Mining Approximative Descriptions of Sets Using Rough Sets

Dan A. Simovici and Selim Mimaroglu
Univ. of Massachusetts Boston,
Dept. of Comp. Science,
100 Morrissey Blvd.
Boston, Massachusetts
02125 USA
{dsim,smimarog}@cs.umb.edu

Abstract

Using concepts from rough set theory we investigate the existence of approximative descriptions of collections of objects that can be extracted from in data set, a problem of interest for biologists that need to find succinct descriptions of families of taxonomic units. Our algorithm is based on an anti-monotonicity of borders of object set and makes use of an approach that is, in a certain sense, a dual of the Apriori algorithm used in identifying frequent item sets.

1 Introduction

Rough sets are approximative descriptions of sets that can be achieved using equivalences (or partitions). This fertile idea was introduced by the Polish mathematician Z. Pawlak in [6]. Excellent surveys supplemented by large bibliographies are [3] and [1]. The reference [4] contains a rich collection of applications of rough sets.

The problem that we address here is mining a table (or, in the terminology of rough sets, an information system) for approximative descriptions of sets of objects. Finding such descriptions is relevant for biological applications, where such descriptions serve for specimen identification in the field work.

We begin by introducing some terminology and basic facts from rough set theory. Unless stated otherwise, all sets are finite. If U is a subset of a set S, we denote its complement $S - U$ by U^c.

Let S be a set. An *approximation space* is a pair (S, ρ), where ρ is an equivalence relation defined on the set S.

DEFINITION 1.1. Let (S, ρ) be an approximation space and let U be a subset of S. The *ρ-lower approximation of U* is the union of all ρ-equivalence classes included in the set U:

$$\mathsf{lap}_\rho(U) = \bigcup\{[x]_\rho \in S/\rho \mid [x]_\rho \subseteq U\}.$$

The *ρ-upper approximation of U* is the union of ρ-equivalence classes that have a nonempty intersection with the set U:

$$\mathsf{uap}_\rho(U) = \bigcup\{[x]_\rho \in S/\rho \mid [x]_\rho \cap U \neq \emptyset\}.$$

$\square$

The following statements hold in an approximation space (S, ρ) (see, for example [7]):

$$\begin{aligned}
\mathsf{lap}_\rho(\emptyset) &= \mathsf{uap}_\rho(\emptyset) = \emptyset && (1.1)\\
\mathsf{lap}_\rho(S) &= \mathsf{uap}_\rho(S) = S, && (1.2)\\
\mathsf{lap}_\rho(U \cap V) &= \mathsf{lap}_\rho(U) \cap \mathsf{lap}_\rho(V), && (1.3)\\
\mathsf{uap}_\rho(U \cup V) &= \mathsf{uap}_\rho(U) \cup \mathsf{uap}_\rho(V), && (1.4)\\
\mathsf{lap}_\rho(U \cup V) &\supseteq \mathsf{lap}_\rho(U) \cup \mathsf{lap}_\rho(V), && (1.5)\\
\mathsf{uap}_\rho(U \cap V) &\subseteq \mathsf{uap}_\rho(U) \cap \mathsf{uap}_\rho(V), && (1.6)\\
\mathsf{lap}_\rho(U^c) &= \left(\mathsf{uap}_\rho(U)\right)^c && (1.7)\\
\mathsf{uap}_\rho(U^c) &= \left(\mathsf{lap}_\rho(U)\right)^c, && (1.8)\\
\mathsf{lap}_\rho(\mathsf{lap}_\rho(U)) &= \mathsf{uap}_\rho(\mathsf{lap}_\rho(U))\\
&= \mathsf{lap}_\rho(U) && (1.9)\\
\mathsf{uap}_\rho(\mathsf{uap}_\rho(U)) &= \mathsf{lap}_\rho(\mathsf{uap}_\rho(U))\\
&= \mathsf{uap}_\rho(U), && (1.10)
\end{aligned}$$

for every $U, V \in \mathcal{P}(S)$.

