Parallel Text Processing - Sentiment Analysis

Author: Srinivas
Email: srinivasacademics@gmail.com
LinkedIn: LinkedIn

Project Links

- **GitHub Repository:** GitHub
- **Certificate:** <CERTIFICATE_LINK>

Abstract

This project implements a complete end-to-end parallel text processing pipeline designed to handle large-scale tweet datasets efficiently. It integrates data ingestion, deep cleaning, NLP normalization, multiprocessing-based analysis, sentiment classification using both TextBlob and a Hugging Face transformer model, SQLite storage, automated email summaries, CSV export, and a Streamlit-based interactive interface. All work is documented across milestones using Jupyter notebooks and PDF reports.

Problem Statement

Processing text sequentially leads to slow execution for large datasets. This project was developed to:

- accelerate text preprocessing and sentiment analysis using multiprocessing,
- compare a rule-based model (TextBlob) with a transformer-based LLM,
- build a complete automated workflow involving SQLite, CSV export, and email reporting,
- provide an interactive Streamlit UI to demonstrate the complete system.

Objectives

- Build a clean end-to-end text analysis pipeline.
- Use NLP techniques for text cleaning and normalization.
- Implement multiprocessing for faster processing.
- Integrate TextBlob and Hugging Face sentiment models.
- Store and export results using SQLite/CSV.
- Provide automated email summaries.
- Develop a Streamlit UI for demonstration.

Tech Stack

Python, Pandas, NumPy, spaCy, TextBlob, Hugging Face Transformers, SQLite3, Multiprocessing, Streamlit, Matplotlib, Seaborn.

Dataset

Tweets dataset (approx. 27,000 rows). A 1,000-tweet subset was used for performance tests.

Main fields: author, content, country, date_time, id, language, latitude, longitude, number_of_likes, number_of_shares.

System Architecture Overview

1. Data ingestion (CSV -> Pandas -> SQLite)
2. Cleaning and normalization (regex, emoji removal, spaCy lemmatization)
3. Parallel processing (multiprocessing)
4. Sentiment analysis (TextBlob and Hugging Face)
5. Storage (SQLite)
6. Export (CSV)
7. Email summary generation
8. Streamlit UI

Milestones - Tasks and Work Completed

Milestone 1 - Data Ingestion & Cleaning

Tasks Completed

- Imported dataset using Pandas.
- Stored dataset into SQLite database.
- Implemented text cleaning (lowercasing, regex-based cleaning, emoji removal).
- Implemented spaCy-based tokenization, lemmatization, stopword removal.
- Implemented parallel word count using multiprocessing.

Minimal Original Code (from SRINIVAS_MILESTONE-1.ipynb)

Load CSV -> SQLite

```
import pandas as pd, sqlite3

df = pd.read_csv("tweets.csv")
conn = sqlite3.connect("tweets.db")
df.to_sql("tweets", conn, if_exists="replace", index=False)
```

Cleaning Function

```
import re

def clean_text(x):
    x = x.lower()
    x = re.sub(r'[^\x00-\x7F]+',' ', x)
    x = re.sub(r'[^\w\s]','', x)
    return x
```

Lemmatization

```
import spacy
nlp = spacy.load("en_core_web_sm")

def normalize(text):
    doc = nlp(text)
    return " ".join(
        t.lemma_ for t in doc if t.is_alpha and not t.is_stop
    )
```

Parallel Word Count

```
from multiprocessing import Pool
from collections import Counter

def count_chunk(chunk):
    c = Counter()
    for line in chunk:
        c.update(line.split())
    return c
```

Files Delivered (Look Appendix):

- Srinivas_Milestone-1.pdf
- SRINIVAS_MILESTONE-1.ipynb

Milestone 2 - TextBlob Sentiment & Email Summary

Tasks Completed

- Implemented TextBlob sentiment polarity and labeling.
- Saved sentiment results into SQLite.
- Exported sentiment results to CSV.
- Implemented automated email summarization with sentiment counts.

