MIMIC IV Data Preparation

November 2022

Introduction

MIMIC IV is a relational database containing real hospital stays for patients admitted to a tertiary academic medical center in Boston, MA, USA between 2008-2019. MIMIC IV contains comprehensive information for each patient while they were in the hospital: laboratory measurements, medications administered, vital signs documented, and so on. The database is intended to support a wide variety of research in healthcare.

However, MIMIC IV provides data in a heterogeneous way. If we want to make use of it, the first thing is to clean and reorganize all the data you need. In this article, we elaborate on how to clean this database and the output of the cleaning program.

The official document of MIMIC IV: https://mimic.mit.edu/docs/iv/.

NOTE: This article and the corresponding program are based on MIMIC IV (v1.0). Since MIMIC is subject to change, please check the version before you download the data or consult the official documentation.

1 Summary of the Final Dataset

In this section, we will discuss the final dataset after the cleaning. The final data contains the EHR of 382,278 patients, out of them 53,150 were in ICUs.

These data comes from **core** (transfers), **hospital tables** (diagnoses_icd, drgcodes, hcpcsevents, labevents, prescriptions, procedures_icd), and **ICU tables** (chartevents, inputevents, outputevents, procedureevents).

Some codes from these tables are aggregated into high-level codes. We convert procedure codes, including HCPC, ICD-9, and ICD-10-PCS, to CCS codes. Diagnosis codes, including ICD-9 and ICD-10-CM, are grouped into PheCodes. NDC codes are aggregated to RXNORM CUIs. The statistics of these aggregated codes are shown in Table 1.

Hint: For medications, some NDC codes are mapped to more than one RxNorm codes. Here we put these codes together and join them with "_" (same for the labels of them). For example, the NDC code "00121176130" is mapped to "612_6581_9796" (aluminum hydroxide, magnesium hydroxide, and simethicone).

Table 1: Statistics of aggregated codes

Original Code	Source Table	Number of original codes	Target code
ICD9/ICD10PCS	procedures_icd	13,016	CCS
HCPCS	hcpcsevents	2,239	CCS
ICD9/ICD10CM	diagnoses_icd	27,192	PheCode
NDC	prescriptions	5,831	RxNorm

After the aggregation, codes whose frequency of occurrence is less than 1000 are dropped. Some ICD codes from diagnoses_icd with high frequency cannot be grouped to PheCode by the mapping table we use, and we still keep them in the dataset. Statistics of code occurrences of the final dataset are shown in Table2.

Table 2: Statistics of final dataset

Code	Source Table	Num of codes	Total freq	Mean freq	Median freq	Max freq
Trans code	transfers	4	2,189,535	547,383	523,744	661,053
Chart code	chartevents	1,533	316,630,780	206,543	25,691	9,184,087
Input code	inputevents	184	9,435,569	51,280	9,258	1,326,336
Output code	outputevents	53	4,451,819	83,996	10,225	3,137,879
Proc code	procedureevents	55	704,033	12,800	6,380	90,805
Lab code	labevents	503	121,995,684	242,536	14,006	3,453,735
ICD10CM	diagnoses_icd	17	54,083	3,181	1,489	15,636
ICD9	diagnoses_icd	52	216,276	4,159	2,870	23,940
PheCode	diagnoses_icd	637	4,627,556	7,264	3,253	162,450
DRG	drgcodes	219	600,803	2,743	1,788	52,072
CCS	hcpcs/proc_icd*	129	900,115	6,977	3,479	138,016
RxNorm	prescriptions	551	14,612,462	26,519	5,680	894,474
Total	-	3,937	476,418,715	121,010	7,322	9184087

*CCS codes are derived from hcpcsevents and procedures_icd.

Some codes occur with numeric values in the dataset, and we call them **valued codes** or **codes with values** here. However, since some valued codes occur with more than one unit of measurement, we have to drop the value of those occurrences whose units cannot be normalized. As a result, for some codes, their frequency of occurrences with values could be less than the frequency of all occurrences. Statistics of code occurrences with numeric values are shown in Table3.

