Intern Assignment4

November 25, 2024

Intern Assignment : 4 Done by: Prateek Kumar

[1]:
```
import os
import json
import pandas as pd
from dotenv import load_dotenv
from groq import Groq

# Load environment variables
load_dotenv()
GROQ_API_KEY = os.
 ↪getenv('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')

# Initialize Groq client
client = Groq(
    api_key="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",
)

def load_sdoh_codes(csv_path):
    """Load SDOH codes from CSV file"""
    try:
        sdoh_df = pd.read_csv(csv_path)
        # Create a dictionary mapping SDOH factors to their codes
        return dict(zip(sdoh_df['SDOH factor'].str.lower(), sdoh_df['Code']))
    except Exception as e:
        print(f"Error loading SDOH codes: {e}")
        return {}

def extract_patient_info(clinical_note):
    """Extract patient information using Groq LLM"""

    prompt = (
        "Extract and organize the following information from the clinical note␣
 ↪into JSON format:\n"
        "- Patient's full name\n"
        "- Complete address\n"
        "- Hospital name\n"
        "- List of allergies\n"
```

```
        "- List of major medical problems\n"
        "- List of social determinants of health (SDOH) factors (specifically␣
↪looking for: "
        "radiation exposure, workplace stress, environmental factors, housing␣
↪conditions, "
        "nutrition, healthcare access)\n\n"
        f"Clinical Note:\n{clinical_note}\n\n"
        "Format the response as a JSON object with these exact keys:\n"
        "{\n"
        '  "patient_name": "",\n'
        '  "address": "",\n'
        '  "hospital": "",\n'
        '  "allergies": [],\n'
        '  "medical_problems": [],\n'
        '  "sdoh_factors": []\n'
        "}\n"
        "Ensure all lists are properly formatted and all SDOH factors are in␣
↪lowercase."
    )

    try:
        response = client.chat.completions.create(
            messages=[{
                "role": "user",
                "content": prompt
            }],
            model="mixtral-8x7b-32768",
            temperature=0.1,
            max_tokens=1000
        )

        # Print the raw response for debugging
        print("Raw LLM Response:")
        print(response.choices[0].message.content)

        # Extract and parse JSON from response
        content = response.choices[0].message.content
        # Remove any potential markdown formatting
        if "```json" in content:
            content = content.split("```json")[1].split("```")[0]
        elif "```" in content:
            content = content.split("```")[1].split("```")[0]

        # Clean up the content
        content = content.strip()

        # Parse JSON
```

```
        extracted_info = json.loads(content)
        return extracted_info

    except Exception as e:
        print(f"Error in extraction: {e}")
        print("Full response content:")
        print(response.choices[0].message.content if 'response' in locals()␣
 ↪else "No response received")
        return None

def match_sdoh_codes(extracted_info, sdoh_codes):
    """Match extracted SDOH factors with their corresponding codes"""
    if not extracted_info or 'sdoh_factors' not in extracted_info:
        print("No SDOH factors found in extracted info")
        return extracted_info

    # Convert extracted SDOH factors to lowercase for matching
    sdoh_factors = [factor.lower() for factor in extracted_info['sdoh_factors']]

    # Match codes and create new dictionary with codes
    sdoh_with_codes = []
    for factor in sdoh_factors:
        matched_code = 'CODE_NOT_FOUND'
        # Try to find the best matching code
        for known_factor, code in sdoh_codes.items():
            if factor in known_factor or known_factor in factor:
                matched_code = code
                break

        sdoh_with_codes.append({
            'factor': factor,
            'code': matched_code
        })

    # Update the extracted info with coded SDOH factors
    result = extracted_info.copy()
    result['sdoh_factors'] = sdoh_with_codes

    return result

def process_clinical_notes(clinical_note, sdoh_codes):
