Final Project - NLP Chicago Carjackings Shiny Dashboard

Surya Hardiansyah and Astari Raihanah

2025-02-10

```
import requests
import pandas as pd
from spacytextblob.spacytextblob import SpacyTextBlob
import spacy
import time  # To avoid hitting rate limits

# Load spaCy with TextBlob for Sentiment Analysis
nlp = spacy.load("en_core_web_sm")
nlp.add_pipe("spacytextblob")

# Function to fetch paginated results from SerpAPI
def fetch_serpapi_results(query, api_key, num_results=100, start=0):
    url = "https://serpapi.com/search"
    params = {
        "q": query,
        "hl": "en",  # Language: English
        "gl": "us",  # Country: US
        "api_key": api_key,
        "num": num_results,  # Max number of results per page
        "start": start,  # Offset for pagination
    }
    response = requests.get(url, params=params)
    if response.status_code == 200:
        return response.json().get("organic_results", [])
    else:
        print(f"Error: {response.status_code}, Message: {response.text}")
        return []

# Function to analyze sentiment
```

```python
def analyze_sentiment(text):
    doc = nlp(text)
    return doc._.blob.polarity, doc._.blob.subjectivity

# SerpAPI key and expanded queries
api_key = "a294a642b519834b1905e86841770361e944c1b48eb3c597bc2f958893f31a4b"
queries = [
    "Chicago car insurance policy",
    "auto insurance Chicago",
    "carjacking auto insurance Chicago",
    "auto insurance Chicago car theft",
    "Chicago carjacking",
    "Chicago car theft"
]

# Collect results
data = []
for query in queries:
    total_observations = 0
    start = 0  # Start pagination from 0
    while total_observations < 500:  # Target total of 500 results
        results = fetch_serpapi_results(query, api_key, num_results=100,
 ↪  start=start)
        if not results:
            break  # Stop if no more results
        for result in results:
            title = result.get("title", "")
            link = result.get("link", "")
            snippet = result.get("snippet", "")
            date = result.get("date", "")  # Published date if available
            if snippet:  # Ensure snippet exists
                polarity, subjectivity = analyze_sentiment(snippet)
                data.append({
                    "query": query,
                    "title": title,
                    "link": link,
                    "date_published": date,
                    "snippet": snippet,
                    "polarity": polarity,
                    "subjectivity": subjectivity
                })
        total_observations += len(results)
        start += 100  # Move to the next page
        time.sleep(2)  # Avoid hitting rate limits

# Create DataFrame
df = pd.DataFrame(data)

# Save results to CSV
df.to_csv("serpapi_expanded_results_2.csv", index=False)

print(f"Total results collected: {len(df)}")
```

```
Total results collected: 999
```

```
import pandas as pd
import altair as alt

# Load the data from the CSV file into a Pandas DataFrame
data = pd.read_csv("serpapi_expanded_results_2.csv")

# Ensure Altair can render without row limits
alt.data_transformers.disable_max_rows()

# Average Polarity by Query
avg_polarity_chart = (
    alt.Chart(data)
    .mark_bar(color="steelblue")
    .encode(
        x=alt.X("query:N", title="Query", sort="-y"),
        y=alt.Y("mean(polarity):Q", title="Average Polarity"),
        tooltip=["query", "mean(polarity):Q"]
    )
    .properties(title="Average Sentiment Polarity by Query", width=600,
 ↪  height=400)
)

# Add text labels for the bars
text = avg_polarity_chart.mark_text(
    align='center',
    baseline='middle',
    dy=-10  # Adjust position of the text above the bars
).encode(
    text=alt.Text("mean(polarity):Q", format=".2f")
)

# Combine bar chart with text labels
avg_polarity_with_labels = avg_polarity_chart + text

# Average Subjectivity by Query
avg_subjectivity_chart = (
    alt.Chart(data)
    .mark_bar(color="orange")
    .encode(
        x=alt.X("query:N", title="Query", sort="-y"),
        y=alt.Y("mean(subjectivity):Q", title="Average Subjectivity"),
        tooltip=["query", "mean(subjectivity):Q"]
    )
    .properties(title="Average Subjectivity by Query", width=600, height=400)
)

# Add text labels for the bars
text_subjectivity = avg_subjectivity_chart.mark_text(
    align='center',
    baseline='middle',
    dy=-10  # Adjust position of the text above the bars
).encode(
    text=alt.Text("mean(subjectivity):Q", format=".2f")
)

# Combine bar chart with text labels
avg_subjectivity_with_labels = avg_subjectivity_chart + text_subjectivity

# Display the charts
avg_polarity_with_labels | avg_subjectivity_with_labels
```

```
alt.HConcatChart(...)
```