Identifying Diseases using Symptoms

Deepkumar Patel
DATA-51000-001, Summer 2020
Data Mining and Analytics
Lewis University
deepkumarpatel@lewis.edu

I. Introduction

The dataset that I have found includes diseases and their related symptoms in a tabular format. The data source also includes symptoms description, symptoms precautions, and symptoms effectiveness on a patient for two days. The description, precautions, and effectivity are all in a separate tabular format. The author of the data source has not released the link for the original data set, however, only the subset is provided by the author. The dataset is available on the Kaggle website [1], which is the world's largest site for data scientists. The author provided the data under a creative commons license, which is free to use, share, and adapt.

The main purpose of this report is to create a system that can find diseases, given a set of symptoms. The dataset itself may have come from real patients, however, this is not cleared by the author. Nevertheless, the symptoms and diseases provided in the data set do make sense logically. The methods which will be used to find the association rules are, the Frequent Pattern (FP) Growth algorithm [2], and the Apriori algorithm [3]. Specifically, this report uses Orange as a tool, which uses the FP algorithm with bucketing optimization [4] for efficiency. Weka is another tool that uses the Apriori algorithm to find association rules for our given dataset.

The future sections of this report describe the data preprocessing, dataset, two different methodologies using two different tools, results gathered from both methods along with a discussion, and a conclusion. Annexed is a list of sections with a more detailed description. Section II describes the process that is involved prior to any analysis made on this dataset. Section III describes the dataset after preprocessing, which will then get fed into two different tools for rule mining. Section IV describes the Apriori algorithm along with an example, the understanding of different measurements, and also goes over the association rules resulted using the Weka tool. Section V describes the FP Growth algorithm and different measurements derived using the Orange tool. Section VI compares both the methods along with a discussion on how this could be implemented as a full-fledged disease finder application. Section VII is the conclusion of this report.

II. DATA PREPROCESSING

The original raw dataset contained 4,920 instances and 18 attributes [1]. However, for the purpose of this report, diseases with four or less than four symptoms were extracted. The reason for extracting only less than or equal to four symptoms is because the frequency of four symptoms is the highest as shown in Fig 1. This does not mean we cannot use a greater number of symptoms. We can definitely apply any number of attributes, however, for the purpose of this report, it would be difficult to find the association rules later on as we apply different algorithms. Data preprocessing is done using python in Jupyter notebook, which is included with the submission of this report. In order to extract diseases with four or less than four symptoms the following steps were taken:

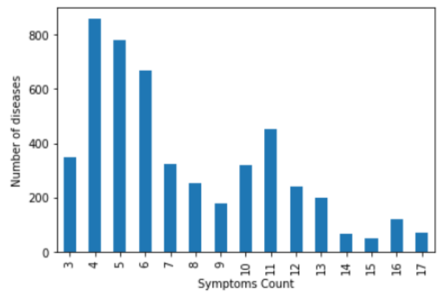

Fig. 1. Number of diseases per symptoms count

A. Getting Count

A lambda expression is used to get the count of all the symptoms per disease, excluding null values. Doing so will give us the count of symptoms, which are then mapped with the original data frame.

B. Extraction

Since we have the count mapped to the data frame, using a condition, diseases that have more than four symptoms are then easily removed from the data frame. Then, the needed columns are extracted into a new data frame, this is done using python's iloc property. The needed columns are the disease and symptoms column from one to four.

C. Arrange a Column

This step is solely done for the Weka tool because the class index if set to -1, the last attribute is taken as the class attribute in configuring the Apriori algorithm. Therefore, using python's pop method, the disease column is popped and placed at the end of the data frame.