Note that, in general, $\mathsf{lap}_\rho(U) \subseteq \mathsf{uap}_\rho(U)$ for any set U. Also, we have

$$\begin{aligned}
\mathsf{uap}_\rho(U) &= \{t \in S \mid (t, s) \in \rho \text{ for some } s \in U\},\\
\mathsf{lap}_\rho(U) &= \{t \in U \mid (t, s) \in \rho \text{ implies } s \in U\}.
\end{aligned}$$

A subset U of S is *ρ-rough* if $\partial_\rho(U) \neq \emptyset$ and is *ρ-crisp* otherwise.

In defining and retrieving approximative descriptions of sets the notion of border plays a fundamental role.

DEFINITION 1.2. The *positive ρ-border* of U is the set

$$\partial^+_\rho(U) = U - \mathsf{lap}_\rho(U),$$

while the *negative ρ-border* of the same set is

$$\partial^-_\rho(U) = U - \mathsf{lap}_\rho(U),$$

The ρ-border of U is:

$$\partial_\rho(U) = \partial^+_\rho(U) \cup \partial^-_\rho(U) = \mathsf{uap}_\rho(U) - \mathsf{lap}_\rho(U).$$

□

Observe that

$$\begin{aligned}\partial_\rho(U) &= \{t \in S \mid (t,s) \in \rho \text{ and } (t,z) \in \rho \\ &\qquad \text{for some } s \in U \text{ and } z \notin U\}.\end{aligned}$$

For every subset U of an approximation space (S, ρ) we have the equalities:

$$\begin{aligned}\partial^+_\rho(U^c) &= \partial^-_\rho(U), &(1.11)\\ \partial^-_\rho(U^c) &= \partial^+_\rho(U). &(1.12)\end{aligned}$$

Indeed, we can write

$$\begin{aligned}\partial^+_\rho(U^c) &= U^c - \mathsf{lap}_\rho(U^c)\\ &= U^c - \left(\mathsf{uap}_\rho(U)\right)^c\\ &\qquad \text{(by Equality 1.7)}\\ &= U^c \cap \mathsf{uap}_\rho(U)\\ &= \mathsf{uap}_\rho(U) - U\\ &= \partial^-_\rho(U),\end{aligned}$$

which is Equality 1.11. A similar argument works for Equality 1.12.

Let (S, ρ) be an approximation space and let U and V be two subsets of S. If $U \subseteq V$, then $\mathsf{lap}_\rho(U) \subseteq \mathsf{lap}_\rho(V)$ and $\mathsf{uap}_\rho(U) \subseteq \mathsf{uap}_\rho(V)$.

Let ρ, σ be two equivalences on the set S. If $\rho \subseteq \sigma$, then each σ-equivalence class is a ρ-saturated set. Therefore, if $U \subseteq S$ we have

$$\mathsf{lap}_\sigma(U) \subseteq \mathsf{lap}_\rho(U) \subseteq U \subseteq \mathsf{uap}_\rho(U) \subseteq \mathsf{lap}_\sigma(U). \quad (1.13)$$

These inclusions imply

$$\partial_\rho(U) \subseteq \partial_\sigma(U). \quad (1.14)$$

Therefore, we have

$$\partial_{\rho_1 \wedge \rho_2}(U) \subseteq \partial_{\rho_1}(U) \cap \partial_{\rho_2}(U). \quad (1.15)$$

for every subset U of S.

2 Exact descriptions of Sets of Objects

Let H be a finite set, $H = \{A_1, \ldots, A_m\}$. We refer to the elements of H as *attributes* and we assume that for each attribute A_i we have a set that contains at least two elements referred to as the *domain of* A_i and denoted by $\mathrm{Dom}(A_i)$.

A *data set on the set of attributes* H is a function $T : \{1, \ldots, n\} \times H \longrightarrow \bigcup_{j=1}^m \mathrm{Dom}(A_j)$ such that $T(i, A_j) \in \mathrm{Dom}(A_j)$ for $1 \leq i \leq n$ and $1 \leq j \leq m$.

The sequence $t_k = (T(k,1), \ldots, T(k,m))$ is the k^{th} *object of* T for $1 \leq k \leq n$. The numbers $1, \ldots, n$ are the object identifiers (abbreviated as *oids*). The set of objects of T is the set $\mathcal{O}_T = \{t_1, \ldots, t_n\}$.

If $L = \{A_{i_1}, \ldots, A_{i_p}\}$ be a subset of H. The projection of the object $t_k = (T(k,1), \ldots, T(k,m))$ on L is the p-tuple $(T(k,i_1), \ldots, T(k,i_p))$, denoted by $t_k[L]$.

