Introduction

This method aims to extract meaningful events from the Reuters news dataset, analyse their relationships, and visualise them as a causal graph. By connecting events using causal relationships and analysing their sentiment, this method offers an interactive and intuitive way to explore how news stories relate. The final visualisation highlights these connections and includes additional features like sentiment analysis and metadata for deeper insights into the dataset.

Overview of the Approach

This project uses Python to process the Reuters dataset and extract events. It relies on Natural Language Processing (NLP) tools like spaCy to identify actions, subjects, and objects in sentences and TextBlob for sentiment analysis. NetworkX builds a directed graph (causal graph), where nodes represent events and edges represent causal relationships. Finally, the graph is visualised using Plotly for interactive exploration.

Steps in the Process

The process involves the following steps:

1. **Loading and Preprocessing the Reuters Dataset**
   The Reuters dataset is loaded using nltk. Since the dataset contains many articles, we limit processing to the first 5,000 lines to keep computations manageable. Each article is stored with its content, and placeholder dates are assigned since the dataset lacks explicit temporal metadata.
2. **Event Extraction**
   Using spaCy, events are extracted from the text. Each sentence in the dataset is analysed to identify the main action (verb), the agent acting (subject), and the entities affected (objects). This structured information becomes the foundation for building the graph.
   For example:
   - Sentence: "The oil spill caused damage to the coastline."
   - Extracted Event:
     - Verb: "caused"
     - Subject: "oil spill"
     - Objects: ["damage", "coastline"]
3. **Coreference Resolution**
   To ensure consistency in the graph, coreferences like "it" or synonyms like "spill" are resolved to their most meaningful terms, such as "oil spill." This step helps unify references and avoids redundant nodes in the graph.
4. **Sentiment Analysis**
   Each event is analysed for sentiment using TextBlob. Events are classified as positive, negative, or neutral based on their polarity. For example:
   - "The tourism industry recovered" → Positive
   - "The oil spill caused damage" → Negative
5. Sentiment is stored as metadata and used to colour the nodes in the graph:
   - Green: Positive

    - Red: Negative
    - Orange: Neutral
6. **Causal Relationship Detection**
   Events are connected by analysing their sentences for causal keywords like "caused by," "due to," or "led to." If a sentence contains such a keyword and refers to another event, a causal link (edge) is created.
   For example:
    - Event A: "The oil spill caused damage to the coastline."
    - Event B: "The damage led to a decline in tourism."
    - Relationship: A → B (labeled "causes")
7. **Graph Construction**
   A directed graph is built using NetworkX. Each event becomes a node, and causal relationships form directed edges. Metadata such as event sentences, dates, and sentiment is stored in the graph for interactivity.
8. **Graph Visualization**
   The graph is visualised using Plotly. Nodes are positioned using the Kamada-Kawai layout for clear separation and minimal overlap. Nodes are colour-coded based on sentiment, and hovering over a node reveals its details, including the sentence, sentiment, and date. Edges are labelled with the causal relationship, providing a clear understanding of how events are interconnected.

Visualisation Features

The final graph offers the following features:

- **Interactive Nodes:**
  Each node represents an event and displays its details when hovered over, including the sentence, sentiment, and date. Nodes are coloured:
    - **Green:** Positive sentiment (e.g., "Tourism industry recovered.")
    - **Red:** Negative sentiment (e.g., "Oil spill caused damage.")
    - **Orange:** Neutral sentiment (e.g., "Government plans cleanup.")
- **Labeled Edges:**
  Directed edges connect events with causal relationships, labelled with terms like "causes" or "leads to."
- **Readable Layout:**
  The Kamada-Kawai layout ensures that nodes are spaced evenly, avoiding clutter and improving readability.
- **Scalability:**
  Although the graph currently processes 100 events for readability, the underlying code can handle larger datasets with some adjustments.

Results

The final graph successfully visualises events and their causal relationships from the Reuters dataset. For example:

- **Node:** "The oil spill caused damage to the coastline." (Negative sentiment, Red)
- **Edge:** Labeled "causes," connecting to another node: "The coastline damage led to a decline in tourism."

Users can interact with the graph to explore event relationships, analyse sentiment, and understand the flow of causality in the dataset. Sentiment analysis adds an extra layer of insight, highlighting the emotional tone of each event.