Table 3: Statistics of code with numeric values

Code	Source Table	Number of codes	Total freq	Mean freq	Median freq	Max freq
Chart code	chartevents	557	114,317,075	205,237	24,891	6,798,187
Output code	outputevents	53	4,451,819	83,996	10,225	3,137,879
Lab code	labevents	331	103,379,431	312,324	18,592	3,444,528
Total	-	941	222,148,325	236,076	18,730	6,798,187

2 Description of Output Files

The final cleaned datasets (5 files) generated after going through the data cleaning pipeline as described in section 3 can be accessed here: Download output files. You can use these files to check the results generated on your end. A description of each output file is provided below

tuples.csv, **string_tuples.csv**, **tuples_demo.csv**, **patients_dict.csv**, and **code_dict.csv**.

2.1 tuples.csv

The **tuples.csv** contains all tuples of patient organized according to the patients' ID defined in **patients_dict.csv**. Each tuple indicates an occurrence of code. The format of this file is:

$$[patientID, admissionID, time, code, value].$$

The *code* is in form of "type_code" such as "ccs_101" and "phecode_401.1". Each $admissionID$ indicates one admission of a patient to the hospital. The $admissionIDs$ of some tuples are empty because the patient was not hospitalized at that time.

Moreover, as we discussed in the previous section, only valid numeric values and transfer targets (only for transfer codes, indicating the wards that patients are transferred to) are presented in this file in the *value* column. For some entries in original MIMIC, the values are presented in form of strings (text) or the unit of measurement cannot be converted to the wanted unit. Therefore, we use special labels in tuples.csv for the *value* column:

- _EMPYT: The code is without values and the original value of the entry (tuple) is empty.
- _MISSING: The code is with values and the original value of the entry (tuple) is empty.
- _STRING: The code is with values and but the unit of measurement of the entry (tuple) is invalid. Or, the code is without values and the value of the entry (tuple) is a string.

If the value of a tuple is _STRING, we present this tuple with the original string in another file **string_tuples.csv**.

2.2 string_tuples.csv

The format of **string_tuples.csv** is identical to **tuples.csv**. If the value of a tuple is _STRING in tuples.csv, we present this tuple with the original string in string_tuples.csv. For example, a tuple in tuples.csv is:

$$[10127185, 27920583, 2141-10-31\ 17:51:00, mimic\_223780, \_STRING].$$

Then there will be a corresponding tuple in string_tuples.csv:

$$[10127185, 27920583, 2141-10-31\ 17:51:00, mimic\_223780, Hemodynamic\ Instability].$$

Moreover, as we discussed in the previous section, we exclude occurrences of valued code with inconvertible units of measurement into tuples.csv. For these codes, we present them in string_tuples.csv in form of "value#unit". For example, a tuple in tuples.csv is:

$$[10018928, , 2134-01-17\ 22:39:00, mimic\_51249, \_STRING].$$

Then there will be a corresponding tuple in string_tuples.csv:

$$[10018928, , 2134-01-17\ 22:39:00, mimic\_51249, 31.9\#g/dl].$$

2.3 tuples_demo.csv

The **tuples_demo.csv** contains some data from tuples.csv, which could help you understand the format of the file better.

2.4 patients_dict.csv

The **patients_dict.csv** provides a patient's information. Only *subject_id*, *gender*, and *age* are provided, if the patient has never been hospitalized. Moreover, the check-in time of a few patients is later than the check-out time in MIMIC, please be cautious about it when dealing with time stamps.

- subject_id: INTEGER(UNIQUE)

  Patient's unique identifier.

- gender: STRING

  Genotypical sex of the patient.

- age: INTEGER

  Patient's age.

- ethnicity, marital_status, language: STRING

  Patient's ethnicity, marital_status, and language.

- in_time, out_time: TIMESTAMP

  in_time gives the date and time the patient was admitted to the hospital in the first hospitalization. *out_time* gives the date and time the patient was discharged from the hospital in the last hospitalization.

- death_time: STRING

  death_time is only present if the patient died in-hospital or it would be empty.

- los: FLOAT

  The total length of stay for the patient in ICUs, measured in days.

2.5 code_dict.csv

The **code_dict.csv** is the dictionary of all codes. The attributes are defined as:

- index: INTEGER

  Index of code.

