Exploring Vital Signs and Medications

Yoon Choi, Kelvin Encarnacao, Zhao Zhang

# **Introduction**

Patient clinical notes are an important piece of information that can be used to analyze health data to solve an important problem. Our objective in this study was to determine whether we can gain insights from the dataset in regard to vital signs/physical exam readings and medication across the dataset as a whole and across the dataset at a patient level. We got the data from Kumar et al. paper, extracted relevant information for the health data analysis, and finally created charts to analyze it.

# **Methods**

We were given dataset in XML format from Kumar et al. paper titled “Creation of a new longitudinal mix of clinical narratives.” The dataset is a mix of texts and tags. With the data in our hands, we analyzed the vital signs / physical exam readings through the free-text portions and the medications through the xml tagged portion. Our group decided the most efficient method for parsing and analyzing the dataset was to use Python as our programming language and write scripts that iterate through all the relevant information.

For analyzing vital signs / physical exam readings (question 1), we --- (TO BE DONE BY ZHAO).

For analyzing medication (questions 2-5), we converted the XML tags to CSV file. Putting all medication information in a single CSV file allowed for easier data parsing which saved both time and effort [1]. The relevant headers for the columns in the CSV file was patient\_id, visit\_id, medication\_id, start, end, time, text, type1, type2, and comment. Patient\_id and visit\_id are integers that are split by ‘-‘ from the name of each XML file. For example, patient\_id and visit\_id would be 100 and 01, respectively, if the file name was 100-01.xml. The rest of the headers are the same as from the medication tags. Utilizing Python’s csv libraries, the output process was a simple iterative function that took each row in the CSV file and outputting relevant results in dictionary. All rows that didn’t match the criteria for each related question were skipped. It was important for us to understand that each row has medication\_id that started with ‘DOC’ or ‘M’, which we call row in first category and row in second category, respectively. This information is important because the first category encompasses the second category and also has less information. The second category has more information but usually contain the same information if they are encompassed by the same first category.

For analyzing ------- (question 6), we --- (to be done by kelvin).

# 2.1 Question 1:

Explained in detailed by Zhao.

# 2.2 Question 2a: Freq. Distribution of the Medications Taken

This method returns a list of tuples where first element of the tuple is a medication name, and the second element is a frequency of the medication. We loop through only the rows that fit the second category (starts with 'M') in medication.csv. This is because the ones from the first category does not have the 'text' field that indicates the medications. We utilize the list 'intermerdiate\_medicines' to store 'text' field from each row in the second category until the next row's medication\_id starts with 'DOC' (first category), which then we will flush out the elements in the list. This means that this method doesn't double-count the medication if they are the same name and under the same first category but does not prevent from double-counting if they are from different first category. This means double-counting of the same medications is possible even if the only difference between the first category is the time. However, we left it as is because time is an important aspect of the medication frequency due to the possibility of getting on/off the drugs from before/during/after the DCT.

# 2.3 Question 2b: Freq. Distribution of the Medications Categories Taken

This method returns a list of tuples where first element of the tuple is a medication category, and the second element is a frequency of the medication category. We loop through only the rows that fit the first category (starts with 'DOC') in medication.csv. This is because the ones from the first category encompasses the ones from the second category that follows after. We store the 'type1' and 'type2' field from each row without regard to which type it is. We disregarded the different types and simply aggregated them because the medicines with two types still mean that the medicines are within the bounds of these two and we cannot ignore one another and doesn't mean the first type is more important than the second type. We also safely skipped over the second category because the first category rows encompasses the second category rows that follow.

# 2.4 Question 3 and 4: 10 Individuals taking the Greatest / Least Number of Medication Types

This method returns a list of tuples where first element of the tuple is a patient id and the second element is a frequency of the medication types for the said patient. Similarly with question 2b, we loop through only the rows that fit the first category (starts with 'DOC') in medication.csv. This is because the ones from the first category encompasses the ones from the second category that follows after. For each patient, we aggregate the type1 and type2 field of the rows as long as we haven't counted the medication type inside the patient's medication type taken. After we gathered all the medication types from a patient, we simply count the number of them and thus we get our result.

