the articleBody and title columns of the DataFrame.

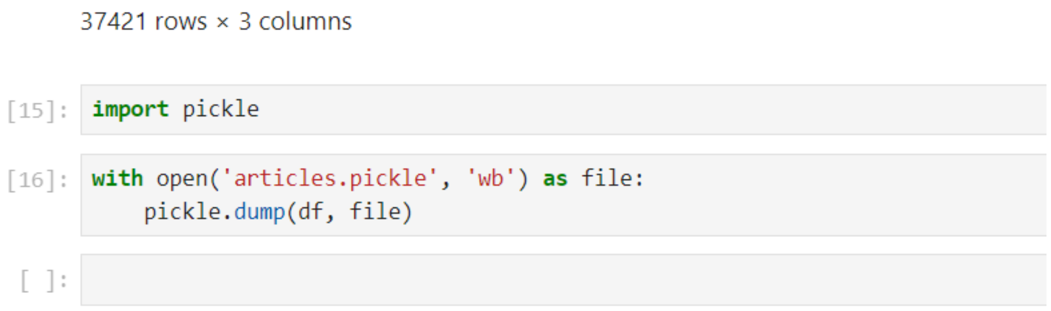


```
[13]: new_column_names = {'scrapedDate': 'date', 'articleBody': 'desc'}
      df = df.rename(columns=new_column_names)

      df.head()
```

[13]:

	desc	date	title
0	Sanjay Raut a member of the Shiv Sena UBT part...	2023-10-27 13:12:18.339000+00:00	Shiv Sena MP Sanjay Raut Responds To Hamas Rem...
1	Kozhikode Kerala India October 27 ANI Pointing...	2023-10-27 13:12:45.595000+00:00	At IUMLs proPalestine rally in Kerala Tharoor ...
2	Mumbai Oct 24 PTI Maharashtra Chief Minister E...	2023-10-27 13:12:18.339000+00:00	Uddhav buried Bal Thackerays Hindutva for powe...
3	Sensex Nifty rebound over 1 pc after six sessi...	2023-10-27 13:12:41.618000+00:00	New Bills replacing IPC CrPC Evidence Act will...
4	October 26 2023 0815 pm Updated 0838 pm IST Ko...	2023-10-27 13:12:45.595000+00:00	Israel biggest terrorist nation in the world s...

Column names are renamed to new column names



37421 rows × 3 columns

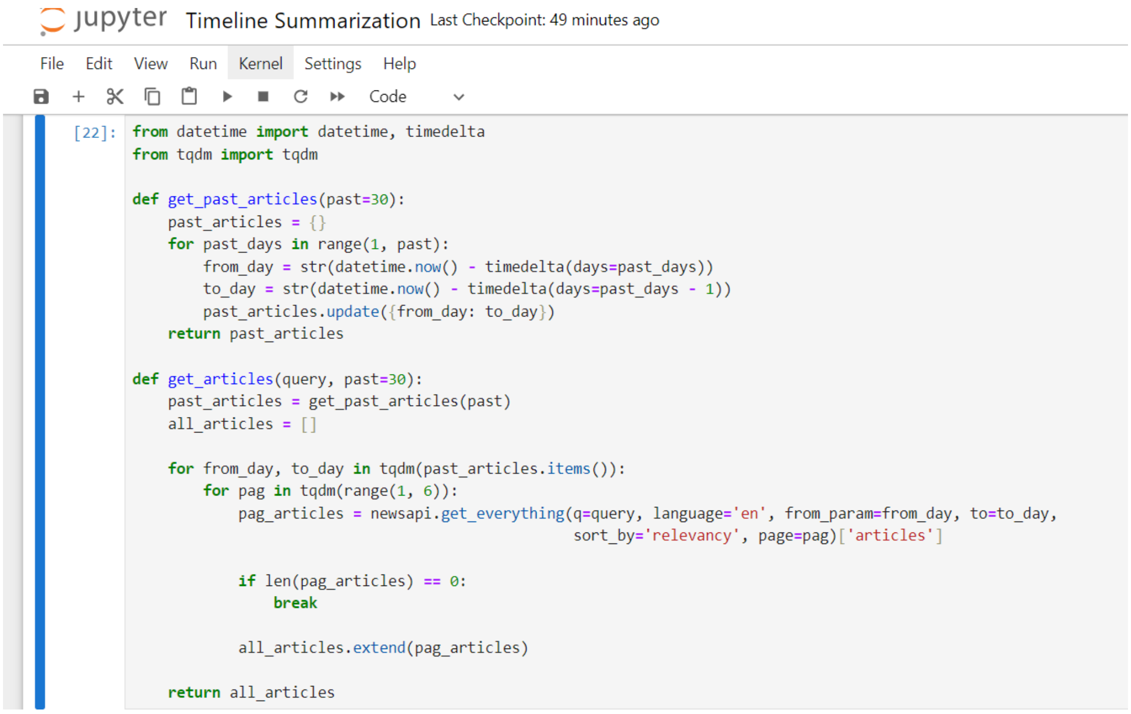
```
[15]: import pickle
```

```
[16]: with open('articles.pickle', 'wb') as file:
          pickle.dump(df, file)
```

```
[ ]:
```

The cleaned DataFrame is saved to a pickle file for later use.