Code Highlights:

Key components of the code include:

Event Extraction:
```
def extract_events(text):
    doc = nlp(text)
    events = []
    for sent in doc.sents:
        verbs = [token for token in sent if token.pos_ == 'VERB']
        for verb in verbs:
            subject = ''
            objects = []
            ...
            events.append(event)
    return events
```

1. This function extracts the verbs, subjects, and objects from sentences to structure events.

Sentiment Analysis:
```
def get_sentiment(text):
    analysis = TextBlob(text)
    if analysis.sentiment.polarity > 0:
        return 'positive'
    elif analysis.sentiment.polarity < 0:
        return 'negative'
    else:
        return 'neutral'
```

2. Sentiment is calculated using TextBlob and assigned to nodes for visualisation.

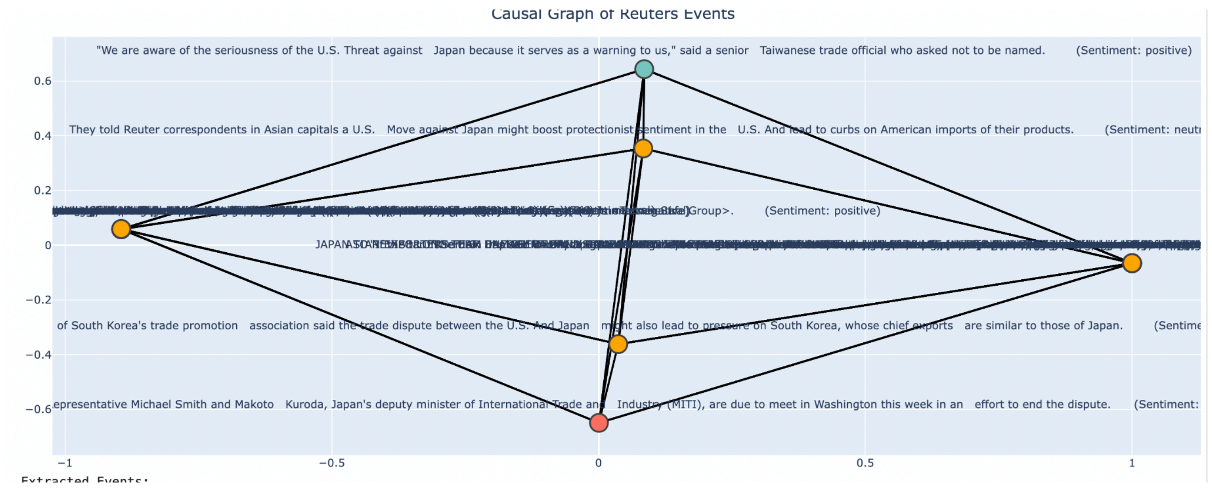
Graph Construction:
```
for event in all_events_sorted:
    event_id = hash((event['sentence'], event['date']))
    G.add_node(event_id, label=event['sentence'], date=event['date'],
sentiment=event['sentiment'])

for source, target in causal_relations:
    G.add_edge(source_id, target_id, label='causes')
```

3. Nodes and edges are added to the graph with relevant metadata.

Visualization:
```
node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers+text',
    marker=dict(color=node_colors, size=20, line_width=2),
    text=node_text
)
edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=2, color='#000000'),
    mode='lines'
)
fig = go.Figure(data=[edge_trace, node_trace])
fig.show()
```

4. The graph is visualized using Plotly with interactive nodes and edges.



"We are aware of the seriousness of the U.S. Threat against   Japan because it serves as a warning to us," said a senior   Taiwanese trade official who asked not to be named.   (Sentiment: positive)

They told Reuter correspondents in Asian capitals a U.S.   Move against Japan might boost protectionist sentiment in the   U.S. And lead to curbs on American imports of their products.   (Sentiment: neutr

[illegible] Group>.   (Sentiment: positive)

JAPAN [illegible]

of South Korea's trade promotion   association said the trade dispute between the U.S. And Japan   might also lead to pressure on South Korea, whose chief exports   are similar to those of Japan.   (Sentime

epresentative Michael Smith and Makoto   Kuroda, Japan's deputy minister of International Trade and   Industry (MITI), are due to meet in Washington this week in an   effort to end the dispute.   (Sentiment:

```
Extracted Events:
```