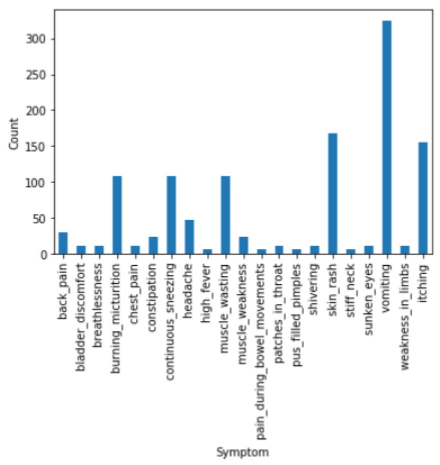

Fig. 2. Counts per symptoms 1

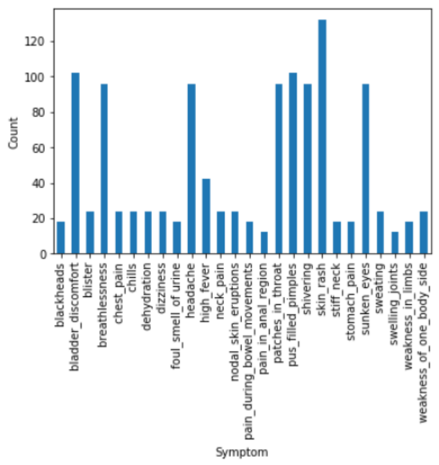

Fig. 3. Counts per symptoms 2

D. *Exporting Data*

The final step is to export the data frame into an excel document so that it can be imported into Weka and Orange. Other than the above steps, this dataset did not require any additional processing or cleaning and the final size we are left with, includes 1,206 instances and 5 attributes. The only null values that were included were symptoms of some diseases, which make sense, because not all diseases have the same number of symptoms.

III. DATA DESCRIPTION

All the attributes that are used in this report are the nominal type. After the data preprocessing step, we are left with the following attributes shown in Table I.

TABLE I
DISEASE AND SYMPTOMS

Attribute	***Type***	***Example Value***	***Description***
Symptom 1	Nominal (string)	Vomiting	Type of Symptom
Symptom 2	Nominal (string)	Bladder Discomfort	Type of Symptom
Symptom 3	Nominal (string)	Sweating	Type of Symptom
Symptom 4	Nominal (string)	Chest Pain	Type of Symptom
Disease	Nominal (string)	Heart Attack	Type of Disease

As mentioned earlier, only four symptoms will get analyzed along with diseases; this does mean that our final analysis will miss out on more than half of the diseases. However, the choice is made only for the purpose of this report, the methods that are detailed in the future sections can easily be applied to any number of attributes, it is just the matter of picking the correct algorithm. In order to better understand the dataset, let's take a look at Fig.2 and Fig.3.

Visually, we can see that vomiting and skin rash are the symptoms with the highest count. Perhaps one can ask a question by including vomiting and skin rash as an antecedent (if) and a consequent would be a disease (then), or maybe the consequent can be another symptom. Such a condition could happen possibly because of emotional distress or simply because the symptoms are related in nature. There are many questions one can ask about this dataset. Aside from the above plots, additional plots are created in the Jupyter notebook.

IV. APRIORI ALGORITHM IN WEKA

Apriori algorithm is used to find frequent itemsets, assuming inputs are given from a transactional database. The output is the set of itemset that has the support of no less than the minimum support threshold. This algorithm iteratively scans the database a number of times, until it reaches a point where no frequent itemset is further found. At every scan, it calculates the support, confidence, and lift. There could be more measurements depending on the implementation and the tool. It is an array-based algorithm that requires large memory space if the candidate generation is more. Therefore, this algorithm is very slow and the bottleneck is candidate generation.

Let us take a look at a simple example of how this algorithm works on our given dataset. For Table II consider TID as each transaction with the minimum support of 2 and minimum confidence of 0.5.

TABLE II
PATIENT AND SYMPTOMS

TID	***Symptoms***
1	S1, S2, S3
2	S2, S4
3	S1, S2, S4

A. *Apriori Scan-1*

Using Table II, we first scan each symptom to get the unique values, then get the support of each itemset to create the first candidate set, as shown on the left table of Table III.

TABLE III
APRIORI SCAN 1

Candidate set 1 (C1)		Large Itemset 1 (L1)	
Itemset	***Support***	***Itemset***	***Support***
{S1}	2	{S1}	2
{S2}	3	{S2}	3
{S3}	~~1~~		
{S4}	2	{S4}	2

TABLE IV
APRIORI SCAN 2

Candidate set 2 (C2)		Large Itemset 2 (L2)	
Itemset	***Support***	***Itemset***	***Support***
{S1, S2}	2	{S1, S2}	2
{S1, S4}	~~1~~		
{S2, S4}	~~1~~		

B. Apriori Scan-2

Compare the support of each itemset in C1 with the minimum given support. We see that the support of S3 is crossed out, because it supports are less than the minimum support. Therefore, the only eligible itemset is S1, S2, and S4 as shown on the right table of Table III.

