Data Mining In Negative Database

Mohamed Ibrahim[1], Hatem Abdul-Kader[2,] Mohiy Hadhoud[2] and Ali Abouzied[1]

[1]High Institute for Computer Science and Information Systems, Egypt
[2] Faculty of Computers and Information, Menoufiya University, Egypt

Abstract: *Over the few recent years knowledge discovery (data mining) has been establishing itself as one of the major disciplines in computer science with growing industrial environment. Research in data mining will continue and even increase over coming decades. In the other hand the need for data to still private against any hacking still growing dramatically.*
This paper shares in the way of data privacy applications through applying data mining techniques on negative database. The most important concept of a negative database is that a set of records DB is represented by its complement set. That is, all the records not in DB are represented, and DB itself is not explicitly stored. It is shown that a database consisting of n records, l-bit-length records can be represented negatively using O(ln) records. It is also clear that membership queries applied on DB can be processed over the negative representation itself in time no worse than linear in its size and that reconstructing the database DB represented by a negative database NDB given. Assume total universe U of finite-length records (or strings), all of the same length L, and defined over a binary alphabet. We logically divide the space of possible strings into two disjoint sets: Positive or real database DB representing the set positive records (holding the real information of interest), and U −DB denoting the set of all strings not in DB. We assume that DB is uncompressed (each record is represented explicitly), but we allow U − DB to be stored in a compressed form called NDB. We refer to DB as the positive database and NDB as the negative database.

Keywords: *Data mining, Negative database, Security, Privacy, positive data base, association rule*

Received April 11, 2011; Accepted February 29, 2012

1. Introduction

Large size of data in both dimensions vertical and horizontal is increasing, and the demands we have to keep track of them continue to increase. We expect data to be available on demand and to be protected from unauthorized parties; we would like to have the ability to mine these data collections in new ways searching for noble, non-trivial and important knowledge to have the ability to take nontraditional decisions. In the other hand keeping the data privacy enabled, which means apply mining functions in a secure format. Content and the rules for accessing it must be continually updated and, eventually, we want the ability to audit the uses to which our personal data are put, Which means we should be able to query, update, insert, delete and mine tuples from any site without accessing the positive Database itself.

In this paper, we introduce an approach representing data mining techniques applied on this secure data format called Negative Data Base (NDB).
Our goal is to devise data Mining Model that prevent inappropriate queries and inferences, while supporting legal Data mining algorithms over the negative format of the data.

There are several motivating scenarios for this work. Consider, a credit card database needed to be mined without transforming these data to hacked format. This database should be in the other hand available to be accessed that in a normal way.
Under this scenario, it is desirable that the database supports only the legal queries while protecting the privacy of individual records.
This paper is organized as followed; the next section (section 2) describes the methodology of negative database showing that there are different algorithms in NDB schema focusing on the prefix algorithm as simple and reversible algorithm, next section (section 3) includes the normal process that may be applied on positive database and how it can be applied on negative database such as select, insert, delete, join…..etc., then jumping to the problem by mentioning little bit description about data mining algorithms and techniques in section(4), and finally in section 5, I illustrated the proposed work with full description of the proposed schema followed by a numeric example of the problem.

2. Negative database

Negative databases are presented as a specific example of representing data negatively. In this approach, the negative image of a set of data records is represented rather than the records themselves (Table 1).

Initially, we assume a universe U of finite-length records (or strings), all of the same length l and defined over a binary alphabet (0,1). We logically divide the

space of possible strings into two disjoint sets: DB representing set of records that holds the information of itself, and (U– DB) denoting the set of all strings not in DB.

We allow (U−DB) to be stored in a compressed form called NDB. We refer to DB as the positive database and NDB as the negative database [7].

Consider, for example, students database contains the following attributes "Name, Subject name, Grade# with a total length of 500 bits, where the first field stores the name of a students, the second has the subject name of the students transaction like “Math” or “physics” and the third stores the grade the transaction took place. The positive database will contain the actual names of the student’s, their transactions, while the negative database will instead have all possible 500 bit combinations except the ones corresponding to the student’s actual data. From a logical point of view, either database will suffice to answer questions regarding DB. However, the different representations present different advantages. For instance, in a positive database, inspection of a single record provides meaningful information. However, inspections of a single (negative) record reflect little about the contents of the original database.