    """Process clinical notes and return structured JSON output"""
    # Extract information
    print("Extracting information from clinical note...")
    extracted_info = extract_patient_info(clinical_note)

    if extracted_info:
```

```
        print("Successfully extracted information. Matching SDOH codes...")
        # Match SDOH codes
        final_output = match_sdoh_codes(extracted_info, sdoh_codes)

        # Save to JSON file
        output_file = 'extracted_healthcare_info.json'
        with open(output_file, 'w') as f:
            json.dump(final_output, f, indent=2)

        print(f"Results saved to {output_file}")
        return final_output
    return None

# Main execution
if __name__ == "__main__":
    try:
        print("Loading SDOH codes...")
        # Load SDOH codes from the CSV
        sdoh_codes = load_sdoh_codes("sdoh_factors2.csv")
        print(f"Loaded {len(sdoh_codes)} SDOH codes")

        print("Reading clinical note...")
        # Read the clinical note from the provided text
        with open("clinical_note_Prateek.txt", 'r') as f:
            clinical_note = f.read()

        print("Processing clinical notes...")
        # Process notes and get results
        results = process_clinical_notes(clinical_note, sdoh_codes)

        if results:
            print("\nSuccessfully extracted and coded healthcare information:")
            print(json.dumps(results, indent=2))
        else:
            print("Error: No results generated")

    except FileNotFoundError as e:
        print(f"Error: Could not find one of the required files - {e}")
    except Exception as e:
        print(f"Error: An unexpected error occurred - {e}")
```

```
Loading SDOH codes…
Loaded 10 SDOH codes
Reading clinical note…
Processing clinical notes…
Extracting information from clinical note…
Raw LLM Response:
{
```

```
  "patient_name": "Michael A. Davidson",
  "address": {
    "street": "1567 Park west Rd",
    "unit": "12C",
    "city": "Seabrook",
    "state": "NH",
    "zip": "03874",
    "phone": "(555) 897-6543"
  },
  "hospital": {
    "name": "Seabrook Memorial Hospital",
    "address": {
      "street": "789 Ocean Ave",
      "city": "Seabrook",
      "state": "NH",
      "zip": "03874"
    },
    "phone": "(555) 444-9999"
  },
  "allergies": [
    "Statins",
    "Contrast dye",
    "Aspirin"
  ],
  "medical_problems": [
    "NSTEMI",
    "severe hyperlipidemia"
  ],
  "sdoh_factors": [
    "radiation exposure",
    "workplace stress",
    "environmental factors (ionizing radiation)",
    "housing conditions (staff housing 2mi from plant)",
    "nutrition (processed cafeteria food consumption)",
    "healthcare access (limited social support besides wife)"
  ]
}
Successfully extracted information. Matching SDOH codes…
Results saved to extracted_healthcare_info.json

Successfully extracted and coded healthcare information:
{
  "patient_name": "Michael A. Davidson",
  "address": {
    "street": "1567 Park west Rd",
    "unit": "12C",
    "city": "Seabrook",
    "state": "NH",
```

```
    "zip": "03874",
    "phone": "(555) 897-6543"
  },
  "hospital": {
    "name": "Seabrook Memorial Hospital",
    "address": {
      "street": "789 Ocean Ave",
      "city": "Seabrook",
      "state": "NH",
      "zip": "03874"
    },
    "phone": "(555) 444-9999"
  },
  "allergies": [
    "Statins",
    "Contrast dye",
    "Aspirin"
  ],
  "medical_problems": [
    "NSTEMI",
    "severe hyperlipidemia"
  ],
  "sdoh_factors": [
    {
      "factor": "radiation exposure",
      "code": "CODE_NOT_FOUND"
    },
    {
      "factor": "workplace stress",
      "code": "CODE_NOT_FOUND"
    },
    {
      "factor": "environmental factors (ionizing radiation)",
      "code": "CODE_NOT_FOUND"
    },
    {
      "factor": "housing conditions (staff housing 2mi from plant)",
      "code": "CODE_NOT_FOUND"
    },
    {
      "factor": "nutrition (processed cafeteria food consumption)",
      "code": "CODE_NOT_FOUND"
    },
    {
      "factor": "healthcare access (limited social support besides wife)",
      "code": "CODE_NOT_FOUND"
    }
  ]
```