C. Association Rule

Using L1 generated in step 2, we create C2, which is generated by joining and pruning the itemset. After that, we calculate the support of each itemset in C2 using the original dataset. The result we would get is shown on the left table of Table IV. Again, we follow a similar procedure of eliminating the itemset with the support that has less than the minimum support. The only itemset that is left is {S1, S2}, therefore we stop here, as there is no more frequent itemset.

D. Rule and Measures

The only association rule we could generate is S1→S2. The confidence implies how likely S2 can happen if S1 were to happen. In this case, we could have also created a rule where S2→S1 , but the confidence would turn out to be the same as S1→S2. The downside of using confidence, is that it only accounts for how frequent S1 or S2 is. That is why we need to use the lift as one of the measures. It implies that how likely S2 can happen if S1 were to happen, and at the same time, combining the frequency of both S1 and S2. Generally, the lift can be examined in three different ways. If the lift is less than 1, then S2 is unlikely to happen if S1 happens, means they are negatively correlated. If the lift is equal to 1, then there is no association between the items. If the lift is more than 1 then S2 is likely to happen if S1 happens. In our example, we can see that lift is less than 1, which means that symptom 2 is unlikely to happen, given that symptom 1 were to happen.

TABLE V
ASSOCIATION RULE AND MEASURES

Association Rule	***Confidence***	***Lift***
S1→S2	$\frac{Support(\{S1, S2\}}{Support(\{S1\})} = 1$	$\frac{Support(\{S1, S2\})}{Support(\{S1\}) * Support(\{S2\})} = 0.333$

Now that we have a basic understanding of this algorithm and the measures, let's take a look at the actual rules that were generated using Weka.

The main purpose of this report is to find a disease with given symptoms. As mentioned earlier, the disease column was moved at the end because Weka's class index by default is set to -1 which is considered as the last column. The reason for doing this is because we want the disease to be the consequent, and this approach will force Weka to mostly generate rules that have diseases on the right-hand side, at least this is the case with this dataset. While setting different measures in Weka, it tends to produce tons of association rules, therefore after many trials and errors, annexed is a list of measurements that worked best for this report. Minimum support was set to 0.013, metric-type was set to Confidence, minimum confidence was set to 0.1, and the number of rules was set to 2,000. For more details on full property selection, take a look at Table VI. It ended up generating more than 2,000 rules, and most of them are not useful for our analysis. Therefore, I only observed rules that have a disease as a consequent. Having said that, all the other rules are also perfectly valid, but not useful to our analysis.

TABLE VI
WEKA APRIORI CONFIGURATION

Property	***Value***
Car	False
ClassIndex	-1
Delta	0.05
DoNotCheckCapabilities	False
LowerBoundMinSupport	0.013
MetricType	Confidence
MinMetric	0.1
NumRules	2000
OutPutItemSets	False
RemoveAllMissingCols	False
SignificanceLevel	-1.0
TreatZeroAsMissing	False
UpperBoundMinSupport	1.0
UpperBoundMinSupport	False

For the analysis, I picked two rules from a pool of 2,000 rules. We can see how they are similar, but only one of them is valid.

TABLE VII
ASSOCIATION RULE AND MEASUREMENT USING APRIORI

Rule	***Antecent***
1	{Skin rash, high fever, red sore around nose}
2	{blister, red sore around nose, yellow crust ooze}

Rule	***Consequent***	***Confidence***	***Lift***	***Leverage***
1	Impetigo	1	28.71	0
2	Impetigo	1	28.71	0.02

Let us analyze the results given in Table VII. For both rules, the confidence is 1, which means that given these symptoms, it is likely that the disease will be impetigo. As mentioned earlier, confidence is not always a good measure, therefore let's take a look at the lift. It turns out the lift is also similar for both, 28.71 is a positive number, that means that given these symptoms, it is highly likely that the disease is impetigo. However, the leverage for both is different. Now, what is Leverage? Assuming antecedent and consequent are independent, leverage computes the difference between the probability of a given rule to occur and its expected probability. If the leverage is negative, or below 0, then it is negatively correlated. If the leverage is 0, then it is statistically independent. If the leverage is positive or greater than 0, then it is positively correlated, and this is what we want our rule to have. As we can see, rule 1 is invalid in this case, because the leverage is 0. Therefore, the valid rule would be rule 2. Logically, if we take a look at the symptoms, it makes sense, because high fever would have nothing to do with impetigo disease. The combination of lift and leverage would tremendously benefit a system to pick the correct rule. The only issue with association rule mining is it creates many rules, and to able to go through each rule is time-consuming. In the later section, we will see how we can get rid of most of the rules by using the number of symptoms.

