Development of a Question Answering System and Timeline Summarization Plot for the Israel-Hamas War

Introduction

This document details the thought process and the implementation steps for creating a rudimentary Question Answering (QA) system and generating a timeline summarization plot for major events during the Israel-Hamas war. The project uses several Python libraries, including `transformers` for language models and `plotnine` for visualization.

Objectives

1. **Question Answering System**: Develop a system that can accurately answer questions related to the Israel-Hamas war by extracting relevant information from a set of articles.
2. **Timeline Summarization**: Create a plot or list summarizing the major events of the Israel-Hamas war during the specified period in the JSON file.

Approach and Implementation

Question Answering System

Selection of Tools

- **Language Model**: We chose `DistilBERT`, a distilled version of `BERT`, because it provides a good balance between efficiency and performance for the QA task.
- **Library**: The `transformers` library by Hugging Face was selected for loading and running the language models due to its wide acceptance and ease of use.
- **Context Creation**: Relevant articles were filtered based on specific keywords related to the Israel-Hamas war.

Code Explanation

Dependencies and Imports:

```
python
Copy code
import json
import re
from datetime import datetime
import pandas as pd
from transformers import pipeline
from plotnine import ggplot, aes, geom_line, geom_point, labs,
theme_minimal, element_text, theme
import matplotlib.pyplot as plt
```

- `json`, `re`, and `datetime` are standard libraries for handling JSON data, regular expressions, and date-time objects, respectively.
- `pandas` is used for data manipulation.
- `transformers` provides tools for loading and running pre-trained models.
- `plotnine` and `matplotlib` are used for visualization.

Text Cleaning and Filtering:

```
# Function to clean text
def clean_text(text):
    text = re.sub(r'[^A-Za-z0-9\s]', '', text)
    text = ' '.join(text.split())
    return text.lower()

# Function to identify relevant articles
def is_relevant(article):
    keywords = ['israel', 'hamas', 'gaza', 'palestine']
    return any(keyword in article['articleBody'].lower() for keyword in
keywords)
```

- `clean_text`: Removes non-alphanumeric characters and normalizes whitespace, making text easier to process.
- `is_relevant`: Checks if the article contains keywords related to the Israel-Hamas conflict to filter relevant articles.

Loading and Preprocessing Articles:

```
# Load and preprocess articles
def load_articles(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        articles = json.load(file)
    relevant_articles = [article for article in articles if
is_relevant(article)]
    return relevant_articles

# Select relevant context from articles
def create_context(articles):
    context = ' '.join([clean_text(article['articleBody']) for article in
articles])
    return context
```

- `load_articles`: Loads articles from a JSON file and filters them based on relevance.
- `create_context`: Combines the text of relevant articles into a single context string for the QA system.

QA System Implementation:

```
# Initialize QA pipeline
qa_pipeline = pipeline("question-answering", model="distilbert-base-
uncased-distilled-squad")

# Sample question
question = "What happened at the Al-Shifa Hospital?"
answer = get_answer(question, context, qa_pipeline)
print(f"Answer: {answer}")
```

- The QA pipeline uses `DistilBERT` to answer questions based on the provided context.

Timeline Summarization

Extraction and Summarization of Events

Event Extraction:

```
# Function to extract events with dates
def extract_events(articles):
    events = []
    date_regex = r'\b\d{1,2} \w+ \d{4}\b'
    for article in articles:
        dates = re.findall(date_regex, article['articleBody'])
        for date in dates:
            try:
                event_date = datetime.strptime(date, '%d %B %Y')
                events.append((event_date, article['articleBody']))
            except ValueError:
                continue
    return events
```

- `extract_events`: Extracts dates and corresponding events from the articles. The regular expression captures dates in the format "day month year".

Sorting and Summarizing:

```
# Extract and sort events
events = extract_events(articles)
events.sort(key=lambda x: x[0])

# Convert events to DataFrame
df_events = pd.DataFrame(events, columns=['Date', 'Event'])

# Summarize events by counting occurrences per date
event_counts = df_events.groupby('Date').size().reset_index(name='Counts')
```

- Events are sorted by date, and a DataFrame is created to count the number of events per date.

Plotting the Timeline:

```
# Create the plot
plot = (ggplot(event_counts, aes(x='Date', y='Counts'))
        + geom_line(size=1.2)
        + geom_point(size=2)
        + labs(title='Timeline of Israel-Hamas War Events', x='Date',
y='Number of Events')
        + theme_minimal()
        + theme(axis_text_x=element_text(rotation=45, hjust=1)))

# Display the plot
print(plot)

# Alternatively, to display the plot using matplotlib (if needed)
plt.figure(figsize=(12, 6))
plt.plot(event_counts['Date'], event_counts['Counts'], marker='o',
linestyle='-')
plt.title('Timeline of Israel-Hamas War Events')
plt.xlabel('Date')
plt.ylabel('Number of Events')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

- The timeline plot shows the number of events per date, with points and lines indicating the timeline's key events.

`requirements.txt`

```
transformers==4.33.2
torch==2.0.1
pandas==1.5.3
plotnine==0.12.1
matplotlib==3.7.2
```

This file lists all the necessary packages for running the QA system and generating the timeline plot.

Testing and Validation

- **Testing**: The system was tested by running various queries related to the Israel-Hamas war to ensure it provides accurate and relevant answers.
- **Validation**: The timeline was validated by checking the accuracy of event dates and the completeness of the summarization.

Conclusion

The developed QA system effectively answers questions related to the Israel-Hamas war by leveraging a pre-trained language model and a well-prepared context from relevant articles. The timeline summarization provides a clear visualization of the major events, facilitating a better understanding of the conflict's progression.This project demonstrates the use of modern NLP techniques and visualization tools to analyze and present complex information comprehensively and effectively.