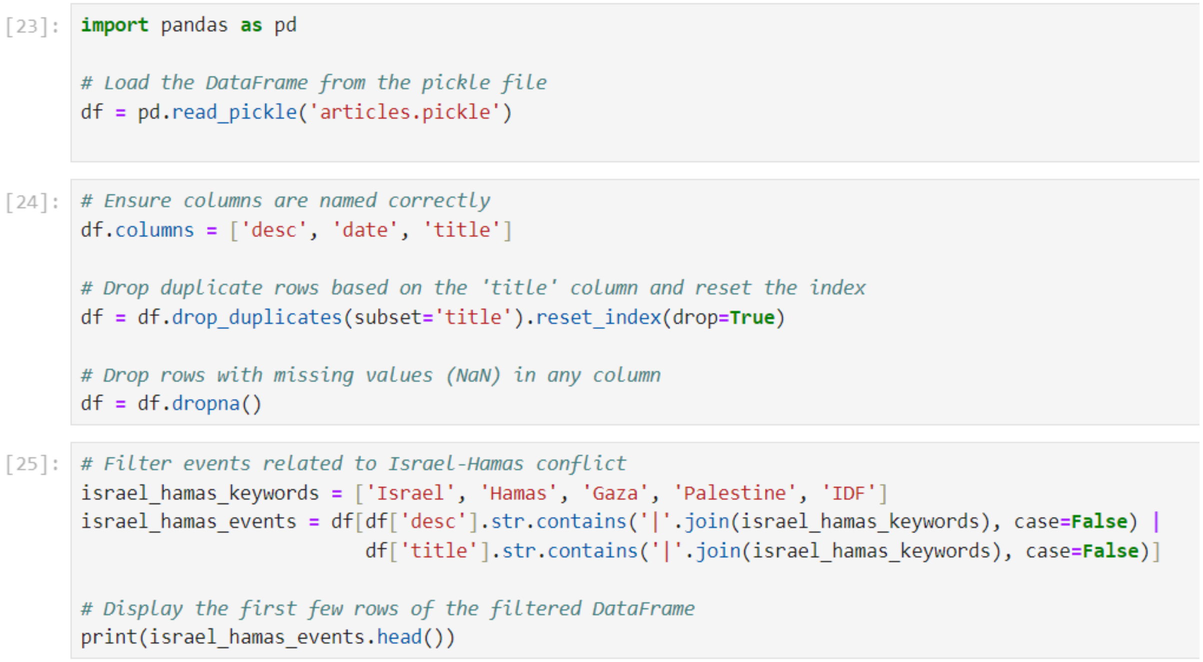
Timeline Summarization.ipynb



```
[22]: from datetime import datetime, timedelta
      from tqdm import tqdm

      def get_past_articles(past=30):
          past_articles = {}
          for past_days in range(1, past):
              from_day = str(datetime.now() - timedelta(days=past_days))
              to_day = str(datetime.now() - timedelta(days=past_days - 1))
              past_articles.update({from_day: to_day})
          return past_articles

      def get_articles(query, past=30):
          past_articles = get_past_articles(past)
          all_articles = []

          for from_day, to_day in tqdm(past_articles.items()):
              for pag in tqdm(range(1, 6)):
                  pag_articles = newsapi.get_everything(q=query, language='en', from_param=from_day, to=to_day,
                                                        sort_by='relevancy', page=pag)['articles']

                  if len(pag_articles) == 0:
                      break

                  all_articles.extend(pag_articles)

          return all_articles
```