DEFINITION 2.1. Let T be a data set and let L be a set of attributes of T. The equivalence ρ_L on $\mathcal{O}_T$ defined by

$$\rho_L = \{(t,t') \in \mathcal{O}_T^2 \mid t[L] = t'[L]\}.$$

□

It is easy to see that if L and K are two attribute sets then $\rho_{KL} = \rho_K \cap \rho_L$, where KL is an alternative notation for the union $K \cup L$ of the two attribute sets. Therefore, if $L \subseteq K$, then $\rho_K \subseteq \rho_L$, so by the inclusion (1.14) we have:

$$\partial_{\rho_L}(U) \subseteq \partial_{\rho_K}(U). \quad (2.16)$$

In other words, the border of a set of objects relative to an attribute set is anti-monotonic with respect to the attribute set. To simplify notations we denote the set $\partial_{\rho_K}(U)$ by $\partial_K(U)$.

DEFINITION 2.2. A set of objects $\mathcal{D}$ is *described by a set of attributes* K if $\partial_K(\mathcal{D}) = \emptyset$ and we refer to K as an *exact description* of $\mathcal{D}$. □

The notion of *template* is defined as a formula $\tau = (A_{i_1} = a_{i_1}) \wedge (A_{i_2} = a_{i_2}) \wedge \cdots \wedge (A_{i_p} = a_{i_p})$, where $h \neq k$ implies $A_{i_h} \neq A_{i_k}$ for $1 \leq h, k \leq p$ (see, for example [5]). The *length* of the template is p and the *support* of the template τ in the data set T is

$$\mathsf{supp}_T(\tau) = |\{t \in \mathcal{O}_T \mid t[A_{i_1} \cdots A_{i_p}] = (a_{i_1}, \ldots, a_{i_p})\}|.$$

Note that each template $\tau = (A_{i_1} = a_{i_1}) \wedge (A_{i_2} = a_{i_2}) \wedge \cdots \wedge (A_{i_p} = a_{i_p})$ that has a non-zero support corresponds to an equivalence class of ρ_K, where $K = \{A_{i_1} \cdots A_{i_p}\}$.

It is shown in [5], starting from the problem of the balanced complete bipartite subgraph [2], that for a bipartite graph $G = (V_1 \cup V_2, E)$ and two positive integers

Table 1. Data Set T and the set of objects $\mathcal{D} = \{t_5, t_6, t_7, t_8, t_9\}$

	T			
t_1	a_1	b_2	c_1	d_1
t_2	a_2	b_2	c_1	d_2
t_3	a_3	b_1	c_2	d_1
t_4	a_4	b_1	c_2	d_3
t_5	a_1	b_1	c_1	d_2
t_6	a_3	b_1	c_1	d_2
t_7	a_5	b_3	c_3	d_4
t_8	a_1	b_3	c_3	d_2
t_9	a_2	b_3	c_2	d_3
t_{10}	a_3	b_3	c_2	d_3
t_{11}	a_4	b_2	c_2	d_1
t_{12}	a_1	b_3	c_4	d_4

$k_1 \leq |V_1|$ and $k_2 \leq |V_2|$, it is an NP-complete problem to determine the existence of two subsets U_1 and U_2 of V_1 and V_2, respectively such that $|U_1| = k_1$, $|U_2| \geq k_2$ and $\{u_1, u_2\} \in E$ for any $u_1 \in U_1$ and $u_2 \in U_2$. This variant of the problem is known as the Complete Bipartite Subgraph (CBS) problem. Then, given a table and two positive integers k and ℓ, the existence of a template τ containing ℓ conjunctions and having support at least k is polynomially equivalent to the CBS problem and therefore, is NP-complete.

We focus in this paper on approximate descriptions of sets of objects which can be stated as follows: given a data set T, a set of objects $\mathcal{D} \subseteq \mathcal{O}_T$, we seek to determine whether there exists an attribute set K containing no more than k attributes such that the size of border $\partial_K(D)$ is less than p. Thus, a search for a template of minimum support is replaced by the search for a set of attributes of limited size that describes the set $\mathcal{D}$ with a certain precision.