Because the positive tuples are never stored explicitly, negative representation would be much more difficult to misuse. Similarly, depending on the specific representation of NDB, the efficiency of certain kinds of queries may be significantly different than the efficiency of the same query under DB. Some applications may benefit from this change of perspective. Most applications seek to retrieve information about DB as efficiently and accurately as possible, and they typically are not explicitly concerned with U−DB. Yet, in situations where privacy is a concern it may be useful to adopt a scheme in which certain queries are efficient and others are provably inefficient.

2.1. Creating Negative Database NDB

There are several algorithms for creating a negative database (NDB) given as input DB [6], the main distinction between them is the size of the resulting NDB and the ease with which DB can be retrieved from NDB. The prefix algorithm introduced here (Figure 1) is deterministic and reversible, which has consequences for the kinds of inferences that can be made efficiently from NDB. We would like some inferences to be hard (e.g., inferring the original DB from NDB) and other inferences to be easy, depending on the application (e.g., finding certain kinds of correlations in DB) [5, 2].

Table 1. 4 bit length example of a positive database (DB), all the strings not in DB (U-DB), and corresponding negative database (NDB) defined over {1,0,*}.

Positive Database (DB)	U-DB	NDB
0000	1111	1*10
0011	1001	10*1
0101	1101	**10
1100	0001	100*
Positive data base (real data base)	1110	01*0
	1010	*001
	1000	1*01
	1011	Negative Data Base
	0110	
	0010	
	0100	
	0111 Complement Of real data	

Let wi denote an i-bit prefix and Wi a set of i-length bit patterns.

1. *i ← 0*
2. *Set Wi to the empty set*
3. *Set Wi+1 to every pattern not present in B’s wi+1 but with prefix in Wi*
4. *for each pattern Vp in Wi+1{*
5. *Create a record using Vp as its prefix and the remaining positions set to the don’t care symbol.*
6. *Add record to NDB.}*
7. *Increment i by one*
8. *Set Wi to every pattern in DB’s wi*
9. *Return to step 3 as long as i < l.*

Table 2. The complete output of the prefix algorithm.

DB	U-DB	NDB	c-key	RNDB
0001	0000	11**	11**	11**
0100	0010	001*	0*1*	0*1*
1000	0011	011*	*11*	1110
1011	0101	0000	00*0	*111
	0110	0101	*1*1	00*0
	0111	1001	1*01	1*1*
	1001	1010	**10	0101
	1100			1*01
	1101			**10
	1110			*010
	1111			

As cleared from (Table 2) Column # 1 is an example of DB, column #2 gives the corresponding U − DB, column #3 gives corresponding NDB generated by the prefix algorithm, Column #4 presents some possible c-keys extracted from NDB, column #5 gives an example output of RNDB. C-key column is bit pattern not present in DB with no external bits, i.e. a c-key defines a minimal pattern in U − DB in the

sense that the removal of any bit yields a pattern in DB. A c-key is the bitwise complement of a c-key.

3. Database Functions Applied on Negative Database

In order to use negative database as a reliable schema for normal database it should support group of functions which are necessary in any database.

In this section we will describe how database supports these functions like modifying data, such as insert and delete and extends that work by defining a set of relational operators for negative representations. For each relational operator, the corresponding negative operator is defined such that the result of the negative operator applied to a negative representation is equivalent to the positive version applied to the positive representation. Algorithms for each relational Operator are described and compared to its positive counterpart [3].

This work enhances the practicality of negative databases and expands their range of application. Suppose there is a database attribute contains only these two values (000 & 101), table 3- shows that the corresponding U-DB and the corresponding NDB

Table 3. data base file contains only two values, and corresponding database and negative database.

DB	(U − DB)	NDB
000	111	01*
101	110	11*
	001	0*1
	011	1*0
	110	
	100	

First, consider following definition:-

- x, y, z: strings.
- x [i, ..., j]: string x projected onto positions i, ..., j.
- Un: the universe of all possible binary strings of length n.
- DB1 and DB2: subsets of Un and Um respectively, referred to as positive databases.
- NDB1 and NDB2: negative databases representing Un − DB1 and Um − DB2.