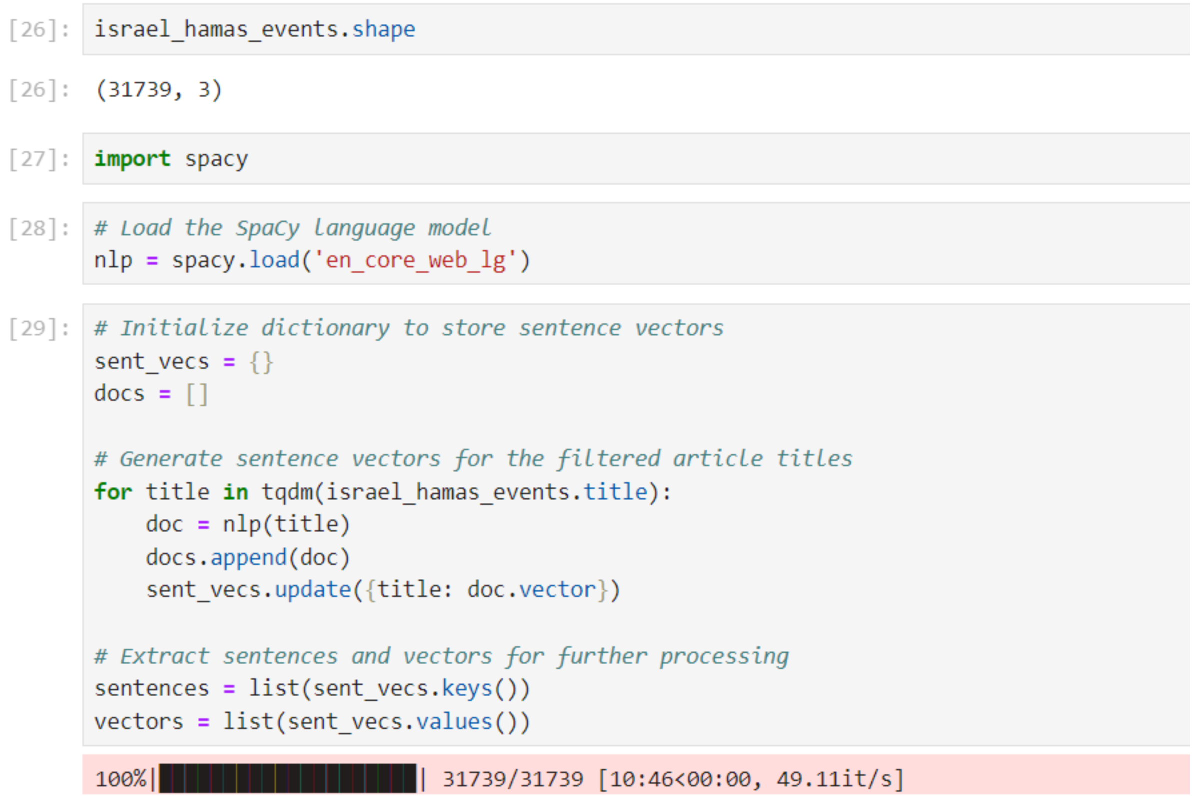
The get_past_articles function generates a dictionary where each key is a string representation of a past date and each value is the string representation of the next day. The function covers a range of past days.

The get_articles function retrieves articles related to a specific query from the News API over a range of past days and combines them into a list.



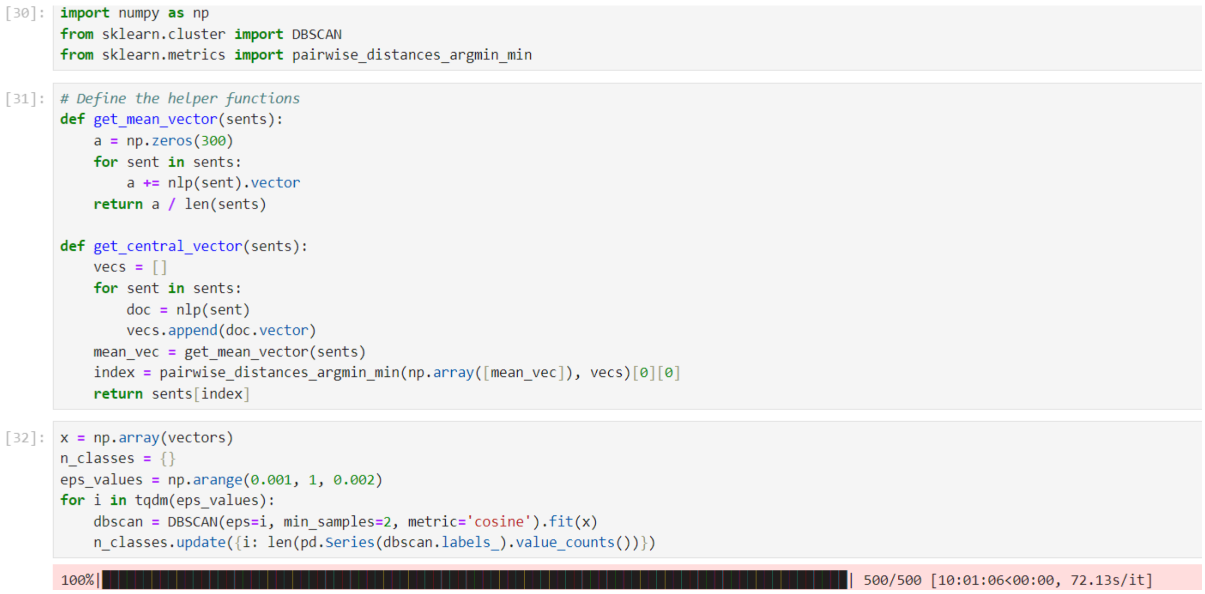
```
[23]: import pandas as pd

      # Load the DataFrame from the pickle file
      df = pd.read_pickle('articles.pickle')

[24]: # Ensure columns are named correctly
      df.columns = ['desc', 'date', 'title']

      # Drop duplicate rows based on the 'title' column and reset the index
      df = df.drop_duplicates(subset='title').reset_index(drop=True)

      # Drop rows with missing values (NaN) in any column
      df = df.dropna()

[25]: # Filter events related to Israel-Hamas conflict
      israel_hamas_keywords = ['Israel', 'Hamas', 'Gaza', 'Palestine', 'IDF']
      israel_hamas_events = df[df['desc'].str.contains('|'.join(israel_hamas_keywords), case=False) |
                               df['title'].str.contains('|'.join(israel_hamas_keywords), case=False)]

      # Display the first few rows of the filtered DataFrame
      print(israel_hamas_events.head())
```