V. FREQUENT PATTERN GROWTH ALGORITHM IN ORANGE

Frequent pattern Growth (FP) algorithm generates an FP tree by compressing the database into a tree format. Each node of the tree represents the item along with it's support, and each branch represents the association of the item [3]. The tree is compressed in a sense, because there are many transactions in the market basket that often share an item in common. This also means that if the transactions have no common items, then there will be no compression in the output tree. We know, to store the database, it uses a prefix-tree structure, which means that it actually requires less memory space.

Orange is used as a second tool for this report, as it uses the FP algorithm. To be more specific, an add-on was downloaded for the association rule mining. The minimum support was set to 1%, and the minimum confidence was set to 93%. The antecedent filter was added to have minimum items to 4, because in Fig.1 we observed that 4 is the highest count for the symptoms, and I wanted to see if we could get all four symptoms, even though for the same disease, we have a different number of symptoms. The consequent also added a filter to check if it contains the string "Disease=", this way we get diseases on the right-side, and symptoms on the left side. Just as a reminder on why we do this, it is because we want to find out a disease, given any symptoms. The rules that were generated using Orange were shocking in a way, it didn't take much effort to find the diseases.

Given the above constraints, it produced the following results. In total there are 9 rules, but in this report, only 2 of them are shown. More rules are available by opening the Orange file submitted with this report.

TABLE VIII
ASSOCIATION RULE AND MEASUREMENT USING FP GROWTH

Rule	***Antecent***
1	{Skin rash, blister, red sore around nose, yellow crust ooze}
2	{Vomiting, breathlessness, sweating, chest pain}

Rule	***Consequent***	***Confidence***	***Lift***	***Leverage***	***Coverage***
1	Impetigo	1	28.714	0.014	0.015
2	Heart Attack	1	10.050	0.058	0.065

For both of the above rules in Table VIII, lift and leverage are high, and as mentioned earlier, the higher these values the better positive correlation we have. Therefore, we can conclude that the antecedent (Symptoms) is a good predictor of the consequent (Disease), at least that is the case for the above rules. We also have coverage, which essentially is the same as the support. Support is useful when you want to find a relative measure within your dataset. Therefore, we cannot rely only on support for the selection of frequent itemset but for this report, leverage and lift were the most reasonable measurements.

VI. RESULTS AND DISCUSSION

Before we compare the two methods, let's compare the tools. I find Orange to be more controllable in terms of applying string filters to the antecedent and consequent. As far as I'm aware, there is no string filtration in Weka. String filtration makes it easier to see and analyze your data. You should not have to see 2,000 rules and go through them one by one to find what you are looking for. Orange also shows all the association rules in a nice list format. In Weka, sometimes it gets difficult to see the results. In Weka, you can type the min. and max. values, therefore you can be more precise with your input parameters. Weka definitely has more input parameters, which you can precisely control to get your results. In my opinion, both tools are great and useful, and would be the best way to use both the tools in conjunction with better analysis.

Let's compare the Apriori algorithm and the FP growth algorithm. In terms of understanding the inner working of the Apriori algorithm, I think it's very easy to understand and implement. The down-side of using this algorithm is it becomes really slow especially when there are more candidates in the dataset, as the nature of this algorithm is it requires multiple scans of the database [2]. However, I did not notice any time delay while working in Weka, it could be because we only have 1,206 transactions. One way to avoid generating more candidate itemset, is to increase the support threshold.

FP growth algorithm is an efficient algorithm as compared to Apriori, it only requires scanning the database twice to generate frequent pattern tree. As mentioned earlier, it also requires less space because of how the tree is stored in the memory. Implementing this algorithm could get a little difficult, as there are many steps that are very involved before it outputs any association rule. Apriori uses a breadth-first search, an array-based queue where backtracking is not allowed. On the other hand, FP growth uses a depth-first search where the tracking is allowed as it uses the stack [2].