Minimal Original Code (from SRINIVAS_MILESTONE_2.ipynb)

TextBlob Sentiment

```
from textblob import TextBlob

def tb_sentiment(text):
    polarity = TextBlob(text).sentiment.polarity
    label = "positive" if polarity>0 else "negative" if polarity<0 else "
       neutral"
    return polarity, label
```

Save Sentiment into SQLite

```
import sqlite3

conn = sqlite3.connect("tweets.db")
cursor = conn.cursor()

cursor.execute("""
UPDATE tweets SET
```

```
  textblob_polarity=?,
  textblob_label=?
WHERE id=?
""", (pol, label, tweet_id))

conn.commit()
```

Email Summary

```
import smtplib, ssl
from email.message import EmailMessage

msg = EmailMessage()
msg["Subject"] = "Sentiment Summary"
msg["From"] = sender
msg["To"] = receiver
msg.set_content(summary_text)

with smtplib.SMTP_SSL("smtp.gmail.com",465) as smtp:
    smtp.login(sender, app_password)
    smtp.send_message(msg)
```

Files Delivered (Look Appendix):

- SRINIVAS MILESTONE 2.pdf
- SRINIVAS_MILESTONE_2.ipynb

Milestone 3 - Hugging Face Model & Parallel Inference

Tasks Completed

- Integrated Hugging Face model `cardiffnlp/twitter-roberta-base-sentiment-latest`.
- Implemented sequential and parallel inference.
- Measured performance: 141.81s vs 113.92s.
- Generated confusion matrix comparing TextBlob and Hugging Face.

Minimal Original Code (from MILESTONE-3-SRINIVAS.ipynb)

Sequential HF Inference

```
from transformers import pipeline
hf = pipeline("sentiment-analysis",
              model="cardiffnlp/twitter-roberta-base-sentiment-latest")

def hf_single(text):
    return hf(text)[0]
```

Parallel HF Inference

```
def worker(texts):
    model = pipeline("sentiment-analysis",
                     model="cardiffnlp/twitter-roberta-base-sentiment-
                        latest")
    return [model(t)[0] for t in texts]
```

Files Delivered (Look Appendix):

- SRINIVAS MILESTONE 3.docx
- MILESTONE-3-SRINIVAS.ipynb

Milestone 4 - Streamlit UI

Tasks Completed

- Built Streamlit interface for:
  - single-text analysis,
  - batch CSV/XLSX upload,
  - sequential and parallel execution,
  - downloading results,
  - emailing summary.
- Integrated all pipeline code into UI.

Minimal Original Code (from IntershipProject.ipynb)

```
import streamlit as st
import pandas as pd

st.title("Parallel Text Processor - Sentiment Analysis")

text = st.text_area("Enter text:")
if st.button("Analyze"):
    st.write(hf_single(text))
```

```
upload = st.file_uploader("Upload CSV", type=["csv","xlsx"])
if upload:
    df = pd.read_csv(upload)
    if st.button("Run Parallel"):
        result = run_parallel(df)
        st.dataframe(result.head())
```

Files Delivered (Look Appendix):

- IntershipProject.ipynb

Key Results

- Total dataset processed: 27k tweets.
- Parallel speedup: 141.81s -> 113.92s (1.24x).
- 370 disagreements found between TextBlob and Hugging Face.
- UI successfully integrated with full pipeline.

Uploaded Files

- Srinivas_Milestone-1.pdf
- SRINIVAS_MILESTONE-1.ipynb
- SRINIVAS MILESTONE 2.pdf
- SRINIVAS_MILESTONE_2.ipynb

- SRINIVAS MILESTONE 3.docx
- MILESTONE-3-SRINIVAS.ipynb
- IntershipProject.ipynb
- MILESTONE-4.md

Appendix - Code Sections Not Included (Too Lengthy)

The following parts were intentionally excluded to maintain readable documentation:

- Clink into this Github for each milestone code
- Full spaCy token-by-token debug outputs.
- Full dataframe printing in Milestone 1 and 2 notebooks.
- Entire plotting code (matplotlib/seaborn).
- Complete Streamlit UI (200+ lines).
- Full detailed confusion matrix prints.

All these components are fully available in the uploaded notebooks.