We loaded DataFrame from a pickle file named articles.pickle . We also removed duplicate rows from the DataFrame, where duplicates are identified based on the title column. We defined a list of keywords related to the Israel-Hamas conflict. These keywords will be used to filter the DataFrame for relevant events. This block of code filters the DataFrame to include only the rows where the desc or title columns contain any of the keywords defined in israel_hamas_keywords.



```
[26]: israel_hamas_events.shape

[26]: (31739, 3)

[27]: import spacy

[28]: # Load the SpaCy language model
      nlp = spacy.load('en_core_web_lg')

[29]: # Initialize dictionary to store sentence vectors
      sent_vecs = {}
      docs = []

      # Generate sentence vectors for the filtered article titles
      for title in tqdm(israel_hamas_events.title):
          doc = nlp(title)
          docs.append(doc)
          sent_vecs.update({title: doc.vector})

      # Extract sentences and vectors for further processing
      sentences = list(sent_vecs.keys())
      vectors = list(sent_vecs.values())

      100%|████████████████████| 31739/31739 [10:46<00:00, 49.11it/s]
```

The above code processes each title in the filtered DataFrame, computes its vector representation, and stores it. This code extracts the sentences (titles) and their vectors into separate lists for further processing.



```
[30]: import numpy as np
      from sklearn.cluster import DBSCAN
      from sklearn.metrics import pairwise_distances_argmin_min

[31]: # Define the helper functions
      def get_mean_vector(sents):
          a = np.zeros(300)
          for sent in sents:
              a += nlp(sent).vector
          return a / len(sents)

      def get_central_vector(sents):
          vecs = []
          for sent in sents:
              doc = nlp(sent)
              vecs.append(doc.vector)
          mean_vec = get_mean_vector(sents)
          index = pairwise_distances_argmin_min(np.array([mean_vec]), vecs)[0][0]
          return sents[index]

[32]: x = np.array(vectors)
      n_classes = {}
      eps_values = np.arange(0.001, 1, 0.002)
      for i in tqdm(eps_values):
          dbscan = DBSCAN(eps=i, min_samples=2, metric='cosine').fit(x)
          n_classes.update({i: len(pd.Series(dbscan.labels_).value_counts())})

      100%|██████████████████████████████████████████████| 500/500 [10:01:06<00:00, 72.13s/it]
```

The get_mean_vector function computes the mean vector for a list of sentences. The get_central_vector function finds the sentence whose vector is closest to the mean vector of all given sentences.

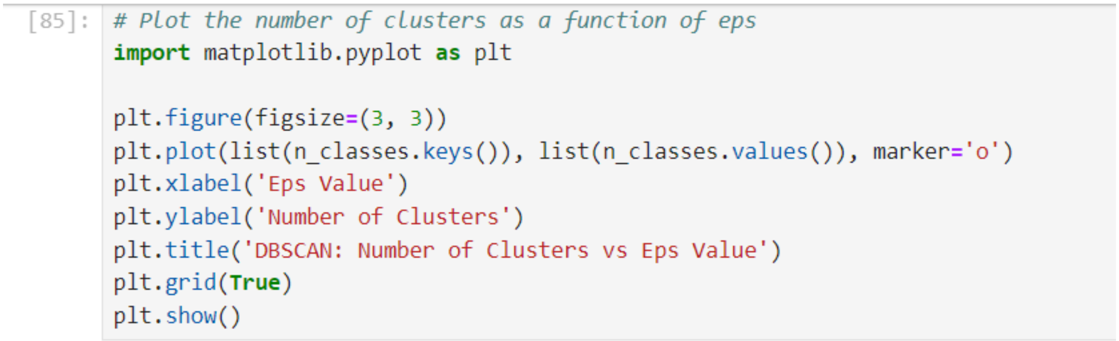
Varibles

x : Converts the list of vectors into a NumPy array. vectors should be a list of sentence vectors previously computed.

n_classes : Initializes an empty dictionary to store the number of clusters for each eps value.

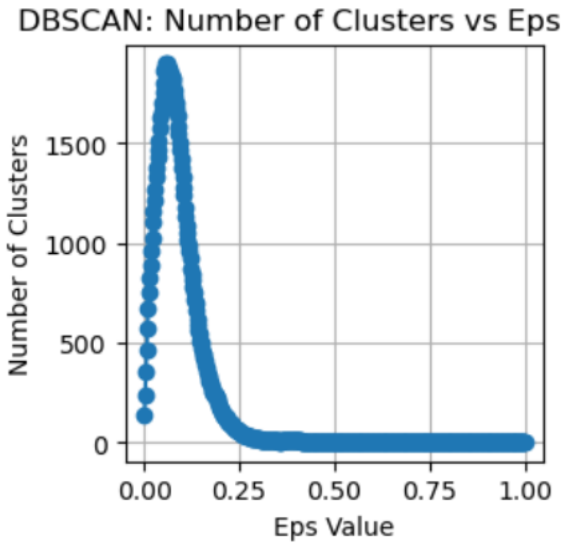
eps_values : Creates an array of values ranging from 0.001 to 1, with a step size of 0.002.