3.1. Negative Select

The Select operation over a positive database restricts the relation according to some criterion in the form of a predicate.

Select is defined in terms of a relation between two attributes (here understood as the values at some string positions), or an attribute and a constant v. We limit our description to the latter case. Thus, the Select operation applied to DB is defined as:

where 'ϒ' is an ordered list of string positions.

The complement of this set is written as:

Match x M y: Two strings, x and y are matched when

$$\text{iff } \forall i \ ((x[i] = y[i]) \ \| \ (x[i] = *) \ \| \ (y[i] = *)).$$

Let NDB1 = U − DB, and let NDB2 contain all the strings in U that do not satisfy the criterion, i.e.

$$\{x : x \in \ U \wedge x[\Upsilon] \ \bar{\theta} \ v\}.$$

Negative Select is defined as:-

$$\bar{\sigma}_{\Upsilon \bar{\theta} v}(NDB) = \{x : x \in NDB_1\} \bar{\cap} \{x : x \in NDB_2\}$$

The above formula shows that Negative Select can be viewed as the Negative Intersection of two databases.

3.2. Delete, Insert and Update

The three major functions delete, insert and update is the most common functions performed by database management system.

The process of delete from database is just adding the complement of the item to NDB. And vice versa, inserting an item to the original database is removing the corresponding records from the NDB. As cleared from the next table (table 4), the more databases increased in size, the more NDB databases increased in size, the more NDB decreased in the size that is because U=DB+NDB.

Table 4. The deletion & insertion process in NDB.

NDB	NDB After Insert (0000)	NDB After Delete (1001)
010*	010*	010*
0*00	00*1	00*1
000*	01*1	01*1
001*	001*	001*
00*1		*001
01*1		010*

3.3. Negative Union

For the purpose of union the length of the two strigs length are equal (m=n).

Definition: Coalesce x & y: Two strings x and y of length n collected into string z iff x matches y and for all I get values from 1 to n:

$$z[i] = \begin{cases} x[i], if \ (x[i] = y[i]) \ \vee \ (y[i] = *) \\ y[i], if \ x[i] = * \end{cases}$$

The union of two databases can be expressed as:

$$DB \cup DB2 = \{x: x \in DB1 \vee x \in DB2\}$$

The complement of the previous is written as:

$$U - (DB1 \cup DB2) = \{x: x \notin DB1 \wedge x \notin DB2\}$$

Then, the Negative Union, $\tilde{\cup}$ (U) is defined as:

$$NDB1 \sim\cup NDB2 = \{z:z=x \odot y,\ x\ M\ y,$$

$$(x \in NDB1 \wedge y \in NDB2)\}$$

3.4. Negative Join

Negative-Join of NDB1 and NDB2 considered as a secured alternative for normal join (see Table 8).

The algorithm is presented below. It is a simplification of the Negative-Join algorithm introduced in [3, 4, and 6]; here every position in NDB1 is used as part of the Join condition and all such positions are contiguous in NDB2. Negative-Join

Table 5: The Join of DB1 and DB2 and the equivalent operation on their negative databases (only a subset of each database is shown).

DB1	DB2	DB3=DB1 DB2
111111	010111–00010	010111–00010
010111	110011–00100	111111–00101
....	111111–00101	
....	....	....

NDB1	NDB2	NDB3=NDB1 ⋈ N DB2
1**01*	1***0*–0**1*	1**01*–*****
1**10*	*11**1–1*1**	1**10*–*****
	00*1**–*1*11	1***0*–0**1*
....	01*0**–*011*	*11**1–1*1**
		00*1**–*1*11
	....	01*0**–*011*
		....

3.5. Complexity Of Negative Database

As cleared from the following the two tables Table 6 and Table 7 the first one apply the normal database operations on positive database environment, while the second table apply the same functions but on a negative environment to ensure the security required specially in transforming data from site to another .

The third (Table 8) held comparison between positive and negative database operations.

Table 6. Two positive databases with 3-bit strings and the result of applying the indicated operations (select, Cartesian product, join, intersection, union, project, and set difference). Note the join condition for 1 is γ 1 = {2, 3}, γ 2 = {1, 2}.

DB1	DB 2	$\sigma\gamma$ =101(DB1)	×	⋈	∩	γ	$\pi\gamma$ 1,2(DB 2)	DB1-DB2
001	001	101	001001	0010	001	001	00	101
101	010		001010	1010		010	01	
			101001			101		
			101010					

Table 7. The results of relational operators on negative databases, NDB1 and NDB2. Corresponding results complement those from Table 6.

NDB1	NDB2	$\overline{\sigma}\gamma$ =101(DB1)	N(×)	N ⋈	N∩	N(∪)	$\pi\gamma$ 1,2(DB2)	DB1-DB2
01*	000	01*	***10*	01**	01*	001	10	01*
*00	011	*00	***11*	*00*	*00	000	11	*00
11*	10*	11*	***000	11**	11*	100		11*
	11*	0**	11****	*000	000	11*		001
		1	*00***	*011	011			
			***011	*10*	10*			
			01****	*11*	11*			

Table 8. Comparison of relational operators' complexity between positive DB and negative DB

Operation	Select	Join& CP	Subtract	Union
Positive DB	O(m+n)(|DB1||DB2|)	O(n|DB1||DB2|)	O(n|DB1| |DB2|)	O(n(|DB1| + |DB2|))
Negative DB	O((m + n)(|NDB1| + |NDB2|))	O((m+n) (|NDB1| + |NDB2|))	O(n(|NDB1 + |NDB2|))	O(n|NDB1||NDB2|)

4. Data Mining

The extraction of useful and non-trivial information from the huge amount of data that is possible to collect in many and diverse fields of science, business and engineering, is called Data Mining (DM). DM is part of a bigger framework, referred to as Knowledge Discovery in Databases (KDD) that covers a complex process from data preparation to knowledge modeling. Within this process, DM techniques and algorithms are the actual tools that analysts have at their disposal to find unknown patterns and correlation in the data. Typical DM tasks are classification (assign each record of a database to one of a predefined set of classes), clustering (find groups of records that are lose according to some user defined metrics) or association rules (determine implication rules for a subset of record attributes). A considerable number of algorithms have been developed to perform these and others tasks, from many fields of science, from machine learning to statistics through neural and fuzzy computing. What was a hand tailored set of case specific recipes, about ten years ago, is now recognized as a proper science [9].

5. Proposed work

The need for mining private data bases in a secure manner is achieved by using negative database methodology. In our applied case; client's data base contains five attributes, the first column is client number consists of 9 alpha numeric digits and the second attribute is (Automatic Teller Machine) ATM code number with length of 5 digits. The third attribute refers to the amount of money deposited or withdrawn from the ATM node with length of 6 digits the next attribute is the time stamp the action toke place in – date and time- with length of 14 digits and the last one

is the type of the process done the ATM machines compressed in one character the action may be check account, print balance account, withdrew or deposit. The total length of the record is 35 byte with 280 bit see Table 9.

Table9. Sample data describing action record contents.

Client #	ATM #	Amount	Time/Date	Action
121 333 454	65432	98750	2009-12-02	W
345 434 564	65420	32900	2009-12-02	D
…	…	…	…	…

These data should be kept in high level of security manipulation even in this high level decision support system and we should perform this task without the need of transforming the Negative form of the data to positive or real data base.

The recent process now is to find the relation between two attributes Client # and ATM #, in other word, test the most ATMs nodes used by which Clients. This action is called association rule mining.

For now and coming we will act only with the negative database; that is because actually we transformed all our database functions like Insert, Delete , Update, Search, Join and Union as cleared above to work only negative database

5.1. Proposed Algorithm

1. *Assign NDB2 = π Υ 1:114(NDB) Table 10*
2. *Create a new Matrix called MNDB with N number of columns equal to Count number of accessed ATM & C Number of records equal to total number of clients.*
3. *Fill Clients ID in the first column of the matrix MNDB.*
4. *for (I=0 ; <= n ; i++)*
5. *for(j =0 ; j <=C ; j++)*
   If (MNDB [I, j] || Client#[j]) Not Found in (NDB2)
   Then MNDB [I, j] =T
   Else MNDB [I, j] = F.

