4.I&Ofeatures

December 5, 2025

1 Input/Output (I&O) Feature Script

1.1 Description

This script processes massive MIMIC-IV event tables (`inputevents`, `outputevents`) to calculate total fluid volume flux per hospital admission.

1.2 Clinical Justification for HAPI Research

Fluid balance is a critical determinant of skin integrity: * **Moisture :** High fluid output can lead to moisture-associated skin damage, weakening the skin barrier. * **Perfusion & Edema:** Positive fluid balance (fluid overload) causes tissue edema, increasing the diffusion distance for oxygen and nutrients to the skin. * **Dehydration:** Negative fluid balance reduces skin turgor and blood volume, impairing tissue perfusion.

1.3 Inputs & Outputs

- **Inputs:** `inputevents.csv`, `outputevents.csv`, `hospitalwide_hapi_labels.csv` (for filtering)
- **Output:** `io_feat.csv`
- **Key Features:**
  - `total_input_ml`: Total volume of fluids/meds administered.
  - `total_output_ml`: Total volume of urine/drainage.
  - `net_io_ml`: Net fluid balance (Input - Output).

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

[2]:
```
# Congiguration

# Base directory for MIMIC-IV data files
BASE_DIR = r"D:\School\5141"

# Path to HAPI labels (useqd to filter to relevant encounters only for␣
 ↪computational efficiency)
LABEL_PATH = os.path.join(BASE_DIR, "hospitalwide_hapi_labels.csv")

#MIMIC-IV IO event files
INPUTEVENTS_PATH  = os.path.join(BASE_DIR, "inputevents.csv", "inputevents.csv")
```

```
OUTPUTEVENTS_PATH = os.path.join(BASE_DIR, "outputevents.csv", "outputevents.
 ↪csv")

# Output file for  IO features
OUTPUT_FEAT_PATH = os.path.join(BASE_DIR, "io_feat.csv")

# Number of rows per chunk
CHUNK_SIZE = 500_000
```

[3]:
```
def get_hadm_list():
    """
    Load the list of HADM_IDs from final HAPI label table.
    """
    df = pd.read_csv(LABEL_PATH, usecols=["hadm_id"], low_memory=False)
    return set(df["hadm_id"].astype("Int64").dropna().unique())

def aggregate_stream(path, hadm_list, value_col, uom_col, out_col_name):
    """
    Process MIMIC event table in chunks.

    Parameters:
    path : str
        CSV path to either inputevents or outputevents.
    hadm_list : set
        Set of HADM_IDs to filter down to.
    value_col : str
        Column containing volume.
    uom_col : str
        Unit of measure column.
    out_col_name : str
        feature being created.

    Returns
    pd.DataFrame
        A table with:
          hadm_id
          summed volume in mL
    """
    print(f"Processing {out_col_name} from {path}...")
    chunks = []

    # Load in chunks
    reader = pd.read_csv(
        path,
        usecols=["hadm_id", value_col, uom_col],
        chunksize=CHUNK_SIZE,
        low_memory=False
```

```
    )

    # Loop over each chunk
    for i, chunk in enumerate(reader):

        # Keep only rows with both hadm_id and the volume column
        chunk = chunk.dropna(subset=["hadm_id", value_col])

        # Consistent type for merging
        chunk["hadm_id"] = chunk["hadm_id"].astype("Int64")

        # Filter to admissions that appear in the HAPI label file
        chunk = chunk[chunk["hadm_id"].isin(hadm_list)]

        # Keep only measurements in mL
        if uom_col in chunk.columns:
            chunk = chunk[chunk[uom_col].str.lower().str.contains("ml",␣
↪na=False)]

        # If any rows remain, group and sum their volumes
        if not chunk.empty:
            grp = (
                chunk.groupby("hadm_id")[value_col]
                .sum()
                .rename(out_col_name)
            )
            chunks.append(grp)

    # If nothing matched, return empty dataframe
    if not chunks:
        return pd.DataFrame(columns=["hadm_id", out_col_name])

    # Combine partial sums from all chunks and re-sum
    final = (
        pd.concat(chunks)
        .groupby("hadm_id")
        .sum()
        .reset_index()
    )
    return final
```

[4]:
```
def main():
    # Only look at admissions that appear in the HAPI label table
    target_hadms = get_hadm_list()

    # Build Input Features
    inputs = aggregate_stream(
```

```
        INPUTEVENTS_PATH,
        target_hadms,
        "amount",
        "amountuom",
        "total_input_ml"
    )

    # Build Output Features
    outputs = aggregate_stream(
        OUTPUTEVENTS_PATH,
        target_hadms,
        "value",
        "valueuom",
        "total_output_ml"
    )

    # Merge both based on HADM_ID
    feat = inputs.merge(outputs, on="hadm_id", how="outer")

    # Fill in missing data with 0 (missing data in IO context usually means␣
↪ZERO volume)
    feat["total_input_ml"] = feat["total_input_ml"].fillna(0)
    feat["total_output_ml"] = feat["total_output_ml"].fillna(0)

    # Compute net fluid balance
    feat["net_io_ml"] = feat["total_input_ml"] - feat["total_output_ml"]

    # Save final feature file
    feat.to_csv(OUTPUT_FEAT_PATH, index=False)
```

[5]:
```
# Execute
if __name__ == "__main__":
    main()
```

```
Processing total_input_ml from D:\School\5141\inputevents.csv\inputevents.csv…
Processing total_output_ml from
D:\School\5141\outputevents.csv\outputevents.csv…
```