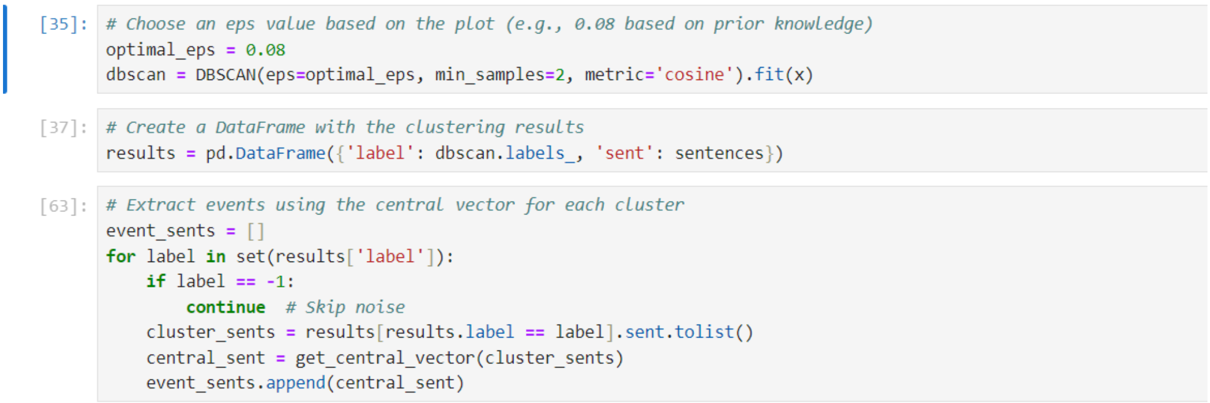
This code applies the DBSCAN clustering algorithm to the vectors with varying eps values and stores the number of clusters found for each eps value. This process helps in understanding how the number of clusters changes with different eps values, which is useful for selecting an appropriate eps value for the DBSCAN algorithm.



[85]:

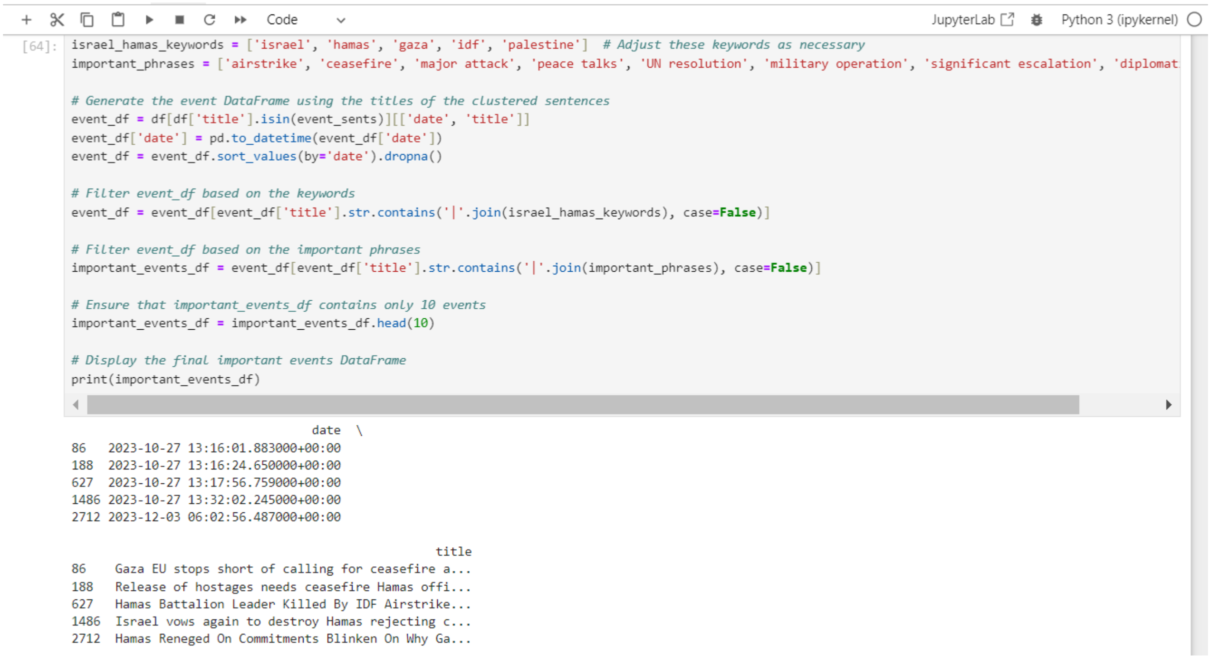
```
# Plot the number of clusters as a function of eps
import matplotlib.pyplot as plt

plt.figure(figsize=(3, 3))
plt.plot(list(n_classes.keys()), list(n_classes.values()), marker='o')
plt.xlabel('Eps Value')
plt.ylabel('Number of Clusters')
plt.title('DBSCAN: Number of Clusters vs Eps Value')
plt.grid(True)
plt.show()
```





```
[35]: # Choose an eps value based on the plot (e.g., 0.08 based on prior knowledge)
      optimal_eps = 0.08
      dbscan = DBSCAN(eps=optimal_eps, min_samples=2, metric='cosine').fit(x)

[37]: # Create a DataFrame with the clustering results
      results = pd.DataFrame({'label': dbscan.labels_, 'sent': sentences})

[63]: # Extract events using the central vector for each cluster
      event_sents = []
      for label in set(results['label']):
          if label == -1:
              continue  # Skip noise
          cluster_sents = results[results.label == label].sent.tolist()
          central_sent = get_central_vector(cluster_sents)
          event_sents.append(central_sent)
```

Extract representative events from clusters by identifying the central sentence for each cluster. This code ensures that for each cluster of sentences, a single representative sentence is selected, capturing the essence of the cluster. This can be useful in summarizing large amounts of textual data by focusing on key events or topics represented by the clusters.

