**1. Importing Necessary Libraries**

python

import csv

import os

import time

from dotenv import load\_dotenv

from langchain\_core.runnables.base import Runnable

from langchain.prompts import PromptTemplate

from fireworks.client import Fireworks

These libraries are essential for handling CSV files, environment variables, time delays, and API calls.

**2. Loading Environment Variables**

python

load\_dotenv()

fireworks\_api\_key = os.getenv("FIREWORKS\_API\_KEY")

This part loads environment variables from a .env file, specifically retrieving the FIREWORKS\_API\_KEY for use with the Fireworks API.

**3. Initializing the Fireworks Client Wrapper**

python

class FireworksWrapper(Runnable):

def \_\_init\_\_(self, api\_key):

self.client = Fireworks(api\_key=api\_key)

def invoke(self, input\_data, \*args, \*\*kwargs):

if isinstance(input\_data, dict):

content = input\_data.get("content", "")

else:

content = str(input\_data)

response = self.client.chat.completions.create(

model="accounts/fireworks/models/llama-v3p1-8b-instruct",

messages=[{"role": "user", "content": content}]

)

return response.choices[0].message.content

llm = FireworksWrapper(api\_key=fireworks\_api\_key)

This class initializes the Fireworks client with the provided API key and defines the invoke method to send input data to the API and retrieve responses.

**4. Creating a Prompt Template for Sentiment Analysis**

python

template = """You are a sentiment analyst. Analyze the following statement and explain your choice of overall sentiment.

Statement: {content}

YOUR RESPONSE:"""

prompt\_template = PromptTemplate(input\_variables=["content"], template=template)

content\_chain = prompt\_template | llm

A prompt template is defined for sentiment analysis, which is then used to create a RunnableSequence that sends input data through the Fireworks API.

**5. Reading and Writing CSV Files**

python

input\_csv\_file\_path = 'tweets-labels.csv'

output\_csv\_file\_path = 'output.csv'

These variables define the paths to the input and output CSV files.

**6. Processing Batches of Rows**

python

def process\_batch(rows, csvwriter):

for row in rows:

content = row[0]

try:

review = content\_chain.invoke({"content": content}).strip('"')

csvwriter.writerow([review, row[1], row[0]])

print(review, row[1], row[0])

except Exception as e:

print(f"Error processing row: {row}, Error: {e}")

outfile.flush()

This function processes batches of rows, invoking the content chain for sentiment analysis and writing the results to the output CSV file.

**7. Delayed Processing**

python

def process\_data(rows):

for i, row in enumerate(rows):

if (i + 1) % 1500 == 0:

time.sleep(61)

This function introduces a delay every 1500 rows to respect rate limits.

**8. Reading and Writing CSV Files in Batches**

python

batch\_size = 1000

with open(input\_csv\_file\_path, mode='r', newline='', encoding='utf-8') as infile, \

open(output\_csv\_file\_path, mode='w', newline='', encoding='utf-8') as outfile:

csvreader = csv.reader(infile)

csvwriter = csv.writer(outfile)

csvwriter.writerow(['Review', 'Original Content', 'Label'])

batch = []

for row in csvreader:

batch.append(row)

if len(batch) >= batch\_size:

process\_batch(batch, csvwriter)

batch = []

time.sleep(70)

if batch:

process\_batch(batch, csvwriter)

This section reads the input CSV file, processes the data in batches, and writes the results to the output CSV file, including delays to respect rate limits.

This script is designed to perform sentiment analysis on a batch of text data from a CSV file, using the Fireworks API, and then write the results to a new CSV file.