Untitled

November 21, 2024

[66]:
```
import pandas as pd
import re
import json

# Load the clinical note
with open("clinical_note_Prateek.txt", "r") as file:
    clinical_note = file.read()

# Load the SDOH factors and their codes from the CSV file
sdoh_csv = "sdoh_factors2.csv"
sdoh_df = pd.read_csv(sdoh_csv)

# Ensure column names are trimmed
sdoh_df.columns = sdoh_df.columns.str.strip()

# Print the column names for debugging
print("Columns in SDOH CSV:", sdoh_df.columns)

# Dynamically find the correct column names
factor_column = None
code_column = None

for col in sdoh_df.columns:
    if "factor" in col.lower():
        factor_column = col
    if "code" in col.lower():
        code_column = col

if not factor_column or not code_column:
    raise ValueError("Unable to find the required columns for SDOH factors or␣
 ↪codes in the CSV file.")

# Extract patient details using regular expressions
def extract_details(note):
    details = {}
    details['patient_name'] = re.search(r"Pt:\s*(.+?)\s*\(DOB", note).group(1)
    details['address'] = re.search(r"residing\s@\s(.+?),\sph#", note).group(1)
```

```
    details['hospital_name'] = re.search(r"Treating facility:\s*(.+?),\sTel",␣
 ↪note).group(1)
    details['allergies'] = re.findall(r"allergies to:\s*(.+?)(?:,|$)", note)
    details['major_medical_problems'] = re.findall(r"Dx:\s*(.+?)(?:
 ↪\s*&|\s*,|$)", note)
    return details

# Extract SDOH factors from the clinical note
def extract_sdoh(note):
    sdoh_factors = sdoh_df[factor_column].tolist()
    extracted = {}
    for factor in sdoh_factors:
        if factor.lower() in note.lower():
            extracted[factor] = factor
    return extracted

# Match SDOH factors with their codes
def match_sdoh_codes(extracted_sdoh):
    matched_sdoh = {}
    for factor, description in extracted_sdoh.items():
        row = sdoh_df[sdoh_df[factor_column].str.lower() == factor.lower()]
        if not row.empty:
            matched_sdoh[factor] = {
                "description": description,
                "code": row.iloc[0][code_column]
            }
    return matched_sdoh

# Extract details and SDOH factors
extracted_details = extract_details(clinical_note)
extracted_sdoh = extract_sdoh(clinical_note)
matched_sdoh = match_sdoh_codes(extracted_sdoh)

# Combine all extracted data into JSON format
output_data = {
    "patient_details": extracted_details,
    "sdoh_factors": matched_sdoh
}

# Output the data as JSON
json_output = json.dumps(output_data, indent=4)
print(json_output)

# Optionally save to a JSON file
with open("extracted_clinical_data.json", "w") as json_file:
    json_file.write(json_output)
```

```
Columns in SDOH CSV: Index(['SDOH factor', 'Code'], dtype='object')
{
    "patient_details": {
        "patient_name": "Michael A. Davidson",
        "address": "1567 Park west Rd, Unit 12C, Seabrook, NH 03874",
        "hospital_name": "Seabrook Memorial Hospital, 789 Ocean Ave, Seabrook,
NH 03874",
        "allergies": [
            "Statins (myalgia)"
        ],
        "major_medical_problems": [
            "NSTEMI"
        ]
    },
    "sdoh_factors": {}
}
```

