Milestone 1 Report

Author: Srinivas
Email: srinivasacademics@gmail.com

---

1. Key Objectives

1. Load and store raw data in a database (**SQLite3**).
2. Perform **text cleaning and normalization**.
3. Implement **tokenization, lemmatization, and stopword removal** using spaCy.
4. Use **multiprocessing** to calculate word frequencies.

---

2. Data Information

- **Data Chosen:** Small
- **Data Stored Format:** CSV File
- **Database Used:** SQLite3
- **Data Handling:** Data was stored in the SQLite3 database and successfully fetched for preprocessing and analysis.

---

3. Dataset

- **Source:** Tweets dataset in CSV format from Kaggle

Attributes:

Column Name	Data Type
author	TEXT
content	TEXT
country	TEXT
date_time	TEXT
id	REAL
language	TEXT
latitude	REAL
longitude	REAL
number_of_likes	INTEGER
number_of_shares	INTEGER

---

4. Implementation

4.1 Loading Data into SQLite3

```
import pandas as pd
import sqlite3
import os
print("path:", os.getcwd())
df = pd.read_csv("tweets.csv", encoding="utf-8")
print(df.shape)
print(df.head())
conn = sqlite3.connect("tweets.db")
cursor = conn.cursor()
df.to_sql("tweets", conn, if_exists="replace", index=False)
print("CSV file stored successfully into SQLite3 database!")
cursor.execute("SELECT COUNT(*) FROM tweets")
print("Total rows stored:", cursor.fetchone()[0])
cursor.execute("SELECT * FROM tweets LIMIT 5")
rows = cursor.fetchall()
for row in rows:
    print(row)
```

4.2 Text Preprocessing

Steps performed:

- Converted text to lowercase
- Removed unwanted characters (\xa0, ?, #)
- Removed punctuation
- Tokenized and lemmatized words
- Removed stopwords using spaCy

```
import sqlite3
import pandas as pd
import spacy
```

```
from spacy.lang.en.stop_words import STOP_WORDS

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

df = pd.read_sql_query("SELECT * FROM tweets", conn)

print(df.head(2))

df['content_clean'] = df['content'].str.lower()

df['content_clean'] = df['content_clean'].str.replace('\xa0', ' ')

df['content_clean'] = df['content_clean'].str.replace('?', ' ')

df['content_clean'] = df['content_clean'].str.replace('#', ' ')

df['content_clean'] = df['content_clean'].str.replace(r'[^\w\s]', '', regex=True)

print(df[['content', 'content_clean']].head(5))

desc = df.content_clean.loc[0]

desc

nlp = spacy.load("en_core_web_sm", disable=["ner", "parser"])

doc = nlp(desc)

[token.lemma_ for token in doc if not token.is_stop][:10]

' '.join([token.lemma_ for token in doc if not token.is_stop])

def token_lemma_nonstop(text):

    doc = nlp(text.lower())

    return " ".join([token.lemma_ for token in doc if token.is_alpha and token.text not in
STOP_WORDS])

print(df[['content', 'content_clean']].head(5))

df['content_clean'] = df['content_clean'].apply(token_lemma_nonstop)

print(df[['content', 'content_clean']].head(5))
```

4.3 Word Frequency (with Multiprocessing)

- Used **multiprocessing** to parallelize word counting
- Extracted **Top 20 most frequent words**

```
from collections import Counter

from multiprocessing import Pool, cpu_count

def count_words_in_chunk(texts):

    counter = Counter()
```

```
    for text in texts:
        words = text.split()
        counter.update(words)
    return counter
def chunks(lst, n):
    k, m = divmod(len(lst), n)
    return [lst[i*k + min(i, m):(i+1)*k + min(i+1, m)] for i in range(n)]
if __name__ == "__main__":
    texts = df['content_clean'].tolist()
    num_cores = cpu_count()
    text_chunks = chunks(texts, num_cores)
    with Pool(num_cores) as pool:
     results = pool.map(count_words_in_chunk, text_chunks)
     total_counts = Counter()
    for r in results:
        total_counts.update(r)
    top_words = total_counts.most_common(20)
   print("Top 20 words and their counts:\n")
    for word, count in top_words:
        print(f"{word:15} : {count}")
```

---

5. Results

- The dataset was **successfully stored** in SQLite3.
- Text was **cleaned, normalized, tokenized, and lemmatized**.
- **Stopwords were removed** to retain only meaningful tokens.
- Using multiprocessing, the **top 20 frequent words** were extracted.
- .All the task in Milestone 1 has been done