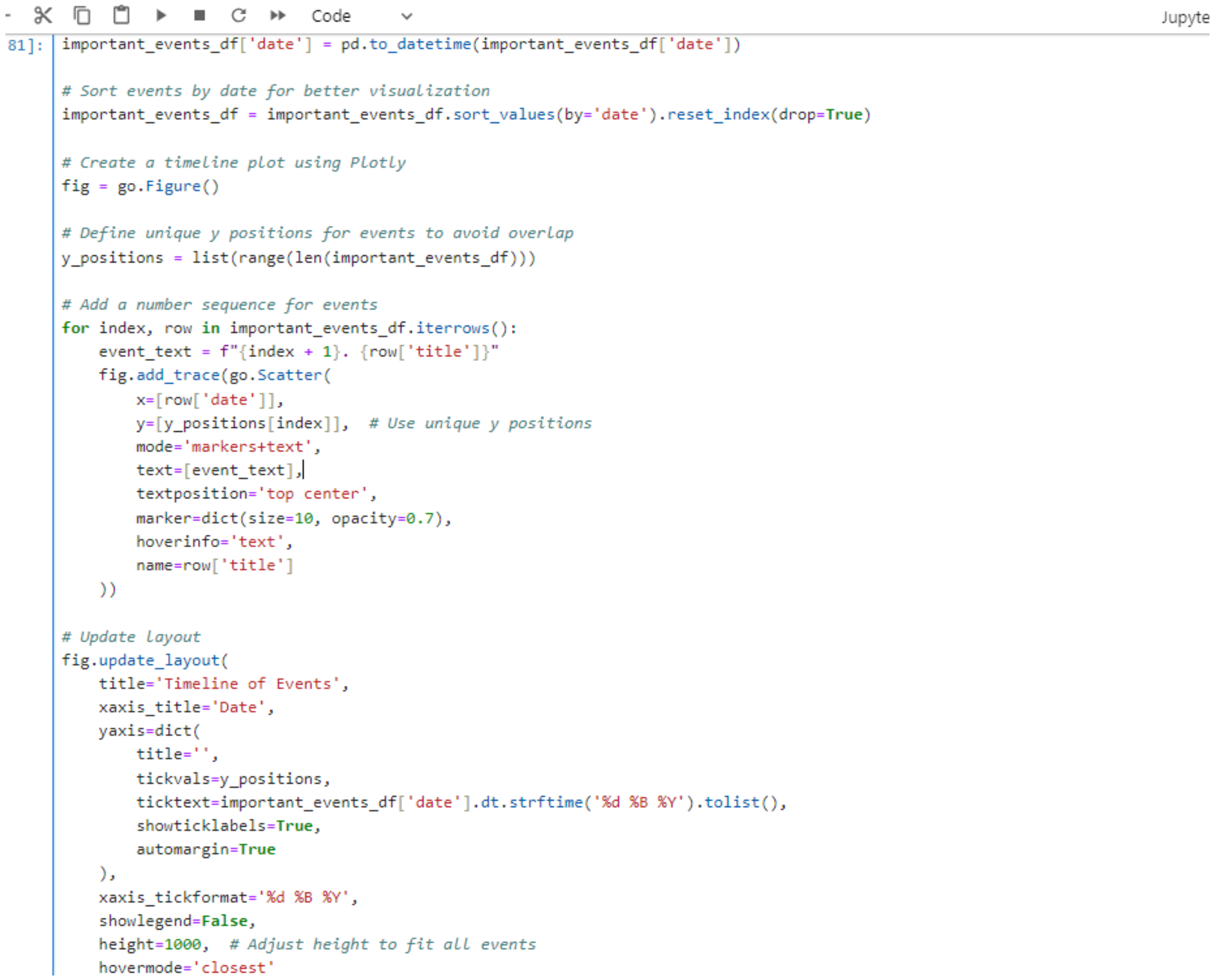


```
[64]: israel_hamas_keywords = ['israel', 'hamas', 'gaza', 'idf', 'palestine']  # Adjust these keywords as necessary
      important_phrases = ['airstrike', 'ceasefire', 'major attack', 'peace talks', 'UN resolution', 'military operation', 'significant escalation', 'diplomat

      # Generate the event DataFrame using the titles of the clustered sentences
      event_df = df[df['title'].isin(event_sents)][['date', 'title']]
      event_df['date'] = pd.to_datetime(event_df['date'])
      event_df = event_df.sort_values(by='date').dropna()

      # Filter event_df based on the keywords
      event_df = event_df[event_df['title'].str.contains('|'.join(israel_hamas_keywords), case=False)]

      # Filter event_df based on the important phrases
      important_events_df = event_df[event_df['title'].str.contains('|'.join(important_phrases), case=False)]

      # Ensure that important_events_df contains only 10 events
      important_events_df = important_events_df.head(10)

      # Display the final important events DataFrame
      print(important_events_df)
```

```
                                date  \
86   2023-10-27 13:16:01.883000+00:00
188  2023-10-27 13:16:24.650000+00:00
627  2023-10-27 13:17:56.759000+00:00
1486 2023-10-27 13:32:02.245000+00:00
2712 2023-12-03 06:02:56.487000+00:00

                                                  title
86    Gaza EU stops short of calling for ceasefire a...
188   Release of hostages needs ceasefire Hamas offi...
627   Hamas Battalion Leader Killed By IDF Airstrike...
1486  Israel vows again to destroy Hamas rejecting c...
2712  Hamas Reneged On Commitments Blinken On Why Ga...
```