[76]:
```
import pandas as pd
import re
import json
import gradio as gr

def process_files(clinical_note_file, sdoh_csv_file):
    try:
        # Handle clinical note file input
        if hasattr(clinical_note_file, 'name'):
            # If it's a file with a name attribute (typical file upload)
            with open(clinical_note_file.name, 'r', encoding='utf-8') as f:
                clinical_note = f.read()
        elif isinstance(clinical_note_file, str):
            # If it's already a string
            clinical_note = clinical_note_file
        else:
            # Try to read content directly
            clinical_note = clinical_note_file.decode('utf-8') if␣
 ↪isinstance(clinical_note_file, bytes) else str(clinical_note_file)

        # Handle SDOH CSV file input
        if hasattr(sdoh_csv_file, 'name'):
            # If it's a file with a name attribute
            sdoh_df = pd.read_csv(sdoh_csv_file.name)
        elif isinstance(sdoh_csv_file, pd.DataFrame):
            # If it's already a DataFrame
            sdoh_df = sdoh_csv_file
        else:
            # Try to read from bytes or convert to DataFrame
```

```
            sdoh_df = pd.read_csv(pd.compat.BytesIO(sdoh_csv_file)) if␣
↪isinstance(sdoh_csv_file, bytes) else pd.read_csv(sdoh_csv_file)

        # Ensure column names are trimmed
        sdoh_df.columns = sdoh_df.columns.str.strip()

        # Dynamically find the correct column names for factors and codes
        factor_column = None
        code_column = None
        for col in sdoh_df.columns:
            if "factor" in col.lower():
                factor_column = col
            if "code" in col.lower():
                code_column = col

        if not factor_column or not code_column:
            return json.dumps({"error": "Unable to find the required columns␣
↪for SDOH factors or codes in the CSV file."}, indent=4)

        # Extract patient details using regular expressions
        def extract_details(note):
            details = {}
            try:
                details['patient_name'] = re.search(r"Pt:\s*(.+?)\s*\(DOB",␣
↪note).group(1)
                details['address'] = re.search(r"residing\s@\s(.+?),\sph#",␣
↪note).group(1)
                details['hospital_name'] = re.search(r"Treating facility:\s*(.+?
↪),\sTel", note).group(1)
                details['allergies'] = re.findall(r"allergies to:\s*(.+?)(?:
↪,|$)", note)
                details['major_medical_problems'] = re.findall(r"Dx:\s*(.+?)(?:
↪\s*&|\s*,|$)", note)
            except Exception as e:
                details['extraction_error'] = str(e)
            return details

        # Extract SDOH factors from the clinical note
        def extract_sdoh(note):
            sdoh_factors = sdoh_df[factor_column].tolist()
            extracted = {}
            for factor in sdoh_factors:
                if factor.lower() in note.lower():
                    extracted[factor] = factor
            return extracted
```

```
        # Match SDOH factors with their codes
        def match_sdoh_codes(extracted_sdoh):
            matched_sdoh = {}
            for factor, description in extracted_sdoh.items():
                row = sdoh_df[sdoh_df[factor_column].str.lower() == factor.
 ↪lower()]
                if not row.empty:
                    matched_sdoh[factor] = {
                        "description": description,
                        "code": row.iloc[0][code_column]
                    }
            return matched_sdoh

        # Extract details and SDOH factors
        extracted_details = extract_details(clinical_note)
        extracted_sdoh = extract_sdoh(clinical_note)
        matched_sdoh = match_sdoh_codes(extracted_sdoh)

        # Combine all extracted data into JSON format
        output_data = {
            "patient_details": extracted_details,
            "sdoh_factors": matched_sdoh
        }

        return json.dumps(output_data, indent=4)

    except Exception as e:
        return json.dumps({"error": f"An error occurred: {str(e)}"}, indent=4)

# Define the Gradio interface
def create_gradio_interface():
    interface = gr.Interface(
        fn=process_files,
        inputs=[
            gr.File(label="Upload Clinical Note (TXT)", file_types=[".txt"]),
            gr.File(label="Upload SDOH Factors CSV", file_types=[".csv"])
        ],
        outputs=gr.JSON(label="Extracted Clinical Data"),
        title="Clinical Data Extraction Tool",
        description="Upload a clinical note (TXT) and a CSV file containing␣
 ↪SDOH factors and codes. The tool will extract patient details and match SDOH␣
 ↪factors with their codes."
    )
    return interface

# Launch the Gradio app
if __name__ == "__main__":
```

```
    interface = create_gradio_interface()
    interface.launch()
```

```
* Running on local URL:  http://127.0.0.1:7874

To create a public link, set `share=True` in `launch()`.
```

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

[ ]:

[ ]: