IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 4, April 2020

Attribute Selection to Identify COVID-19 Disease by using Data set Condensing and Intersection Pruning

Murlidher Mourya[1], D.Ganesh[2]

Assistant Professor, CSE, Vardhaman College of Engineering, Hyderabad, India[1]

Assistant Professor, CSE, Vardhaman College of Engineering, Hyderabad, India[2]

Abstract: Mining of frequent item sets is one of the most fundamental problems in data mining applications. My proposed algorithm which guides the doctor to select the symptoms of a new patients , due to testing constraints there is a limit, say m, on the number of symptoms that can be selected for the identify the covid-19 disease. Although the problems are NP complete. The Approximation algorithm is based on greedy heuristics. DCIP algorithm uses data-set condensing and intersection pruning to find the maximal frequent symptoms. This algorithm differs from all classical maximal frequent item set discovering algorithms; experiments show that this algorithm is valid with moderate efficiency; it is also easy to code for use in KDD applications.

Keywords: Association rules, Data mining, Mining frequent item sets, intersection pruning, and data-set condensing

I. INTRODUCTION

In December, 2019, a local outbreak of pneumonia of initially unknown cause was detected in Wuhan (Hubei, China), and was quickly determined to be caused by a novel corona virus namely severe acute respiratory syndrome corona virus (SARS-CoV-2). The outbreak has since spread to every province of mainland China as well as 27 other countries and regions, with more than 70 000 confirmed cases as of Feb 17, 2020. In response to this ongoing public health emergency, they developed an online interactive dashboard, hosted by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University, Baltimore, MD, USA, to visualise and track reported cases of corona virus disease 2019 (COVID-19) in real time. The dashboard, first shared publicly on Jan 22, illustrates the location and number of confirmed COVID – 19 cases, deaths, and recoveries for all affected countries. It was developed to provide researchers, public health authorities, and the general public with a user- friendly tool to track the outbreak as it unfolds. All data collected and displayed are made freely available, initially through Google Sheets and now through a GitHub repository, along with the feature layers of the dashboard, which are now included in the Esri Living Atlas.

We believe our efforts are crucial to help inform modeling efforts and control measures during the earliest stages of the outbreak. Corona virus disease 2019(COVID-19) is spreading rapidly across China, and as of Feb 16, 2020, had been reported in 26 countries globally. The availability of accurate and robust epidemiological, clinical, and laboratory data early in an epidemic is important to guide public health Decision – making Consistent recording of epidemiological information is important to understand trans- missibility, risk of geographic spread, routes of transmission, and risk factors for infection, and to provide the baseline for epidemiological modeling that can inform planning of response and containment efforts to reduce the burden of disease. Furthermore, detailed information provided in real time is crucial for deciding where to prioritize surveillance.

I. SYMPTOMS SELECTION

Consider the following scenario: If a doctor wants to identify a new Covid-19 patient. He has to identify the most frequent symptoms that are the most of the Covid-19 patient have. If new patient has given an extra some other symptoms, If he has a system, that can suggest top k symptoms of the Covid-19disease, then doctors can easily identify the new patient is affected by Covid-19 or not. General problem also arises in domains beyond e-commerce applications. *The problem here is selecting the proper and the most frequent symptoms of Covid-19 disease.*

To define our problem more formally, we need to develop a few abstractions. Let D be the database of already being Covid -19 patient details. Q be the set of search queries that have been executed against this database in the recent past—thus Q is the "workload" or "query log." The query log is our primary model of what online Covid-19 patient's symptoms are recorded. For new symptoms that need to be inserted into this database, we assume that the doctor has a complete

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 4, April 2020

"ideal" description of the new symptoms of Covid-19.due to testing constraints there is a limit, say m, on the number of symptoms that can be selected for the identify the covid-19 disease and .there is also a limit, say m, on the number of symptoms/keywords that can be selected for entry into the database. Our problem can now be defined as follows.