}

[3]:
```
!pip install gradio
```

```
Requirement already satisfied: gradio in c:\users\prateek
kumar\anaconda3\lib\site-packages (5.6.0)
Requirement already satisfied: aiofiles<24.0,>=22.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (23.2.1)
Requirement already satisfied: anyio<5.0,>=3.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (4.4.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.115.5)
Requirement already satisfied: ffmpy in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.4.0)
Requirement already satisfied: gradio-client==1.4.3 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (1.4.3)
Requirement already satisfied: httpx>=0.24.1 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (0.27.0)
Requirement already satisfied: huggingface-hub>=0.25.1 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.26.2)
Requirement already satisfied: jinja2<4.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (3.1.4)
Requirement already satisfied: markupsafe~=2.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (2.1.5)
Requirement already satisfied: numpy<3.0,>=1.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (1.26.4)
Requirement already satisfied: orjson~=3.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (3.10.11)
Requirement already satisfied: packaging in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (24.1)
Requirement already satisfied: pandas<3.0,>=1.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (10.3.0)
Requirement already satisfied: pydantic>=2.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (2.5.3)
Requirement already satisfied: pydub in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart==0.0.12 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.0.12)
Requirement already satisfied: pyyaml<7.0,>=5.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (6.0.2)
Requirement already satisfied: ruff>=0.2.2 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.7.4)
Requirement already satisfied: safehttpx<1.0,>=0.1.1 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.1.1)
Requirement already satisfied: semantic-version~=2.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (2.10.0)
```

```
Requirement already satisfied: starlette<1.0,>=0.40.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.41.3)
Requirement already satisfied: tomlkit==0.12.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.12.0)
Requirement already satisfied: typer<1.0,>=0.12 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.13.1)
Requirement already satisfied: typing-extensions~=4.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio) (4.12.2)
Requirement already satisfied: uvicorn>=0.14.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio) (0.32.1)
Requirement already satisfied: fsspec in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from gradio-
client==1.4.3->gradio) (2024.6.0)
Requirement already satisfied: websockets<13.0,>=10.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from gradio-client==1.4.3->gradio) (12.0)
Requirement already satisfied: idna>=2.8 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
anyio<5.0,>=3.0->gradio) (3.7)
Requirement already satisfied: sniffio>=1.1 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
httpx>=0.24.1->gradio) (2024.6.2)
Requirement already satisfied: httpcore==1.* in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
httpx>=0.24.1->gradio) (1.0.5)
Requirement already satisfied: h11<0.15,>=0.13 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
httpcore==1.*->httpx>=0.24.1->gradio) (0.14.0)
Requirement already satisfied: filelock in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from huggingface-
hub>=0.25.1->gradio) (3.15.4)
Requirement already satisfied: requests in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from huggingface-
hub>=0.25.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from huggingface-
hub>=0.25.1->gradio) (4.66.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
pandas<3.0,>=1.0->gradio) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
pandas<3.0,>=1.0->gradio) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
pandas<3.0,>=1.0->gradio) (2024.1)
```

```
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from pydantic>=2.0->gradio) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.6 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from pydantic>=2.0->gradio) (2.14.6)
Requirement already satisfied: click>=8.0.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
typer<1.0,>=0.12->gradio) (8.1.7)
Requirement already satisfied: shellingham>=1.3.0 in c:\users\prateek
kumar\anaconda3\lib\site-packages (from typer<1.0,>=0.12->gradio) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
typer<1.0,>=0.12->gradio) (13.7.1)
Requirement already satisfied: colorama in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
click>=8.0.0->typer<1.0,>=0.12->gradio) (0.4.6)
Requirement already satisfied: six>=1.5 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from python-
dateutil>=2.8.2->pandas<3.0,>=1.0->gradio) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.18.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
requests->huggingface-hub>=0.25.1->gradio) (3.3.2)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from
requests->huggingface-hub>=0.25.1->gradio) (2.2.2)
Requirement already satisfied: mdurl~=0.1 in c:\users\prateek
kumar\appdata\roaming\python\python312\site-packages (from markdown-it-
py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio) (0.1.2)


[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

[5]:
```
import os
import json
import pandas as pd
import gradio as gr
from dotenv import load_dotenv
from groq import Groq

# Load environment variables
load_dotenv()
```

```
GROQ_API_KEY = os.
 ↪getenv('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')

# Initialize Groq client
client = Groq(
    api_key="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",
)

def load_sdoh_codes(csv_path):
    """Load SDOH codes from CSV file"""
    try:
        sdoh_df = pd.read_csv(csv_path)
        # Create a dictionary mapping SDOH factors to their codes
        return dict(zip(sdoh_df['SDOH factor'].str.lower(), sdoh_df['Code']))
    except Exception as e:
        print(f"Error loading SDOH codes: {e}")
        return {}

def extract_patient_info(clinical_note):
    """Extract patient information using Groq LLM"""

    prompt = (
        "Extract and organize the following information from the clinical note␣
 ↪into JSON format:\n"
        "- Patient's full name\n"
        "- Complete address\n"
        "- Hospital name\n"
        "- List of allergies\n"
        "- List of major medical problems\n"
        "- List of social determinants of health (SDOH) factors (specifically␣
 ↪looking for: "
        "radiation exposure, workplace stress, environmental factors, housing␣
 ↪conditions, "
        "nutrition, healthcare access)\n\n"
        f"Clinical Note:\n{clinical_note}\n\n"
        "Format the response as a JSON object with these exact keys:\n"
        "{\n"
        '  "patient_name": "",\n'
        '  "address": "",\n'
        '  "hospital": "",\n'
        '  "allergies": [],\n'
        '  "medical_problems": [],\n'
        '  "sdoh_factors": []\n'
        "}\n"
        "Ensure all lists are properly formatted and all SDOH factors are in␣
 ↪lowercase."
    )
```