The rules generated by both algorithms are perfectly valid. Even though Weka generated 2,000 rules, it makes sense, because the minimum support was set too low, therefore, Apriori ended-up generating more candidate sets. Going back to the example of the impetigo disease, both the methods generated similar lift, however, a different number of leverages. The difference in leverages can be easily spotted as to why that is the case. The Apriori algorithm missed "skin rash" as one of the antecedents, which interns made the probability of both X and Y to go down. If more rules were to analyze, it is likely that we would have found this missing rule with 4 symptoms, however, mostly, in association rule mining, we don't know what we are looking for until we observe something interesting.

Before we talk about how association rule mining is useful in healthcare service, let's take a look at how we can avoid generating thousands of rules. In Fig.1 we observed, that for each disease, there's a certain number of symptoms associated with them. To understand how this information is useful, let's take a look at Table IX.

TABLE IX
SYMPTOM COUNT AND DISEASE

Symptom Count	*AIDS*	*Arthrities*	*Heart Attack*
3	1	0	0
4	1	1	0

Each disease can either have 3, 4, or more symptoms. We can quickly observe that if the number of symptoms is known, then we can easily eliminate the diseases where the symptom count is zero. For example, let's assume we have a patient who showed 3 symptoms.Using Table IX, the only disease that has 3 symptoms is Aids, therefore, we eliminate the other two diseases. This is helpful, because now we only generate rules in which the consequent is Aids, and internally this will save more time generating the candidate set. In such cases where we find multiple diseases, we look at the lift and the leverage, and whichever disease has the highest measurement would have more precedence than the other. Another simple way, is to pass the symptoms as input parameter, and filter the antecedent by symptoms. The issue with this solution is, if you have N number of symptoms, then string comparison gets really expensive, and it can tremendously slow the system down. Fig.4 shows how this entire process can be implemented.

VII. CONCLUSION

The main goal of this report is to figure out a disease, given a number of symptoms. Using association rule mining, this is possible because it helps us find rules which are positively correlated and sometimes negatively correlated. That is why we made use of different measurements such as lift and leverage. We also saw how confidence and support are not of as much use in this dataset. Finally, we come up with an idea of using the symptoms count to eliminate unnecessary rules. This method can not only output diseases, but also shows a strong relationship between different symptoms. This invaluable information can be used to build more knowledge and data for future analysis. Data analytics in healthcare is a tremendous asset in the field of medicine. It can associate symptoms to predict diseases, or help healthcare professionals to pinpoint the possibility of whatever disease or sickness a patient has at a given time. It cannot, however, take away the expertise of the physician, or his or her education in this field. It can assist physicians with correlation between symptoms, association of symptoms, habits, and diseases among a broad spectrum of patients and conditions. A good example of this would be diseases that are novel; by entering the patients symptoms that they are experiencing with a new sickness, over a period of time, physicians, scientists, researchers can use this data to aid them in finding antidotal medicines, procedures and vaccines to combat these new diseases or viruses.

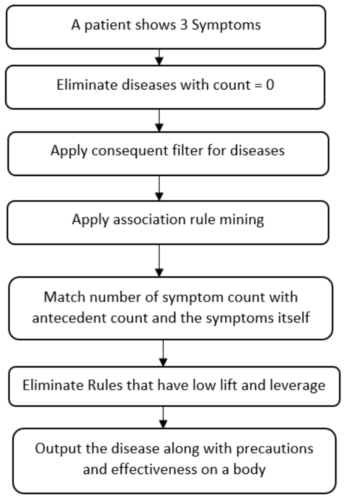
A patient shows 3 Symptoms
Eliminate diseases with count = 0
Apply consequent filter for diseases
Apply association rule mining
Match number of symptom count with antecedent count and the symptoms itself
Eliminate Rules that have low lift and leverage
Output the disease along with precautions and effectiveness on a body

Fig. 4. Symptoms and diagnosis

REFERENCES

[1] P. Patil. "Disease symptom prediction." Kaggle, 24 May 2020, www.kaggle.com/itachi9604/disease-symptom-description-dataset.
[2] J. Han, P. Jian, Y. Yiwen, M. Runying, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach". 21 May 2001, pp. 61-69.
[3] A. Rakesh, S. Ramakrishnan. "Fast algorithms for mining association rules".
[4] R. Agrawal, C. Agrawal, V.Prasad, (2000) "Depth first generation of long patterns".