7. Discharge and RadiologynoteFeatures

December 5, 2025

1 Clinical Notes (NLP) Feature Script

1.1 Description

This script applies Natural Language Processing (NLP) techniques (keyword extraction and regex matching) to unstructured Discharge Summaries and Radiology Reports.

1.2 Clinical Justification for HAPI Research

Structured data often misses subtle risk factors that are only captured in narrative text. We target four specific domains based on the Braden Scale risk model: * **Immobility:** Terms like "bedbound" map directly to Braden Mobility deficits, indicating an inability to offload pressure. * **Moisture:** "Incontinence" identifies Moisture-Associated Skin Damage (MASD) risk, which weakens the skin barrier (stratum corneum). * **Nutrition:** "Cachectic" or "poor appetite" flags hypoalbuminemia and nutritional deficits that impair tissue repair. * **Perfusion:** Mentions of "mottled skin" or "cool extremities" are clinical markers of tissue ischemia and poor microcirculation.

1.3 Inputs & Outputs

- **Inputs:** `discharge.csv`, `radiology.csv`
- **Output:** `feat_notes.csv`
- **Key Features:**
  - `immobility` (Binary Flag)
  - `incontinence` (Binary Flag)
  - `nutrition_bad` (Binary Flag)
  - `perfusion_poor` (Binary Flag)

[1]:
```
import pandas as pd
import os
import re
```

[2]:
```
#Configuration
BASE_DIR = r"D:\School\5141"

# Paths to discharge summaries and radiology reports
DISCHARGE_PATH = os.path.join(BASE_DIR, "discharge.csv", "discharge.csv")
RADIOLOGY_PATH = os.path.join(BASE_DIR, "radiology.csv", "radiology.csv")

# Output file for engineered note-based features
```

```
OUTPUT_PATH = os.path.join(BASE_DIR, "feat_notes.csv")
```

[ ]:
```
# Risk factor KeyWords
# These keywords represent validated HAPI risk factors mentioned in clinical␣
 ↪notes.
# Utilizing regex patterns to capture variations

# Immobility
# Corresponds to Braden Scale Activity and Mobility.
# Patients who cannot reposition themselves independently are at maximum risk.
IMMOBILITY_KW = [
    "bedbound", "bed bound", "bedfast", "chairfast",
    "requires assist", "total care", "max assist",
    "mechanical lift", "hoyier", "unable to turn"
]

# Moisture/ Incontience
# Corresponds to Braden Scale Moisture.
# Ammonia in urine increases skin pH, disrupting the acid mantle and increasing␣
 ↪permeability to bacteria and physical stress (MASD).
INCONT_KW = [
    "incontinent", "fecal incontinence", "urinary incontinence",
    "diaper", "briefs soiled", "foley leaking", "loose stool"
]

# NUTRITION
# Corresponds to Braden Scale Nutrition.
# Malnutrition reduces subcutaneous fat (padding) and protein stores needed for␣
 ↪ischemic repair.
NUTRITION_KW = [
    "poor appetite", "malnourished", "cachectic", "wasted",
    "weight loss", "tube feeding", "parenteral nutrition",
    "tpen", "npo"
]

# PERFUSION
# Poor perfusion reduces oxygen delivery to  the skin, significantly lowering␣
 ↪the pressure threshold required to cause necrosis.
PERFUSION_KW = [
    "mottled", "cool extremities", "poor perfusion",
    "hypotensive", "vasopressor", "shock", "cyanotic",
    "weak pulses", "delayed capillary refill"
]

# NOTE: We do NOT include "stage 2" or "ulcer" keywords here because those are␣
 ↪Ooutcomes (Labels), not predictors. We want to predict risk before the␣
 ↪injury happens.
```

[ ]:
```
# Helper Function

# Convert keyword list to a regex pattern
def make_regex(kw_list):
    """Create a regex OR pattern from a list of keywords."""
    return "|".join([re.escape(k) for k in kw_list])


# Regex patterns
IMMOBILITY_RE = make_regex(IMMOBILITY_KW)
INCONT_RE     = make_regex(INCONT_KW)
NUTRITION_RE  = make_regex(NUTRITION_KW)
PERFUSION_RE  = make_regex(PERFUSION_KW)
```

[ ]:
```
# Loading Function
def load_and_combine_notes():
    """
    Load discharge summaries and radiology reports,
    combine them into one dataset,
    lowercase the text for keyword searching.
    """

    cols = ["hadm_id", "text"]

    d_df = pd.read_csv(DISCHARGE_PATH, usecols=cols, low_memory=False)
    r_df = pd.read_csv(RADIOLOGY_PATH, usecols=cols, low_memory=False)

    # Stack both note sources into one table
    notes = pd.concat([d_df, r_df], ignore_index=True)

    # Clean up missing values
    notes = notes.dropna(subset=["hadm_id", "text"])
    notes["hadm_id"] = notes["hadm_id"].astype("Int64")

    # Lowercase text for searching
    notes["text_lower"] = notes["text"].astype(str).str.lower()

    print(f"Total notes to process: {len(notes)}")
    return notes
```

[ ]:
```
# Feature Extraction Function
def process_features(notes):
    """
    For each note:
    - Search for risk keywords
    - Create binary flags (1 = keyword found, 0 = not found)
```

```
    Then group by hadm_id to see whether the patient had that risk factor.
    """

    # Keyword-based binary flags
    notes["immobility"]      = notes["text_lower"].str.contains(IMMOBILITY_RE,␣
↪regex=True).astype(int)
    notes["incontinence"]    = notes["text_lower"].str.contains(INCONT_RE,␣
↪regex=True).astype(int)
    notes["nutrition_bad"]   = notes["text_lower"].str.contains(NUTRITION_RE,␣
↪regex=True).astype(int)
    notes["perfusion_poor"]  = notes["text_lower"].str.contains(PERFUSION_RE,␣
↪regex=True).astype(int)

    # Aggregate to patient-level features
    # If a patient has 10 notes and ANY of them mention a risk term: they get a␣
↪1
    feat = notes.groupby("hadm_id").agg({
        "immobility": "max",
        "incontinence": "max",
        "nutrition_bad": "max",
        "perfusion_poor": "max"
    }).reset_index()

    return feat
```

[7]:
```
def save_data(df):
    df.to_csv(OUTPUT_PATH, index=False)
    print(f"Saved {len(df)} rows to {OUTPUT_PATH}")

if __name__ == "__main__":
    notes_df = load_and_combine_notes()
    feat_df = process_features(notes_df)
    save_data(feat_df)
```

```
Total notes to process: 1476551
Saved 374285 rows to D:\School\5141\feat_notes.csv
```