EXAMPLE 2.3. Let T be a data set having the set of attributes $H = ABCD$. Our search space is the set of subsets of H. Suppose that we seek to identify a description of the set $\mathcal{D} = \{t_5, t_6, t_7, t_8, t_9\}$, where T is shown in Table 1. Observe that the equivalence classes of the equivalence ρ_{BC} are $\{t_1, t_2\}, \{t_3, t_4\}, \{t_5, t_6\}, \{t_7, t_8\}, \{t_9, t_{10}\}$, $\{t_{11}\}$, and $\{t_{12}\}$. Two of these classes, $\{t_5, t_6\}, \{t_7, t_8\}$ are included in $\mathcal{D}$ and therefore, they constitute the lower approximation of $\mathcal{D}$. The class $\{t_9, t_{10}\}$ intersects both $\mathcal{D}$ and its complement and therefore, $\partial_{BC}(\mathcal{D}) = \{t_9, t_{10}\}$. Also, it is clear that $\partial^+_{BC}(\mathcal{D}) = \{t_9\}$ and $\partial^-_{BC}(\mathcal{D}) = \{t_{10}\}$. □

3 An Algorithm for Mining for Approximate Descriptions

In practice, approximative descriptions are sufficient and we give an algorithm that identifies such descriptions. This algorithm is, in a certain sense, a dual of the well-known Apriori-algorithm for finding frequent item sets.

DEFINITION 3.1. Let ϵ be a number such that $0 \leq \epsilon \leq 1$. A set of objects $\mathcal{D}$ is *ϵ-described by a set of attributes K* if

$$\frac{|\partial_K(\mathcal{D})|}{|\mathcal{D}|} \leq \epsilon.$$

□

The "dual" of the Rymon tree used in the study of frequent item sets is introduced next.

DEFINITION 3.2. Let $H = \{A_1, \ldots, A_n\}$ be a set of attributes. The *dual Rymon tree* of H is a rooted tree having $\mathcal{P}(H)$ as its set of nodes, H as its root, and whose set of edges consist of pairs of the form $(U, V) \in \mathcal{P}(H)$ such that V is obtained by dropping from U an attribute that follows the attributes of $H - U$ in the list $(A_1, \ldots, A_n)$. □

EXAMPLE 3.3. The dual Rymon tree of the set of attributes $H = \{A, B, C, D\}$ is shown in Figure 1. □

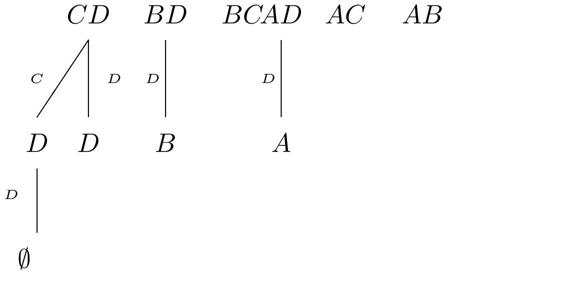

Figure 1. Rymon tree of the set of attributes $H = \{A, B, C, D\}$

Our algorithm makes use of the dual Rymon tree of the sets of attributes of the data set and is based on the anti-monotonicity of the size of the border of a set given by the inclusion (2.16). The goal of the algorithm is to compute the smallest sets of attributes that yield a description of a set of objects whose error is below a presecribed threshold.

For identifying approximative descriptions of $\mathcal{D}$ we search the Rymon tree of the set of attributes in a top-down manner. During the search we apply a pruning technique that reduces substantially the size of the search space.

Starting at the root node of the tree, computation of negative and positive borders take place in breadth first search fashion. In a database having no duplicates the error of the root node is zero. The error of the children of the root are computed next. If the error of a node K is greater than the error threshold there is no need for computing the negative and positive borders for its descendants because of the anti-monotonicity property of the size of the border contained by Inequality 2.16. Thus, we can prune all descendants of K.

We introduce first a technique for computing the border set of a set of objects $\mathcal{D} = \{t_{i_1}, \ldots, t_{i_p}\}$ of a data set T determined by a set of attributes K. Let $\bar{\mathcal{D}} = \mathcal{O}_T - \mathcal{D} = \{t_{j_1}, \ldots, t_{j_q}\}$ be the set of objects of T that do not belong to D.