Create lists of keywords and important phrases related to the Israel-Hamas conflict. Filter the original DataFrame to include dates and titles of events (from event_sents), convert dates to datetime format, and sort by date. Keep only events with titles containing Israel-Hamas keywords.

Filter by Important Phrases:
Further filter events to include only those with important phrases in the titles. Ensure the final DataFrame contains only the top 10 events. Print the final DataFrame of important events.

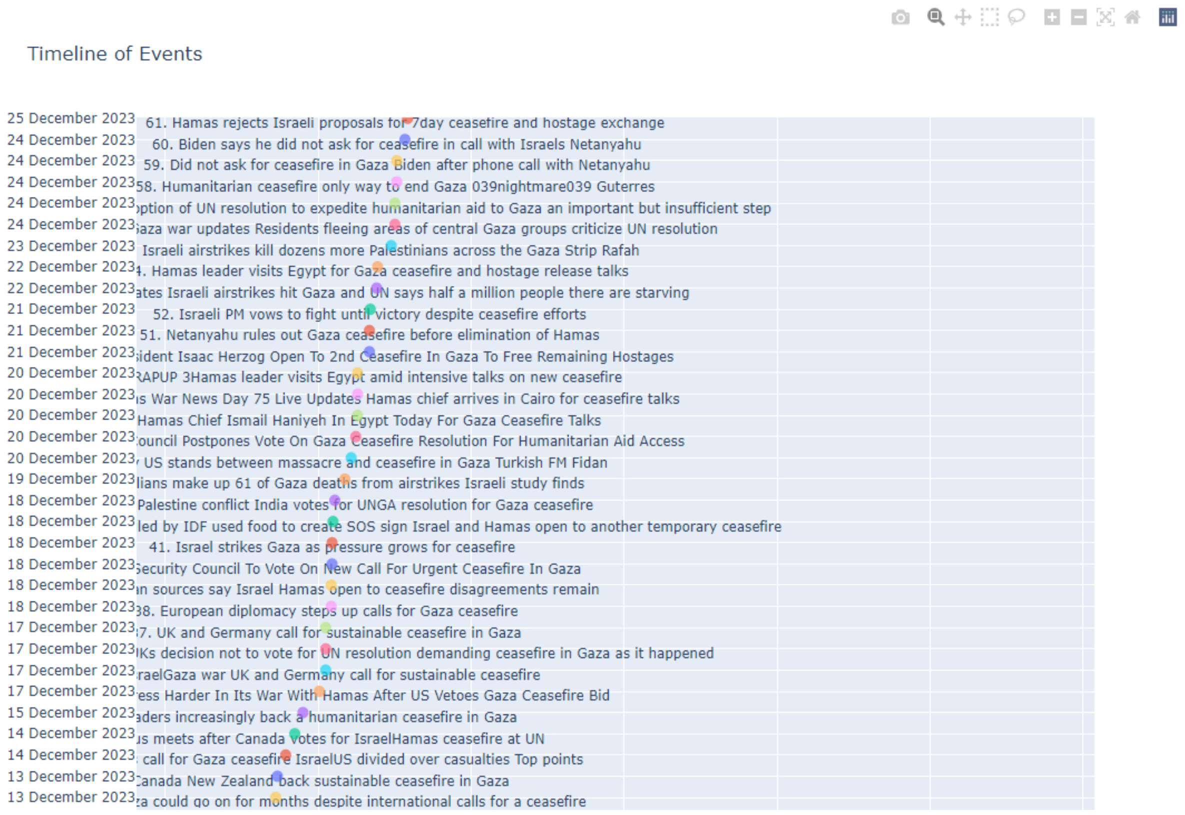


```python
important_events_df['date'] = pd.to_datetime(important_events_df['date'])

# Sort events by date for better visualization
important_events_df = important_events_df.sort_values(by='date').reset_index(drop=True)

# Create a timeline plot using Plotly
fig = go.Figure()

# Define unique y positions for events to avoid overlap
y_positions = list(range(len(important_events_df)))

# Add a number sequence for events
for index, row in important_events_df.iterrows():
    event_text = f"{index + 1}. {row['title']}"
    fig.add_trace(go.Scatter(
        x=[row['date']],
        y=[y_positions[index]],  # Use unique y positions
        mode='markers+text',
        text=[event_text],
        textposition='top center',
        marker=dict(size=10, opacity=0.7),
        hoverinfo='text',
        name=row['title']
    ))

# Update layout
fig.update_layout(
    title='Timeline of Events',
    xaxis_title='Date',
    yaxis=dict(
        title='',
        tickvals=y_positions,
        ticktext=important_events_df['date'].dt.strftime('%d %B %Y').tolist(),
        showticklabels=True,
        automargin=True
    ),
    xaxis_tickformat='%d %B %Y',
    showlegend=False,
    height=1000,  # Adjust height to fit all events
    hovermode='closest'
```