II. PROBLEM FRAMEWORK

Given a database D, a query log Q, a new tuple t, and an integer m, determine the best (i.e., top-m) attributes of t to retain such that if the shortened version of t is inserted into the database, the number of queries of Q that retrieve t is maximized.

PRELIMINARIES:

First we provide some useful definitions Let Covid -19 symptoms/ attribute set A={ Fever, Dry cough, Shortness of breath, Muscle pain, Headache, Sore throat, Loss of taste or smell, Diarrhea, Runny Noise, Nasal Congestion} are represented as {I1,I2,I3,I4.I5,I6,I7,I8,I9,I10} respectively.

Table 1: Database

Patient ID /Symptoms	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
1	0	1	0	1	0	0	0	1	0	1
2	0	1	1	0	0	1	0	0	0	1
3	1	0	0	1	1	0	1	1	1	0
4	1	1	0	1	1	1	0	0	1	0
5	1	1	0	0	0	1	0	0	0	1
6	0	1	0	1	1	0	0	1	0	1
7	0	0	1	1	0	0	0	1	0	0

Boolean database

Let D = {t1 . . . tN} be a collection of Boolean tuples over the attribute set A = {a1 . . . aM}, where each tuple t is a bit-vector where a 0 implies the absence of a feature and a 1 implies the presence of a feature. A tuple t may also be considered as a subset of A, where an attribute belongs to t if its value in the bit-vector is 1.

Table 2: Query Log Q

T ID	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
001	1	1	0	1	1	0	1	0	0	0
002	1	1	0	0	1	1	1	0	0	0
003	0	0	1	0	1	0	1	0	0	1
004	0	0	1	0	0	0	0	1	0	1
005	1	1	1	1	0	0	1	0	0	0
006	0	1	1	0	0	0	1	1	0	0
007	0	0	1	0	0	1	0	0	1	0
008	1	0	1	0	1	0	0	0	1	0
009	1	1	0	0	0	1	0	0	0	0
010	0	0	1	1	0	0	0	1	1	0

Table 3: New Tuple T to be inserted

New Patient ID	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10
8	1	1	1	1	0	1	1	0	0	0

DOI 10.17148/IARJSET.2020.7419

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 4, April 2020

Selecting the threshold value:

There are two alternate approaches to setting the threshold. One approach is essentially a heuristic, where we set the threshold to a reasonable fixed value dictated by the practicalities of the application. Threshold enforces that attributes should be selected such that the compressed tuple is satisfied by a certain minimum number of queries. For example, a threshold of 1 percent means that we are not interested in results that satisfy less than 1 percent of the queries in the query log.

Frequent: Item sets whose support is greater than min_sup are classified as frequent Infrequent: Item sets whose support is less than min_sup are classified as infrequent Unclassified: All other Item sets are said to be unclassified.

IV PROPOSED TECHNIQUE: DCIP ALGORITHM

The first step of DCIP algorithm is to reduce the length of item sets and the volume of data-set. According to Lemma 1, any maximal frequent item set is also a maximal frequent item set corresponding to one transaction in D, so find all maximal frequent item sets correspond to every transaction through intersection pruning, merge them into one set (denoted as FS hereinafter), then delete all infrequent maximal item set in FS, and the remaining set is maximal frequent item set. The two main processes are described as follows.

A. Condensing the Data-set

This process first sorts the data-set with descending order according to the length of its item sets, then moves those high-dimensional transactions whose support are bigger than minimal support threshold to a frequent item set, and deletes all subsets of those transactions to condense the data-set. These steps are as follows:

Step 1: Scan the data-set, finding all frequent 1- item set;
Step 2: Scan the data-set, deleting all items infrequent 1-itemset from all transactions; then add up identical transactions (i.e., if transaction T1=T2, let support(T1) = support(T1) + support(T2), and delete T2 from data-sets). Sorting the data-set descending according to the length of item sets to form a new data-set which we denote as C; Step 3: Process every transaction Ti in C whose support are bigger than minimal support threshold: move T_i to FS and delete all T_j ($T_j \subset T_i$, $>i$);
Step 4: Delete non-MFI from FS; Step 5: End.

