

# Uniqueness Mining

Rohit Paravastu, Hanuma Kumar, and Vikram Pudi

IIIT-H,Gachibowli,Hyderabad 500032,India

[prohit@students.iiit.ac.in](mailto:prohit@students.iiit.ac.in),

[hanuma@students.iiit.ac.in](mailto:hanuma@students.iiit.ac.in),

[vikram@iiit.ac.in](mailto:vikram@iiit.ac.in)

<http://www.iiit.net/vikram>

**Abstract.** In this paper we consider the problem of extracting the special properties of any given record in a dataset. We are interested in determining what makes a given record unique or different from the majority of the records in a dataset. In the real world, records typically represent objects or people and it is often worthwhile to know what special properties are present in each object or person, so that we can make the best use of them. This problem has not been considered earlier in the research literature. We approach this problem using ideas from clustering, attribute oriented induction (AOI) and frequent itemset mining. Most of the time consuming work is done in a preprocessing stage and the online computation of the uniqueness of a given record is instantaneous.

## 1 Introduction

In this paper we consider the problem of extracting the special properties of any given record in a dataset. We are interested in determining what makes a given record unique or different from the majority of the records in a dataset. In the real world, records typically represent objects or people and it is often worthwhile to know what special properties are present in each object or person, so that we can make the best use of them. This problem has not been considered earlier in the research literature.

At first glance it may be tempting to dismiss this problem as another avatar of outlier mining. However, in the latter, we are interested in extracting records that are significantly different from the majority; here, we are interested in extracting the properties that make any given record seem like an outlier.

There are several scenarios where this type of mining can be useful:

1. Deans of universities and principals of schools generally send letters to parents of students describing their child's performance. It may be easy to identify students who got the top rank in each subject. However, it will be nice to be able to identify a student who is the only one who got more than (or less than!) 70% in some subjects  $A$ ,  $B$  and  $C$ . If some unique property of each student were mentioned, it would be interesting to the readers of their reports.

2. A student comes to a faculty member in a university asking for a project. The best outcome can be expected if a project is chosen that makes use of the special skills of this student. One way to identify these special skills would be to mine the database of student marks in various subjects, to find unique combinations of subjects in which this student performs well.
3. The hiring wing of a company would be interested to know what are the special skills of applicants. Many online job portals require applicants to enter their individual skills, academic performance, etc. in digital form. The resulting database can be mined to obtain the speciality of each applicant.
4. Special skills of people working in a company can be used to assign them to various projects.
5. Consider an exam paper having 10 questions. Rather than deciding the weightage of marks for each question, grades to students can be given depending on what unique combinations of questions they answered correctly. For example, if everyone answered questions 1,2 and 3 correctly, these questions wouldn't count much in the final grade. If some student was the only one to answer questions 5 and 6, then it indicates that this student deserves a good grade.

We approach the uniqueness mining problem using ideas from clustering, attribute oriented induction (AOI) and frequent itemset mining. Most of the time-consuming work is done in a preprocessing stage and the online computation of the uniqueness of a given record is instantaneous. We experimentally evaluated our approach on a synthetically generated dataset, by adding records that are known to contain very different properties, and showing that our system is able to identify them.

The remainder of the paper is organized as follows: In Section 2 we formulate the uniqueness mining problem. In Section 3 we present our uniqueness mining algorithm. Then, in Section 5 we experimentally evaluate our algorithm and show the results. Related work that seems connected to the problem definition or the algorithm is reviewed in Section 4. Finally, in Section 6, we summarize the conclusions of our study and identify future work.

## 2 The Uniqueness Mining Problem

In this section we formulate the uniqueness mining problem. We provide a general definition of the uniqueness of objects and show how it applies to records in relational datasets.

Intuitively, we say that an object is unique if it has an important property that is absent in similar objects. With this intuition in place, we tentatively define the notion of uniqueness as follows. This will be refined later by placing restrictions on what properties are applicable in the definition.

**Definition 1 (unique object).** *An object  $x$  in a dataset  $D$  is said to be unique with respect to a boolean property  $P$  that is computable for all objects in  $D$  and a threshold  $\eta$  in the range  $(0, 1)$  iff  $P(x) = \text{True}$  and  $|\{y: y \in D \wedge P(y) = \text{True}\}| < \eta |D|$*

When applied to relational datasets,  $D$  consists of the set of records in the dataset and  $P$  is a boolean property defined on the attributes of  $D$ . Without loss of generality, we treat attributes as being either categorical or numerical. As an example consider a students database. Then  $P$  could be "Is the first language of this student = English??" or "Is the marks in Physics of this student in the range (60,70)??".

