see this is the current code where I am uploading a csv where aspect terms and related sentiments where given already. However, In actual I wanna upload a csv without a these aspect columns. This data would be generated at runtime using the llm that is given below given code. Earlier, I was processing the csv earlier then after processing through llm, I was uploading at gui. Now I know processing the whole csv will take time. So I wanna update the teachers in side bar as soon as the are processed by llm.

system\_prompt = """

    You are an expert in Aspect-Based Sentiment Analysis (ABSA). Your task is to analyze teacher reviews and extract aspect-specific information based on the following predefined aspect categories:

    - Teaching Pedagogy

    - Knowledge

    - Fair in Assessment

    - Experience

    - Behavior

    For each aspect that is \*\*explicitly or implicitly mentioned\*\* in the review:

    1. Identify and extract the \*\*aspect term(s) or phrase(s)\*\* used in the review that are related to the aspect category.

    2. Determine the \*\*sentiment polarity\*\* expressed toward that aspect. Choose one of: {Positive, Negative, Neutral}.

    If an aspect is not mentioned in the review, \*\*do not include it in the output\*\*.

    Return the output in a structured JSON format as follows:

    ```json

    {

    "Aspect Category": {

        "Aspect Terms": ["..."],

        "Extracted Phrase": "...",

        "Polarity": "..."

    },

    ...

    }

"""

import pandas as pd

import json

# Your existing Ollama call function

def ask\_ollama(input\_content, system\_prompt, model\_name="mistral"):

    response = ollama.chat(model=model\_name, messages=[

        {'role': 'system', 'content': system\_prompt},

        {'role': 'user', 'content': input\_content}

    ])

    response\_text = response['message']['content'].strip()

    return response\_text

# Define aspects and related columns

aspects = ["Teaching Pedagogy", "Knowledge", "Fair in Assessment", "Experience", "Behavior"]

term\_columns = [f"{aspect}\_terms" for aspect in aspects]

polarity\_columns = [f"{aspect}\_polarity" for aspect in aspects]

# Add empty columns

for col in term\_columns + polarity\_columns:

    if col not in df.columns:

        df[col] = ""

# Get indices to update

indices\_to\_process = df[df['Target'] == 'Teacher'].index

# Initialize batch buffer

batch\_records = []

batch\_size = 10

model\_name = 'mistral'  # Replace with your model name

for count, idx in enumerate(indices\_to\_process, start=1):

    feedback = df.at[idx, 'Comments']

    try:

        result\_json = ask\_ollama(feedback, system\_prompt)

        df.at[idx, "llm\_response"] = result\_json  # Save raw LLM response

        result\_dict = parse\_json\_safe(result\_json)

        if result\_dict:

            for aspect in aspects:

                if aspect in result\_dict:

                    df.at[idx, f"{aspect}\_terms"] = result\_dict[aspect].get("Extracted Phrase", "")

                    df.at[idx, f"{aspect}\_polarity"] = result\_dict[aspect].get("Polarity", "")

    except Exception as e:

        print(f"Error at index {idx}: {e}")

        continue

    if count % batch\_size == 0 or count == len(indices\_to\_process):

        df.loc[indices\_to\_process].to\_csv("Datasets/checkpoint\_feedback\_" + model\_name +"2.csv", index=False)

        print(f"Checkpoint saved after {count} rows.")