B Intersection Pruning

Any maximal frequent itemset is also the maximal frequent itemset corresponding to a certain transaction in D; merge all maximal frequent itemset corresponding to every transaction into one set (which we denote as FS), then delete all non-frequent maximal itemsets in FS, and the remaining set is the maximal frequent itemset. These steps are as follows: Assume we have a data-set denoted as D, and the minimal support threshold is S.

Step 1: Condense data-set D using the method described in 3.1; if $|D|<S$, terminate the processing for the current data-set;
Step 2: Find intersection of T_1 and T_i($1<i\leq n$); merge all intersections into a new data-set D1; establish the vertical data format of D; delete transaction T_i ($T_i \subset T_1$);
if $|D1| \geq S$, then go to step 1 to perform another intersection pruning circle for D1;
Step 3: Use the vertical data format of D to find the intersection of T_j and T_i ($j=2, 3, 4, ..., m<n$; $j<i\leq n$), merge all intersections into a new data-set D1, go to step 1 to perform another intersection pruning circle for D1; when the volume of the remaining data-set is less than S, stop finding intersections of T_j and T_i, terminate the process for current data-set.
Step 4: End;
Note: Data-set condensing can be performed at the beginning of the intersection pruning process, as well as in the process of step 3

C .Instance Analysis

The following example shows how to discover MFI using DCIP for transaction database D (Table I) with minimum support threshold as 4 (i.e., minsup=4).

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 4, April 2020

TABLE I. TRANSACTION DATA-SET D

Patient ID	Symptons
001	I1,I2,I4,I5,I7
002	I1,I2,I5,I6,I7
003	I0,I3,I5,I7
004	I0,I3,I8
005	I1,I2,I3,I4,I7
006	I2,I3,I7,I8
007	I3,I6,I9
008	I1,I3,I5,I9
009	I1,I2,I6
010	I3,I4,I8,I9

Step 1: Condense transaction data-set D using the method in 3.1, the result is shown in Table II;

TABLE II. RESULT OF CONDENSED D

PatientID	Symptoms	Count	del
1	I1,I2,I5,I7	2	
2	I1,I2,I3,I7	1	
3	I3,I5,I7	1	
4	I2,I3,I7	1	1
5	I1,I3,I5	1	
6	I1,I2	1	1

Attribute Count is the count of corresponding transactions; attribute del indicate whether the corresponding transaction can be ignored in later processing, for example, after step 2, T6 can be ignored.

Step 2: Find intersections of T_1 and T_i (i=2, 3... 7), merge all intersections into data-set D1, as shown in Table III:

TABLE III: INTERSECTION DATA-SET FOR T1 IN TABLE II

Patient ID	symptoms	Count	del
1	I1,I2,I7	1(+2)	
2	I5,I7	1(+2)	
3	I2,I7	1(+2)	1
4	I1,I5	1(+2)	
5	I1,I2	1(+2)	1

Establish vertical data format for D; because in Table II, $T_6 \subset T_1$, T_6.del=1 (see Table II) . The (+2) for attribute Count in table III is the count of T_1 in Table II.

Step 3: Condense the data-sets in Table III; as this example, the result remains no change.

DOI 10.17148/IARJSET.2020.7419

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 4, April 2020

Step 4: Find intersections of T_1 and T_i (i=2, 3, 4, 5) in Table III respectively, merge them into a new data-set D1, as shown in Table IV.

TABLE IV.:INTERSECTION DATA-SET FOR T1 IN TABLE III

Patient ID	Symptoms	Count	del
1	I2,I7	1(+2+1)	
2	I1,I2	1(+2+1)	

Because T_3 and T_5 are subset of T_1 in Table III, delete T_3 and T_5;

Step 5: Condense the data-set in Table IV, produce frequent itemset {{I2, I7}:4, {I1, I2}:4}; Table IV is now empty after condensing;

Step 6: Back to Table III, T_3 and T_5 has been deleted, we only need to find the intersection of T_2 and T_4; but the length of T_2 and T_4 are both 2, no need to find intersection of them.