- code: STRING

  The code itself.

- code_type: STRING

  The type of code, including rxnorm, icd10, icd9, phecode, drg, ccs, transfers, and mimic.

- value_frequency: INTEGER

  Number of tuples with numeric value.

- total_frequency: INTEGER

  Number of all tuples.

- source_table: STRING

  Source table of the code in MIMIC, including transfers, outputevents, labevents, prescriptions, chartevents, inputevents, procedureevents, diagnoses_icd, drgcodes, and ccs.

- unit_of_measurement: STRING

  Unit of measurement of the code.

- with_value: {0,1}

  The value=1 indicates the code has numeric value and the value=0 indicates not.

- label: STRING

  The label or main description of the code, provided by the source of code.

- description: STRING

  Additional description of the code, provided by the source of code.

- category: STRING

  For some codes from labevents, MIMIC provides the LOINC codes of them, and we present these LOINC codes here; for PheCodes, we provide the category information in this column; for Rxnorm codes, this column indicates whether the medication is ingredient-level.

3 Data Cleaning Process

As we know, MIMIC IV is an enormous dataset with scattered EHR tables. Our goal is to reduce errors in MIMIC and merge scattered tables into a uniform and ordered CSV table.

Generally speaking, there are four cleaning procedures for MIMIC IV: data inspection, dictionary generation, tuple generation, and post-processing.

The first step is to download the complete MIMIC IV V1.0 dataset. Once downloaded, you can move the files to your tutorial workspace. You will need to update this path in your settings.py script. You can then run clean_mimic.py to clean the dataset in one go or go through individual scripts in order to understand the pipeline. The description of scripts are provided below.

- **settings.py**: set the path of source files (MIMIC data) and output files.
- **clean_mimic.py**: run the whole cleaning program automatically.
- **generate_dictionary.py**: generate dictionaries for the dataset.
- **generate_tuples.py**: clean MIMIC data and generate cleaned data.
- **post_process.py**: update the code dictionary and generate the patient dictionary.
- **rolluptool.py**: an interface to load dictionaries for rolling up

In the following part of this section, we will discuss the procedures for cleaning MIMIC IV data in detail.

3.1 Data Inspection

At the very beginning, if you are not familiar with MIMIC IV, please read their document thoroughly and inspect the dataset by yourself before you try to clean it. In this process, we will have a general view of these data. Furthermore, we should make some preparations for the following cleaning procedures.

First, generating the mapping tables for rolling up is necessary. As we mentioned in Section 1, we need to roll up ICD, CPT, and NDC codes in MIMIC IV to high-level codes including CCS, PheCode, and RxNorm. To create a mapping for rolling up, we need to download some extra data like the mapping table from CPT to CCS and filter the mappings we need. Here, we provide the mapping tables for rolling up in the folder *rollup* directly. The program will load these tables by functions in *rolluptool.py*.

Second, we need to normalize units in MIMIC IV. In MIMIC IV, units of measurement for a single lab code could vary so we have to determine the main unit (most frequently-used unit for the code in MIMIC IV) for each code with value. For units other than the main unit, we either convert them to the main unit

or discard the value from tuples.csv if they are not convertible. For example, conversion between liter and milliliter is feasible, but conversion between liter and gram is not. In brief, the determination of the main units has to be done, and we also need to manually make the conversion rules between units. Here we provide the result of unit normalization in the folder *tools*.

3.2 Dictionary Generation

Through inspection, we know the general state of MIMIC IV and get what we need for rolling up and unit normalization. Now, we start the cleaning process by generating a dictionary for the dataset, namely *code_dict.csv*. It filters unwanted lab codes in MIMIC IV and provides the frequency, code type, source table, and unit of measurement of each code. Besides, *code_dict.csv* assigns a new numeric index for each lab code to avoid conflict between different types of code. This dictionary will guide the following cleaning work and help us verify the result of tuple generation.

The program in **generate_dictionary.py** first generates a **temporary dictionary** for each table respectively and then merges all these dictionaries together. The structure of the temporary dictionaries is the same as *code_dict.csv*.