The projection of each object t_{i_k} in $\mathcal{D}$ on the set of attributes K is compared to the same projection of each object t_{j_h} in $\bar{\mathcal{D}}$ as shown in the Figure 2.

Pruning is implemented as shown in Figure 3

Failed set of attributes are stored according to their cardinality. As mentioned earlier, each failed set of attributes is represented by a bit vector. Before adding any set of attributes into the working queue for computing the negative and positive borders, we check the failed set of attributes. A set of attributes L is not added to the queue if it has a superset that failed.

4 Experimental Results

Our algorithm is applicable to data sets that have attributes with finite domains of arbitrary cardinality. However, we focus on data sets with binary domains in order to take advantage of the efficient use of memory that bit vectors allow and of the speed of bit operation. If a an attribute A has a non-binary domain, $\mathrm{Dom}(A) = \{a_1, \ldots, a_k\}$, we replace A by k attributes $A^1, \ldots, A^k$ and we replace the A-component of an object t by k components $t[A^1], \ldots, t[A^k]$ defined by

$$t[A^j] = \begin{cases} 1 & \text{if } t[A] = a_j, \\ 0 & \text{otherwise} \end{cases}$$

for $1 \leq j \leq k$. Thus, an object of a binary data set is a sequence $t \in \{0,1\}^n$.

If we define the characteristic n-tuple $r_K = (r_1, \ldots, r_n) \in \{0,1\}^n$ of a set of attributes K as

$$r_i = \begin{cases} 1 & \text{if } A_i \in K, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the condition $t[K] = t'[K]$ can be expressed as $(t \wedge r_K) \oplus (t' \wedge r_K) = \mathbf{0}$, where $\mathbf{0} = (0, \ldots, 0)$. If $t[K]$ equals $t'[K]$, then t is added to the positive border $\partial^+_K(D)$ and t' is added to the negative border $\partial^-_K(D)$. After computing the positive and the negative borders, we obtain the border $\partial_K(D) = \partial^+_K(D) \cup \partial^-_K(D)$.

To evaluate the effectiveness of our algorithm we conducted a set of preliminary experiments on synthetically generated binary databases using a Pentium 3.0GHz computer having 4GB of main memory running on Linux. We implemented our algorithm in Java which offers support for bit vectors and bit vector operations and we use the advantage of the speed of bit operations.

Figure **??** shows the running time results on various size databases. The number of unique descriptors for each error level is shown in Figure **??**.

Experiments show that for even big size database run times for reasonable error rates are practical. This is due to our pruning technique and use of bit vectors.

References

[1] I. Düntsch and G. Gediga. *Rough Sets Analysis - A Road to Non-invasive Knowledge Discovery*. Methoδos, Bangor, UK, 2000.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, 1979.

[3] J. Komorowski, Z. Pawlak, L. Polkowski, and A. Skowron. Rough sets: A tutorial. In S. K. Pal and A. Skowron, editors, *Rough Sets Hybridization: A New Trend in Decision Making*, pages 3–98. Springer-Verlag Telos, 1999.

[4] T. Y. Lin and N. Cercone, editors. *Rough Sets and Data Mining*. Kluwer Academic Publishers, Boston, 1997.

[5] S. H. Nguyen, A. Skowron, and P. Synak. Dicovery of data patterns with applications to decomposition and classification problems. In *Rough Sets in Knowledge Discovery*, volume 2, pages 55–97. Physica-Verlag, Heidelberg, 1998.

[6] Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishing, Dordrecht, 1991.

[7] D. Simovici and C. Djeraba. *Mathematical Tools for Data Mining*. Springer-Verlag, London, UK, 2008.

```
Input: $T$: data set, $\mathcal{D}$: set of objects, $K$: set of
        attributes
Output: Positive and Negative Border of $\mathcal{D}$
1 $Pos := \{\}$ ;
2 $Neg := \{\}$ ;
3 $\bar{\mathcal{D}} := \mathcal{O}_T - \mathcal{D}$ ;
4 foreach $t \in \mathcal{D}$ do
5     foreach $t' \in \bar{\mathcal{D}}$ do
          // project on $K$
6         if $t[K] == t'[K]$ then
7             add $t$ to $Pos$;
8             add $t'$ to $Neg$;
9 output $Pos \bigcup Neg$ ;
```