Step 7: Back to Table II, because T_6 has been deleted, we only need to find the intersections of T_2 and T_i (i=3, 4, 5); merge all intersections into a new data-set D1, as shown in Table V.

Because $T_4 \subset T_2$ in Table II, it should be deleted. Step 8: Condense the data-set in Table V; after Condensing the result is empty;

Step 9: The original data- set D has 10 transactions; Table II shows that 30% (3 transactions) of them has been processed; condense again the remaining data-set in Table II, and the result is empty. The process ends.

Step 10: Merge all resulting frequent item sets, and delete all non-frequent maximal item sets, the final result of MFI is {{I2, I7}: 4, {I1, I2}: 4}.

The steps above use 14 times of intersection calculations for MFI; compared with other Apriori-like algorithms, its simplicity and efficiency is explicit.

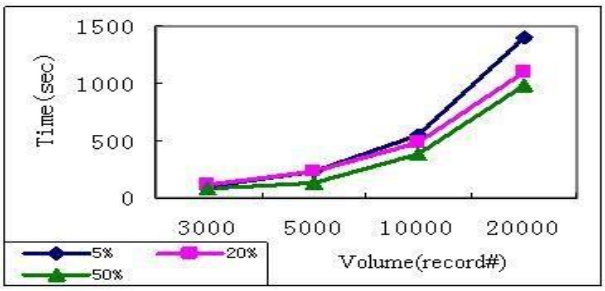
Note: Because the volume of the example data-set D is small (only 10), the above process does not include the utilizing of vertical data format; the reason of introducing vertical data format is to reduce the number of times of finding the intersections.

TABLE V.:INTERSECTION DATA-SET FOR T2 IN TABLE 2

Patient ID	Symptoms	Count	del
1	I3,I7	1(+1)	
2	I2,I3,I7	1(+1)	
3	I1,I3	1(+1)	

III. CONCLUSION

DCIP algorithm is its easy implementation. It is coded and tested using PowerBuilder script language on a microcomputer with Pentium IV/1.80GHz CPU, 512M memory running Windows XP operation system. Testing dataset is extracted from a supermarket's sales record.3000, 5000, 10000 and 20000 transactions are tested respectively with each record having 2-10 categories of commodity (the average number of categories is 6). Figure 1 shows the running time for different volume of datasets with minimum support threshold of 5%, 20% and 50%respectively. The bigger minimum support threshold, the lesser time needed for MFI.



While the problems considered in this paper are novel and important to the area of ad hoc data exploration and retrieval, we observe that our specific problem definition does have limitations. After all, a query log is only an approximate surrogate of real doctor preferences, and moreover, in some applications neither the database, nor the query log may be available for analysis; thus, we have to make assumptions about the nature of the competition as well as about the doctor preferences.

IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 7, Issue 4, April 2020

REFERENCES

[1].M.R. GAREY AND D.S. JOHNSON (1979), "COMPUTERS AND INTRACTABILITY: A GUIDE TO THE THEORY OF NP-COMPLETENESS".
[2].M. Miah, G. Das, V. Hristidis, and H. Mannila, (2008) "Standing Out in a Crowd: Selecting Attributes for Maximum Visibility," Proc. Int'l Conf. Data Eng. (ICDE), pp. 356-365.
[3]. WHO. WHO statement regarding cluster of pneumonia cases in Wuhan, China. Jan 9, 2020. https://www.who.int/china/news/ detail/09-01-2020-who-statement-regarding- cluster-of-pneumonia-cases-in-wuhan-china (accessed Feb 11, 2020).
[4]. Morgan O. How decision makers can use quantitative approaches to guide outbreak responses. *Philos Trans R Soc B Biol Sci* 2019; 374: 20180365.