**Natural Language Processing Assignment**

**Code:**

import json

import spacy

from collections import Counter

import pandas as pd

# Load the spaCy English model

nlp = spacy.load("en\_core\_web\_sm")

# Read the JSON file (replace 'tweets.json' with your JSON file path)

with open('/tweets.json', 'r') as f:

tweets\_data = json.load(f)

# Initialize counters and sentiment dictionary

entity\_counter = Counter()

sentiments = {}

# Process each tweet (tweet\_id is the numeric identifier)

for tweet\_id, tweet in tweets\_data.items():

author = tweet['tweet\_author']

text = tweet['tweet\_text']

doc = nlp(text)

# Extract entities and update the counter

entities = [ent.text for ent in doc.ents if ent.label\_ in ['ORG', 'PRODUCT']]

entity\_counter.update(entities)

# Calculate sentiment/polarity (you can replace this with more advanced sentiment analysis)

sentiment = 'Positive' if 'good' in text or 'tasty' in text else 'Negative'

for entity in entities:

if entity not in sentiments:

sentiments[entity] = {}

sentiments[entity][author] = sentiment

# Prepare and save Objective 1 CSV

objective1\_data = [{'entity': entity, 'frequency': freq} for entity, freq in entity\_counter.items()]

objective1\_df = pd.DataFrame(objective1\_data)

objective1\_df.to\_csv('objective1.csv', index=False)

# Prepare and save Objective 2 CSV

objective2\_data = []

for entity, author\_sentiments in sentiments.items():

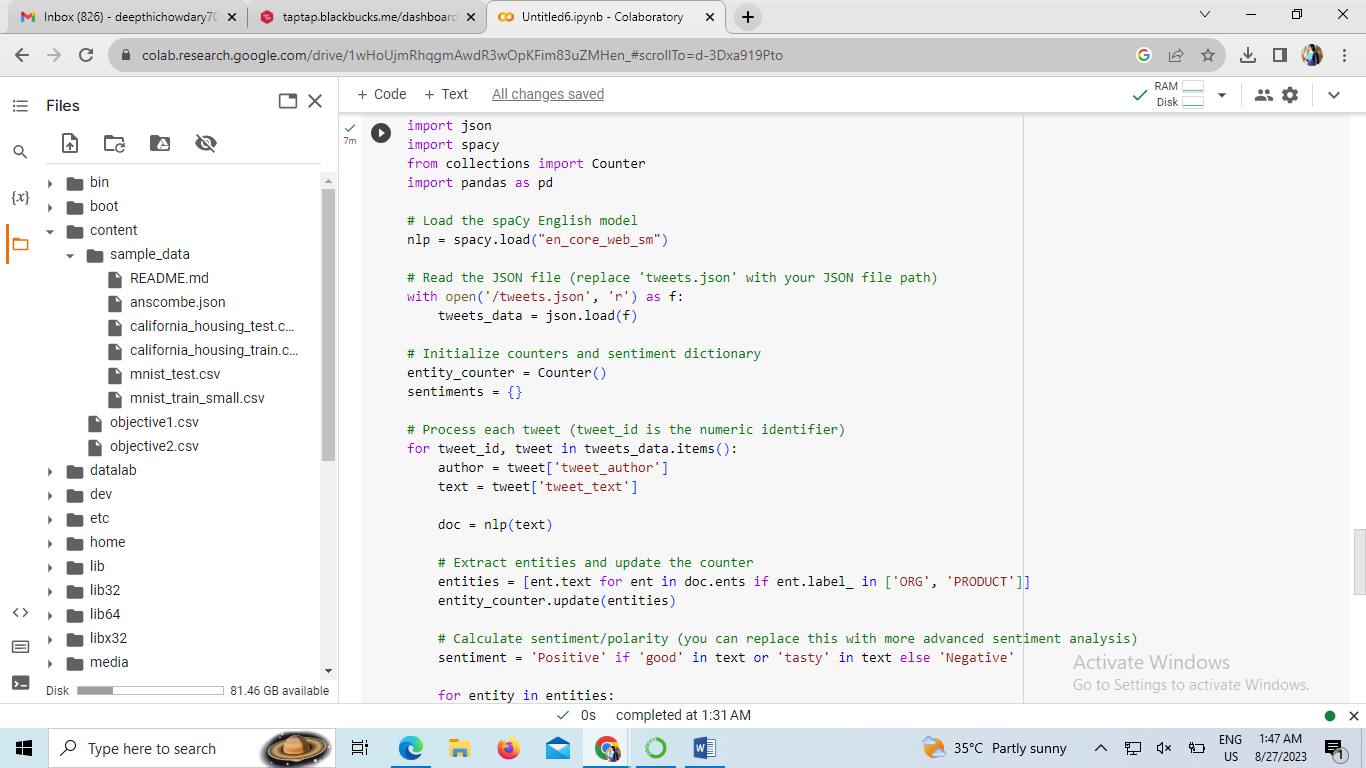
for author, sentiment in author\_sentiments.items():