```
    try:
        response = client.chat.completions.create(
            messages=[{
                "role": "user",
                "content": prompt
            }],
            model="mixtral-8x7b-32768",
            temperature=0.1,
            max_tokens=1000
        )

        # Extract and parse JSON from response
        content = response.choices[0].message.content
        # Remove any potential markdown formatting
        if "```json" in content:
            content = content.split("```json")[1].split("```")[0]
        elif "```" in content:
            content = content.split("```")[1].split("```")[0]

        # Clean up the content
        content = content.strip()

        # Parse JSON
        extracted_info = json.loads(content)
        return extracted_info

    except Exception as e:
        return f"Error in extraction: {e}"

def match_sdoh_codes(extracted_info, sdoh_codes):
    """Match extracted SDOH factors with their corresponding codes"""
    if not extracted_info or 'sdoh_factors' not in extracted_info:
        return "No SDOH factors found in extracted info"

    # Convert extracted SDOH factors to lowercase for matching
    sdoh_factors = [factor.lower() for factor in extracted_info['sdoh_factors']]

    # Match codes and create new dictionary with codes
    sdoh_with_codes = []
    for factor in sdoh_factors:
        matched_code = 'CODE_NOT_FOUND'
        # Try to find the best matching code
        for known_factor, code in sdoh_codes.items():
            if factor in known_factor or known_factor in factor:
                matched_code = code
                break
```

```
            sdoh_with_codes.append({
                'factor': factor,
                'code': matched_code
            })

    # Update the extracted info with coded SDOH factors
    result = extracted_info.copy()
    result['sdoh_factors'] = sdoh_with_codes

    return result

def process_clinical_notes_gradio(clinical_note, sdoh_csv):
    """Gradio wrapper function for processing clinical notes"""
    try:
        # Load SDOH codes from the uploaded CSV
        if sdoh_csv is None:
            return "Please upload an SDOH codes CSV file"

        # Read the CSV file directly using pandas
        sdoh_codes = load_sdoh_codes(sdoh_csv.name)  # Use the file path␣
 ↪directly

        if not sdoh_codes:
            return "Error loading SDOH codes from CSV"

        # Extract information
        extracted_info = extract_patient_info(clinical_note)

        if isinstance(extracted_info, str):  # Error message
            return extracted_info

        # Match SDOH codes
        final_output = match_sdoh_codes(extracted_info, sdoh_codes)

        # Convert to formatted string for display
        if isinstance(final_output, str):  # Error message
            return final_output

        return json.dumps(final_output, indent=2)

    except Exception as e:
        return f"Error processing clinical notes: {e}"

# Create Gradio interface
def create_gradio_interface():
    with gr.Blocks(title="Healthcare Information Extraction") as demo:
```

```
        gr.Markdown("# Healthcare Information Extraction System")
        gr.Markdown("Upload a clinical note and SDOH codes CSV to extract and␣
 ↪code patient information.")

        with gr.Row():
            with gr.Column():
                clinical_note_input = gr.Textbox(
                    label="Clinical Note",
                    placeholder="Paste clinical note here...",
                    lines=10
                )
                sdoh_csv_input = gr.File(
                    label="Upload SDOH Codes CSV",
                    file_types=[".csv"]
                )
                process_button = gr.Button("Process Clinical Note")

            with gr.Column():
                output_display = gr.TextArea(
                    label="Extracted Information",
                    lines=15,
                    interactive=False
                )

        process_button.click(
            fn=process_clinical_notes_gradio,
            inputs=[clinical_note_input, sdoh_csv_input],
            outputs=output_display
        )

    return demo

# Main execution
if __name__ == "__main__":
    demo = create_gradio_interface()
    demo.launch(share=True)
```

```
* Running on local URL:  http://127.0.0.1:7864

Could not create share link. Please check your internet connection or our status
page: https://status.gradio.app.
```

```
<IPython.core.display.HTML object>
```