3.2.1 Temporary Dictionary Generation

The generation process for each table are similar:

Step 1: Scan all the entries in the table, roll up the lab codes, and reserve those codes which cannot be rolled up.

Step 2: Count the frequency of all the lab codes, and codes with frequency less than 1,000 are discared.

Step 3: Record frequency of lab codes and output the temporary dictionary.

As we mentioned in section 1, only lab codes in procedures_icd, hcpcsevents, diagnoses_icd, and prescriptions need to be rolled up to high-level codes. Other tables do not have to execute Step 1.

For the tables not containing lab codes with value, Step 2 counts the frequency of each code directly to fill in the column $total\_frequency$, and the column $valid\_frequency$ of these codes will be set as 0. However, for tables containing lab codes with value, Step 2 also needs to count the frequency of occurrence with valid value and unit to fill in the column $valid\_frequency$.

3.2.2 Temporary Dictionary Merging

The merging process is to concatenate all the temporary dictionaries together, sort the lab codes by frequency, and assign a numeric index (like 1, 2, 3, ..., n) for each lab code. Now, we get a dictionary of all the codes we need, namely *code_dict.csv*.

There are some duplicate lab codes between *chartevents* and *labevents* according to the MIMIC IV official document. Thus, it is necessary to remove these codes in *chartevents* during the merging process.

3.3 Tuple Generation

With the dictionary, we are able to convert each entry in MIMIC IV into a tuple as:

$$[patientID, admissionID, time, code, value].$$

The approach of tuple generation is similar to dictionary generation: first, we generate tuples for each table and then merge all these tuples into a single file *tuples.csv*. The procedure for generating tuples for a single table is:

Step 1: Scan all the entries in the table, roll up the lab codes which can be rolled up, and discard codes not in *code_dict.csv*.

Step 2: Convert original lab codes to corresponding indexes defined in *code_dict.csv*, normalize the unit of measurement and numeric value, and get the tuple of each entry.

Step 3: Group all the tuples by patients' ID and output tuples for each table.

Step 4: Merge tuples of each table into a single file and sort tuples of each patient by time.

In Step 2, the operation of unit/value normalization is only applied to code with value. If the unit of the entry is consistent with the main unit of the code, the numeric value will be filled in the *value* attribute of the tuple; Otherwise, the value is discarded, and the *value* attribute of the tuple will be _STRING.

Some tables (like labevents and chartevents) are so large that the program divides them into several segments and generate tuples for one segment at a time to save memory space.

Through tuple generation, we get *tuples.csv* and *string_tuples.csv* containing all the tuples we need.

3.4 post-processing

Now we got *tuples.csv* and the draft of *code_dict.csv*. The post-processing is to:

Step 1: Generate a dictionary providing personal information of all the patients, namely *patients_dict.csv*.

Step 2: Re-count the frequencies of codes in *code_dict.csv* and add labels to *code_dict.csv* (it requires additional source files to add label, description, and category columns to the dictionary).

Why do we have to re-count the frequency of codes? For one thing, the cleaning rules of tuple generation might be slightly different from dictionary generation. For example, there are four entries containing invalid admission ID, which is not detected by dictionary generation because the program does not scan admission ID in the dictionary generation process. For another thing, when generating *patients_dict.csv*, you may add extra cleaning rules to delete some unwanted patients together with their tuples, which will change the frequencies of the lab code, too.

Post processing can also add label, description, and category columns to the dictionary, but it needs extra source files to do so. The code dictionary *code_dict.csv* provided with this document already contains the three columns.

When the post-processing is done, you will get a cleaned version of MIMIC IV constituted by *patients_dict.csv*, *string_tuples.csv*, *code_dict.csv*, and *tuples.csv*. You can modify this cleaned dataset or make more cleaning rules in the former procedures to let the data meet your need.

3.5 Some Tips

1. There are always some unexpected errors in the original MIMIC dataset. Hence, if you would like to conduct your own research with MIMIC, it is necessary to verify all the intermediate results in the cleaning process to find them.

2. *patients_dict.csv*, *code_dict.csv*, and *tuples.csv* are correlative of each other. Therefore, when you modify any one of them, make sure the data in all these files is consistent.