This code creates a timeline plot using Plotly to visualize a DataFrame of important events.

1. Converting Date Column: Converts the 'date' column in the DataFrame important_events_df to datetime format using pd.to_datetime().
2. Sorting the DataFrame: Sorts the DataFrame by the 'date' column in ascending order to ensure events are plotted chronologically. It then resets the index of the DataFrame.
3. Creating the Plotly Figure: Initializes a Plotly figure object fig.
4. Assigning Y Positions: Creates a list y_positions containing unique y-values for each event. This is done to avoid overlapping events on the plot.
5. Adding Events to the Plot: Iterates over each row in the sorted DataFrame and adds a Scatter trace to the figure for each event. The x-value is the event's date, and the y-value is the corresponding unique y-position. The event's title is displayed as text at the top of the marker.
6. Updating Layout: Updates the layout of the figure. It sets the title of the plot, labels the x-axis with 'Date', and customizes the y-axis ticks to display the event dates instead of the default y-values. The y-axis ticks are labeled with the formatted dates from the 'date' column.
7. Displaying the Plot: Calls fig.show() to display the Plotly figure with the timeline of events.

TIMELINE SUMMARIZATION


Timeline of Events
25 December 2023
24 December 2023
24 December 2023
24 December 2023
24 December 2023
24 December 2023
23 December 2023
22 December 2023
22 December 2023
21 December 2023
21 December 2023
21 December 2023
20 December 2023
20 December 2023
20 December 2023
20 December 2023
20 December 2023
19 December 2023
18 December 2023
18 December 2023
18 December 2023
18 December 2023
18 December 2023
18 December 2023
17 December 2023
17 December 2023
17 December 2023
17 December 2023
15 December 2023
14 December 2023
14 December 2023
13 December 2023
13 December 2023
61. Hamas rejects Israeli proposals for 7day ceasefire and hostage exchange
60. Biden says he did not ask for ceasefire in call with Israels Netanyahu
59. Did not ask for ceasefire in Gaza Biden after phone call with Netanyahu
58. Humanitarian ceasefire only way to end Gaza 039nightmare039 Guterres
ption of UN resolution to expedite humanitarian aid to Gaza an important but insufficient step
aza war updates Residents fleeing areas of central Gaza groups criticize UN resolution
Israeli airstrikes kill dozens more Palestinians across the Gaza Strip Rafah
. Hamas leader visits Egypt for Gaza ceasefire and hostage release talks
ates Israeli airstrikes hit Gaza and UN says half a million people there are starving
52. Israeli PM vows to fight until victory despite ceasefire efforts
51. Netanyahu rules out Gaza ceasefire before elimination of Hamas
ident Isaac Herzog Open To 2nd Ceasefire In Gaza To Free Remaining Hostages
APUP 3Hamas leader visits Egypt amid intensive talks on new ceasefire
s War News Day 75 Live Updates Hamas chief arrives in Cairo for ceasefire talks
Hamas Chief Ismail Haniyeh In Egypt Today For Gaza Ceasefire Talks
ouncil Postpones Vote On Gaza Ceasefire Resolution For Humanitarian Aid Access
US stands between massacre and ceasefire in Gaza Turkish FM Fidan
lians make up 61 of Gaza deaths from airstrikes Israeli study finds
Palestine conflict India votes for UNGA resolution for Gaza ceasefire
led by IDF used food to create SOS sign Israel and Hamas open to another temporary ceasefire
41. Israel strikes Gaza as pressure grows for ceasefire
ecurity Council To Vote On New Call For Urgent Ceasefire In Gaza
n sources say Israel Hamas open to ceasefire disagreements remain
38. European diplomacy steps up calls for Gaza ceasefire
7. UK and Germany call for sustainable ceasefire in Gaza
Ks decision not to vote for UN resolution demanding ceasefire in Gaza as it happened
raelGaza war UK and Germany call for sustainable ceasefire
ess Harder In Its War With Hamas After US Vetoes Gaza Ceasefire Bid
aders increasingly back a humanitarian ceasefire in Gaza
s meets after Canada votes for IsraelHamas ceasefire at UN
call for Gaza ceasefire IsraelUS divided over casualties Top points
anada New Zealand back sustainable ceasefire in Gaza
a could go on for months despite international calls for a ceasefire