Table 10. NDB2 Result of 1 of the algorithm (projection from the record starting from bit no 1 till bit no. 112 representing client number and ATM code).

Client number || ATM number
*1***1**1*1***
*1***1**1**0**
*1***1**1****0
*1***1**1*****
*0***11*1*****
*0***1**1****1
*0***1**1*****
……

By finishing the previous Algorithm the data will be at the form of Boolean data true or false for each cell as here first column represent client number and first row represent ATM codes, the intersection between column and row is true or false clearing that whether this client accesses this ATM or not see Table 11.

Table 11. MNDB, Final form of the preprocessed data ready to mining.

Client#	65432	65432	65432	65432	65432	….
121333454	T	F	T	T	F	….
345434564	T	T	F	F	T	….
122333454	T	F	T	T	T	….
346434564	F	T	F	T	F	….
123333454	T	T	T	F	T	….
347434564	T	T	T	T	F	….

Till this point we just extracted some data from the negative database as a preprocessed stage preparing it to data mining. The rest of steps after that is continued as normal.

Using Generalized Rule Induction (GRI) node discovers association rules in the data. Association rules are statements in the form:

If antecedent(s) then consequent(s)

GRI extracts a set of rules from the data, pulling out the rules with the highest information content. Information content is measured using an index that takes both the generality (support) and accuracy (confidence) of rules into account [8].

The next steps applying GRI model to find association rules between the two attributes client # and ATM code. In the way of achieving this goal we will use SPSS Clementine data miner which is one of the most widely used data miners applications in the field of data mining

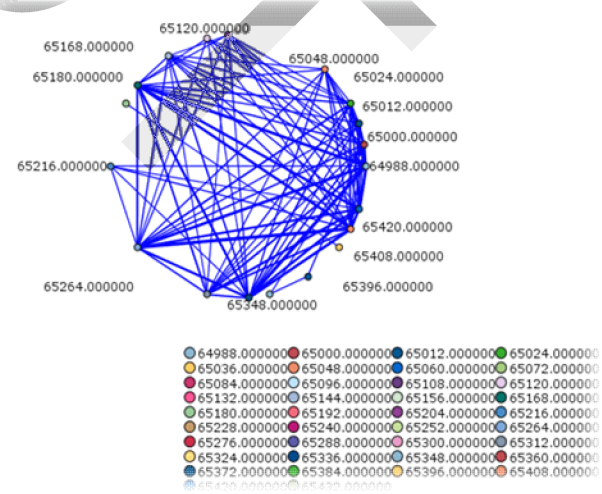


Figure 1.Web graph showing the most ATM nodes used with each other.

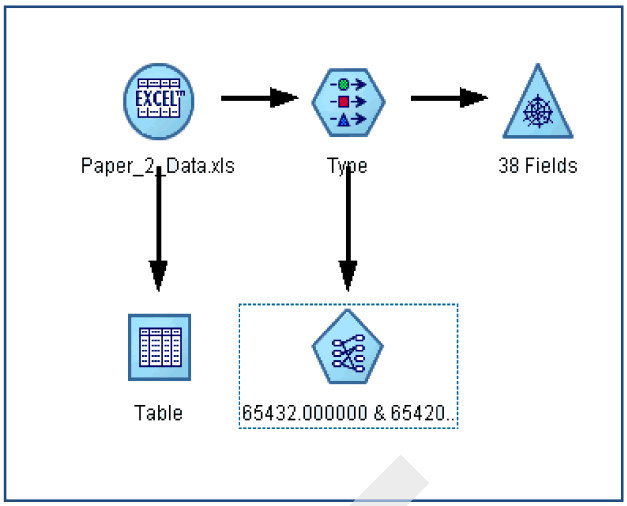


Figure 2. GRI Model applied using SPSS clementine Miner.

Table 11. The result of the previous model is tabulated below.