Figure 2. Computation of Border, $FindBorder(T, \mathcal{D}, K)$

```
Input: $L$: list of failed descriptors, $R$: set of
        attributes
Output: all the qualified $|R| - 1$ size children of $R$
1 enumerate all unique $|R| - 1$ size children of $R$ into
  $P$ ;
2 foreach $p \in P$ do
3     if $L$ contains a superset of $p$ then
4         remove $p$ from $P$ ;
5 output $P$;
```

Figure 3. Prunning of Attribute Sets, $Prunnig(L, R)$

```
Input: $T$: data set, $H$: set of attributes, $\mathcal{D}$: set of
        objects, $err$: error threshold
Output: all the descriptors of $\mathcal{D}$
1 initialize a queue $Q$;
2 initialize a list $L$;
3 add $H$ to $Q$ ;
4 while $Q$ is not empty do
5     $R$ := remove first element from $Q$;
6     if FindBorder($T$,$\mathcal{D}$,$R$) $\leq err$ then
7         output $R$;
8         $children := Prunning(L, R)$;
9         add $children$ to $Q$;
10    else
11        add $R$ to $L$;
```

Figure 4. Find Descriptors of $\mathcal{D}$, $FindAll(T, H, \mathcal{D}, err)$

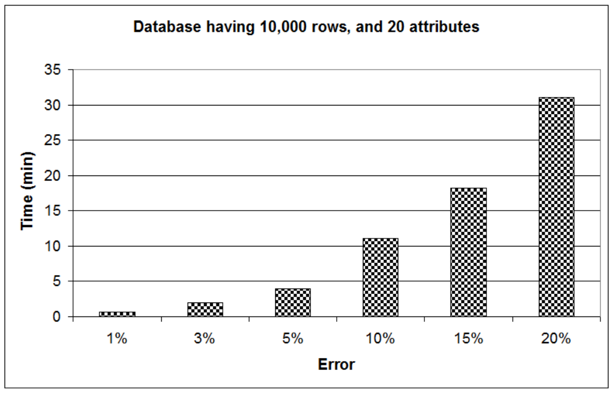

Figure 5. Running Time and Size of Descriptor Sets for 10,000 rows

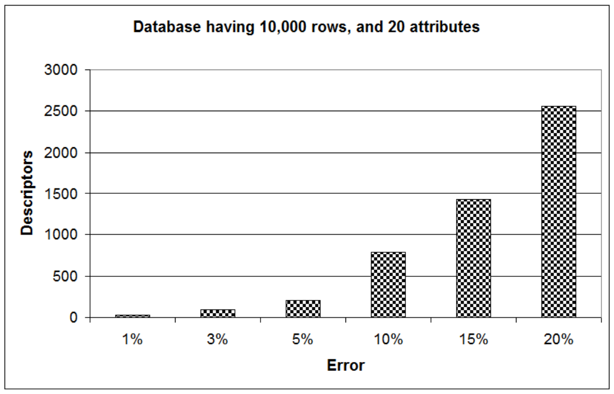

Figure 6. Running Time and Size of Descriptor Sets for 10,000 rows

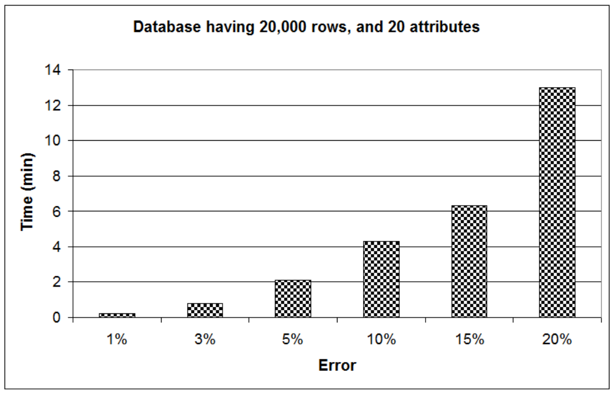

Figure 7. Running Time and Size of Descriptor Sets for 20,000 rows

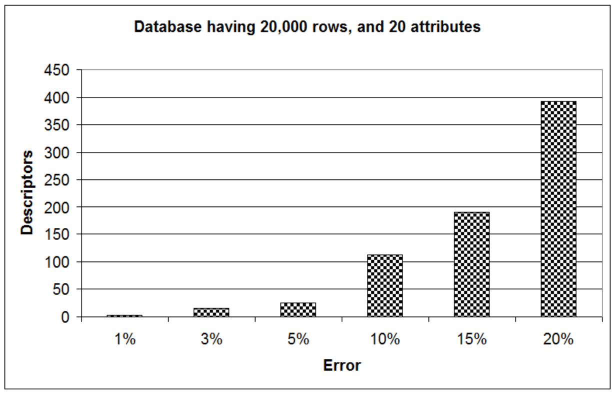

Figure 8. Running Time and Size of Descriptor Sets for 20,000 rows

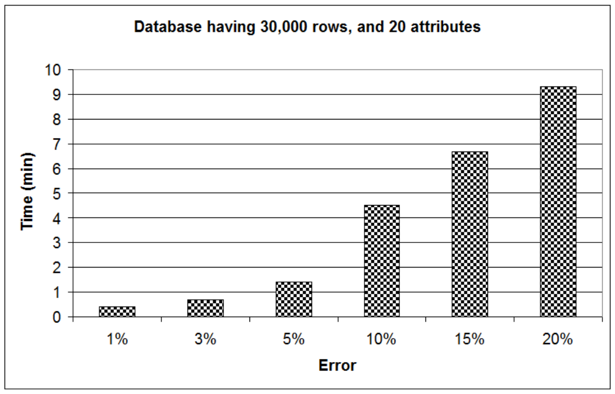

Figure 9. Running Time and Size of Descriptor Sets

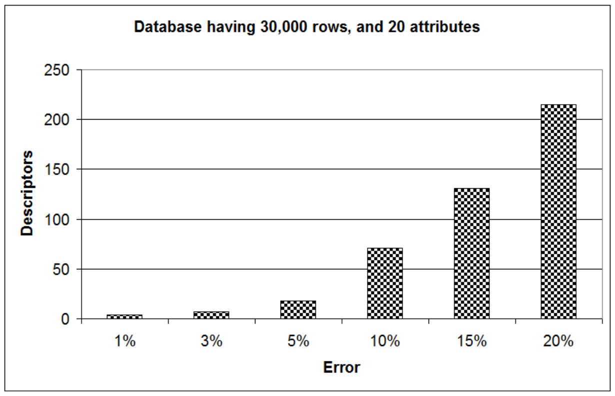

Figure 10. Running Time and Size of Descriptor Sets

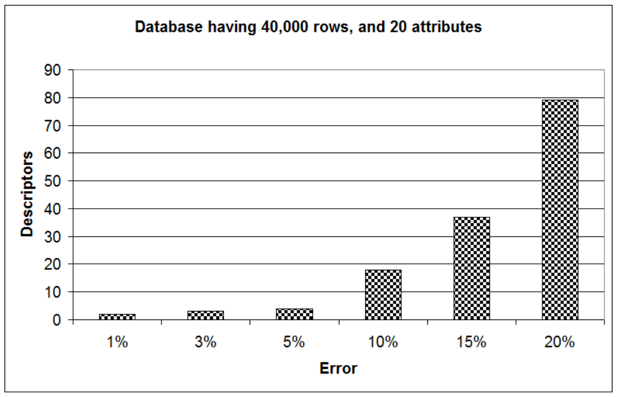

Figure 11. Running Time and Size of Descriptor Sets

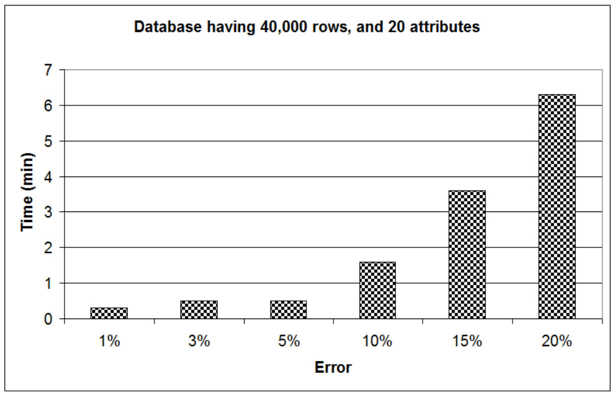

Figure 12. Running Time and Size of Descriptor Sets