# 2.5 Question 5: Freq. Distribution of the Medications Taken

This method returns a list of tuples where first element of the tuple is a patient id and the second element is a frequency of the medication for the said patient. We loop through only the rows that fit the second category (starts with 'M') in medication.csv. This is because the ones from the first category does not have the 'text' field that indicates the medications. Similarly with question 2a, we utilize the list 'intermerdiate\_medicines' to store 'text' field from each row in the second category until the next row's medication\_id starts with 'DOC' (first category), which then we will flush out the elements in the list. This means that this method doesn't double-count the medication if they are the same name and under the same first category but does not prevent from double-counting if they are from different first category. This means double-counting of the same medications is possible even if the only difference between the first category is the time. However, we left it as is because time is an important aspect of the medication frequency due to the possibility of getting on/off the drugs from before/during/after the DCT. Then, similarly with question 4, after we have gathered all the medication names from a patient, we simply count the number of them and thus we get our result.

# 2.6 Question 6: Freq. Distribution of the Medications Taken

Explained in detailed by Kelvin.

Each of the functions that returned dictionaries as outputs were saved to CSV files using the custom python module named csvModule. This allowed for an easier graph creation and more intuitive result of our analysis. We also created a custom python module named printModule in order to get a sense of our data representative. The final data analysis was done under Excel.

# **Results**

# 3.1 Question 1 Result:

Explained in detailed by Zhao.

# 3.2 Question 2a Result: Freq. Distribution of Medications Taken

## 3.2.1

Figure 1: Frequency distribution of the medications taken.

## 3.2.2 Frequency Distribution of Categories of Medications Taken

Figure 2: Frequency distribution of categories of medications taken.

# 3.3 Question 3 Result:

10 individuals taking the greatest number of medication types.

|  |  |
| --- | --- |
| patient id | num of medication types |
| 125 | 13 |
| 216 | 13 |
| 281 | 13 |
| 400 | 13 |
| 100 | 12 |
| 106 | 12 |
| 115 | 12 |
| 156 | 12 |
| 177 | 12 |
| 184 | 12 |
| 196 | 12 |
| 202 | 12 |
| 237 | 12 |

# 3.4 Question 4 Result:

10 individuals taking the least number of medication types.

|  |  |
| --- | --- |
| Patient id | num of medication types |
| 176 | 2 |
| 160 | 3 |
| 142 | 4 |
| 174 | 4 |
| 246 | 4 |
| 251 | 4 |
| 259 | 4 |
| 307 | 4 |
| 318 | 4 |
| 326 | 4 |
| 369 | 4 |

# 3.5 Question 5 Result:

10 individuals taking the least number of medications.

|  |  |
| --- | --- |
| Patient id | num of medication |
| 174 | 4 |
| 176 | 4 |
| 251 | 4 |
| 246 | 5 |
| 259 | 5 |
| 264 | 5 |
| 307 | 5 |
| 213 | 7 |
| 248 | 7 |
| 109 | 8 |
| 181 | 8 |
| 186 | 8 |
| 267 | 8 |
| 275 | 8 |
| 326 | 8 |
| 381 | 8 |

# 3.1 Question 6 Result:

Explained in detailed by kelvin.

# Limitations

First of all, double-counting of the same medications is possible for the frequency questions even if the only difference between the medications is ‘time’. This is due to the fact that tags whose medication\_id starts with 'DOC' differ from one another only by the 'time' field while medication tags inside are identical. However, we must see this not only as limitations, but also by design, because time is an important aspect of the medication frequency due to the possibility of getting on/off the drugs from before/during/after the DCT.

Secondly, we constructed the frequency results based on their relevant strings, so if the 'text' field had any discrepancies from the representatives due to spelling errors or long strings, the medicine didn't match well enough to contribute to the frequency. For example, we couldn't make sense of 'dilt (30mg TID)' from 114-04.xml so we left them out. This could mean that we were missing some relevant medicines. However, our dataset was large enough to justify filtering through the anomalies and we are confident that we had enough data to make a proper analysis.

# Conclusions

There are various information available through patient clinical notes dataset that we can use for health data analysis.

# Bibliography

1. Bibliography here.

# Bibliography

*docs.python.org*. (n.d.). Retrieved from https://docs.python.org/3/library/csv.html