Consequent	Antecedent	Instances	Support%	Confidence %
65420	65408 64988	229	31	55
65420	64988	334	46	53
65420	65408	298	41	48

Table 11 describes that if we consider ATM # 65420 as output then with confidence 30% and support 30 % it will appear in 229 instances both 65408 and 64988 and so on.

As a result of the previous model it is easy to apply Association rule on any attributes in the database and it is easy to apply the other methodologies of data mining such as clustering and classification.

6. Conclusion and future work

Data mining is a powerful tool to extract information from large scale of data under the rule that "There is gold under the mountain of data" but we still in this stage suffering from hacking state and lack of privacy, so converting data to Negative Format give golden solution to this crucial problem.

The more real data increase in size, the less data in Negative database created, which make the size of NDB reasonable and time to create it is competitive somehow.

In this paper we have shown the feasibility of a new approach to representing information. Specifically, we have shown that negative representations are computationally feasible to be mined. However, there are many important questions and issues remaining. Which technique of mining can be computed efficiently and which cannot? Our initial results address very important case of, the ability to apply data mining techniques on the negative form of data, in a private manner of data keeping the confidentially state of the data.

As a future work, it is highly recommended that the data marts and data warehouses should have the ability to deal with negative form of data and as well the ability to perform multi database mining techniques on distributed database, the authors have their research in the points.

References

[1] Esponda, F., and Forrest, S., Enhancing privacy through negative representations of data. *University of New Mexico, Technical report* (2004).

[2] Esponda, F., and Forrest, S. *Negative representations of information, published online*: 2009 Springer.

[3] Esponda, F. Everything That is Not Important: Negative Databases MAY IEEE Computational Intelligence Magazine 2008.

[4] Esponda, F. Negative Representations of Information, Ph.D. thesis, University of New Mexico, 2005.

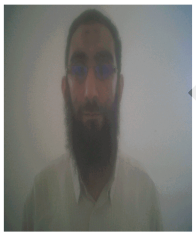
[5] Esponda, F., and Ackley, E., Protecting data privacy through hard-to-reverse negative databases Published online Springer: 24 July 2007.

[6] Esponda F., Ackley, E., Forrest, S., and Helman, P.: On-line negative databases. In: Proceedings of ICARIS (2004)

[7] Esponda, F., Trias, S., Ackley, and Forrest, S., A relational algebra for negative databases, Technical report TR-CS-2007-18, University of New Mexico, 2007.

[8] Max, B. Principles of Data mining, Springer, 2007.

[9] SPSS Clementine Technical manual 2007.



Mohamed Ibrahim obtained his B.S. and M.SC both in Information systems from Sadat Academy from Management Sciences, Egypt in 1993 and 2006 respectively. He obtained his Ph.D. degree in Information Systems also from Menofia University, Faculty of Computers and informatics, Egypt in 2012 specializing in Negative Databases. He is currently a Lecturer in High institute for computer sciences and information systems. He has worked on a number of research topics and consulted for a number of organizations. He has contributed more than 5+ technical papers in the areas of Negative Databases and data mining, Database applications, Information security and Internet applications

Hatem Abdul-kader obtained his B.S. and M.SC. (by research) both in Electrical Engineering from the Alexandria University , Faculty of Engineering , Egypt in 1990 and 1995 respectively. He obtained his Ph.D. degree in Electrical Engineering also from Alexandria University, Faculty of Engineering, Egypt in 2001 specializing in neural networks and applications. He is currently an Associate professor in Information systems department, Faculty of Computers and Information, Information systems department, Faculty of Computers and Information, Menoufiya University, Egypt since 2004. He has worked on a number of research topics and consulted for a number of organizations. He has contributed more than 30+ technical papers in the areas of neural networks, Database applications, Information security and Internet applications.

Mohiy Hadhoud has a PhD from college of information technology, university of Kent, UK. His interest in Image processing, neural network, AI, Machine learning, Algorithm, He is working as vise president in Menofia university, Egypt.

Ali Abou-Zaid has a PhD from college of information technology, university of Kent, canter bary, UK. His interest is neural network, AI, Machine learning, Algorithm, He is working as dean of HIC, Egypt.