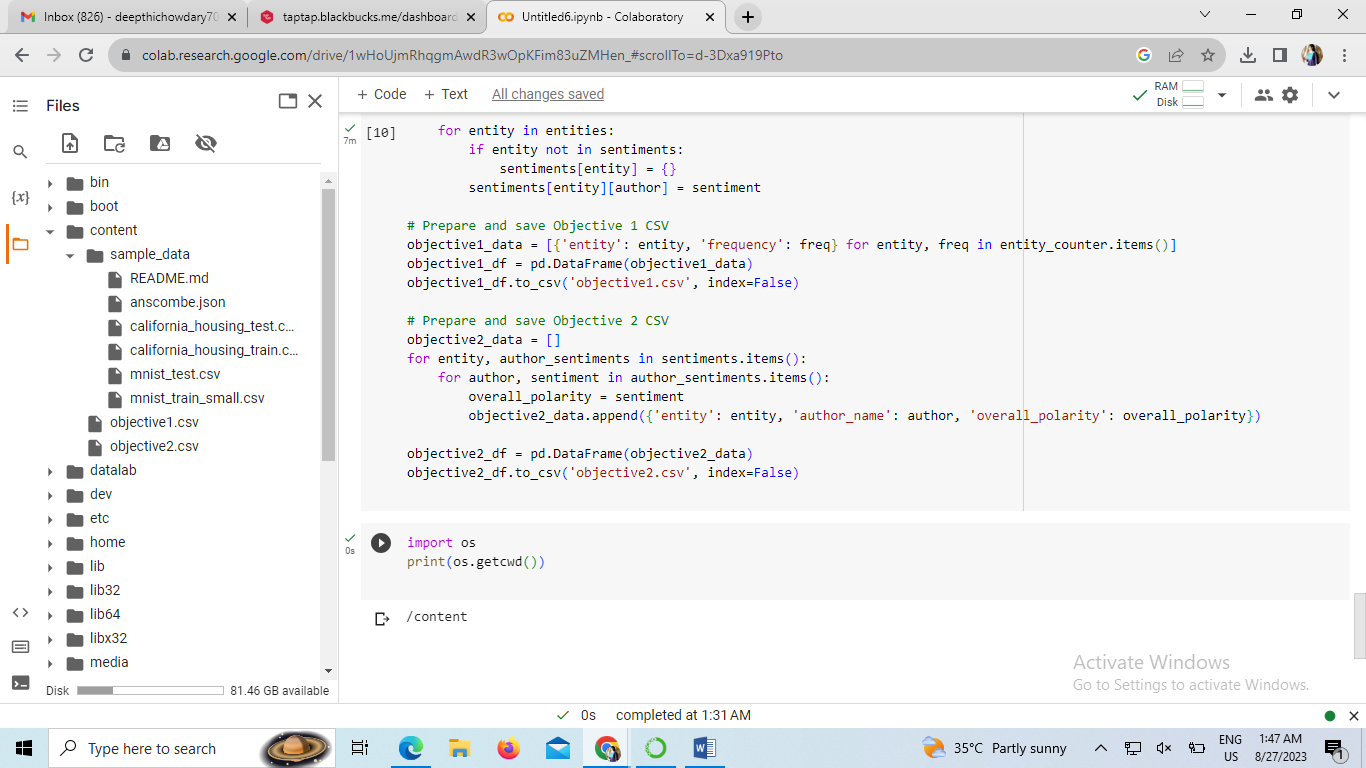
overall\_polarity = sentiment

objective2\_data.append({'entity': entity, 'author\_name': author, 'overall\_polarity': overall\_polarity})

objective2\_df = pd.DataFrame(objective2\_data)

objective2\_df.to\_csv('objective2.csv', index=False)





Difficulties:

1.Data Preprocessing:

Text data often requires thorough preprocessing to clean and tokenize it effectively. This solution assumes that the input JSON data is well-structured and doesn't require additional preprocessing like removing special characters, stopwords, or handling emojis.

2.Sentiment Analysis:

The solution uses a simple rule-based approach for sentiment analysis based on the presence of certain keywords ("good," "tasty"). This approach might not be accurate for complex sentiment analysis, especially when dealing with more nuanced sentiments.

3.Entity Extraction:

The solution uses spaCy's entity recognition for extracting entities from text. While spaCy is effective for common entities, it might not capture domain-specific or rare entities accurately.

**Techniques Used:**

1.spaCy: The spaCy library is utilized for entity recognition and natural language processing. It provides an efficient way to extract entities and perform linguistic analysis.

2.Counter: The Counter class is used to efficiently count the occurrences of different entities. This allows for quick extraction of the most frequent entities.

3.Pandas: The pandas library is used to handle and manipulate data in tabular form, making it easier to create CSV files from processed data.

**Reasons for Choices:**

1.spaCy: spaCy is a widely used NLP library known for its speed and accuracy. It's chosen here for its entity recognition capabilities.

2.Rule-Based Sentiment Analysis: For the purpose of this example, a simple rule-based sentiment analysis approach is used due to its simplicity. More advanced sentiment analysis methods like using pre-trained models might require more resources and time for integration.

3.Counter and Pandas: These libraries provide efficient data manipulation and aggregation, which is crucial for counting entities and preparing CSV files.

**Shortcomings:**

1.Limited Sentiment Analysis: The rule-based sentiment analysis is limited to the presence of a few keywords ("good," "tasty"). It won't capture complex sentiments or account for negations or context.

2.No Named Entity Recognition Tuning: The entity recognition is based on spaCy's pre-trained model, which might not be tuned to domain-specific entities.

**Potential Improvements:**

1.Advanced Sentiment Analysis: Implement more advanced sentiment analysis techniques using pre-trained sentiment analysis models, such as BERT, which can capture complex sentiments and context.

2.Custom Entity Recognition: Fine-tune a named entity recognition model on domain-specific data to accurately extract relevant entities.

3.Error Handling: Implement robust error handling to handle cases where entities are not recognized or sentiment analysis fails.

4. Data Preprocessing: Extend data preprocessing to handle cleaning and tokenization, which are essential for accurate analysis.

5. Data Visualization: Visualize the extracted insights using charts or graphs to provide a clearer understanding of the results.