In general, any "where" clause of an SQL query can be treated as a boolean property. However, we restrict ourselves in this paper to properties of the form: (categorical-attribute = value) and (numerical-attribute in range (value1, value2)). We also consider combinations of several categorical and numerical attributes along with their corresponding values and ranges.

Notice that if an object  $x \in D$  is unique with respect to a property  $P$ , then all objects  $y \in D$  for which  $P(y) = True$  must also be unique with respect to  $P$ . For convenience, we define the notion of uniqueness of a property as follows:

**Definition 2 (Unique Property).** *A boolean property  $P$  that is computable for all objects in  $D$  is unique with respect to a threshold  $\eta$  in the range  $(0, 1)$  iff  $P(x) = True$  and  $|\{y: y \in D \wedge P(y) = True\}| < \eta(D)$ , where  $x$  is a record in the dataset  $D$ .*

*Too many unique objects problem:* One shortcoming of Definition 1 is that it does not consider the "structure" of the boolean property being considered. In the case of relational data, the structure consists of attributes and their values or ranges of values. As a consequence of ignoring the structure, too many unique objects may be obtained as illustrated below.

Suppose our student database consists of 100 students and  $\eta = 10\%$ . Also, suppose that there are 10 first languages in the dataset. In a random dataset, on an average for 5 of these languages, there will be slightly less than 10 students and for the remaining 5, there will be slightly more than 10 students. It is thus *normal* that according to our definition, about 50% of the students will be considered unique with respect to their respective first language!

Clearly, our definition of uniqueness fails to satisfy our intuitive notion of uniqueness. Half of the records cannot be considered unique with respect to a single factor. If any definition of uniqueness results in a *large* number of records being labelled as unique, then the definition has failed its purpose. In order to overcome this condition, we need to neglect sets of properties that apply to a large number of records. We refer to such properties as *trivially unique*. Before we formally define such properties, we first need to develop the notion of a *sibling property*:

**Definition 3 (Sibling Property).** *For any boolean property  $P$  applicable on objects in a dataset  $D$ , the set of all specializations of a generalization of  $P$  are said to be sibling properties of  $P$ .*

By a generalization of property  $P$ , we mean any property obtained by relaxing one of the requirements of  $P$ . By a specialization of property  $P$ , we mean any property obtained by adding one more requirement on  $P$ . Notice that any boolean property  $P$  is a sibling of itself.

In the case of relational data, boolean properties may be of the form: (attribute = value) or (attribute in range (value1,value2)). A *generalization* of such a property would be to leave the value or range unspecified. This can be represented by the form: (attribute = ?) or (attribute in range (?,?)). The specializations of this generalization would correspond to all possible values or ranges that can be substituted for the '?' marks.

Thus, in the above example, the generalization for (first-language = English) would be (first-language = ?) and its specializations would include all possible first-languages in the domain. Alternatively, another kind of useful specialization is to add additional requirements on the values of attributes. For example, (first-language = English and physics-marks in (60,70)) is a specialization of (first-language = English).

**Definition 4 (Trivially Unique Property).** *If according to definition 1, the total number of unique objects with respect to  $(D, P', \eta)$  over each sibling  $P'$  of some boolean property  $P$ , is large ( $\geq \eta \mid D \mid$ ), then every unique  $P'$  is trivially unique.*

In the above example, this definition simply states that if the total number of unique students with respect to all first-languages exceeds a threshold, then every unique first-language is trivially unique.

Thus to fix the above mentioned problem of too many unique records, the boolean properties considered in Definition 1 must not be trivially unique.

We also note that the uniqueness threshold  $\eta$  would in general be inversely proportional to the number of siblings of the property being considered. Hence, we set  $\eta = \sigma/N_P$  where  $\sigma$  is a user-specified threshold in the range (0,1) and  $N_P$  is the number of siblings of the property  $P$  being considered.

The uniqueness mining problem that we consider in this paper is for relational datasets and is to extract for a given record all non-trivially unique boolean properties of the form (categorical-attribute = value) and (numerical-attribute in range (value1,value2)). If there is no single attribute that makes this record unique, we consider combinations of attributes along with their corresponding values and ranges.

### 3 The Uniqueness Mining Algorithm

In this section we describe our algorithm to mine the uniqueness of records in a relational dataset. Our task is to extract for a given record all unique boolean properties of the form (categorical-attribute = value) and (numerical-attribute in range (value1,value2)). If there is no single attribute that makes this record unique, we consider combinations of attributes.

To enable efficient algorithms to be built for uniqueness mining, we first identify the following lemmas. Their proofs follow easily from the definitions of uniqueness given in the previous section.

**Lemma 1.** *If a property  $P$  is unique in a dataset  $D$ , then any specializations of  $P$  must also be unique.*

**Lemma 2.** *If a Property  $P$  is trivially unique in a dataset  $D$ , then any specialization of  $P$  must also be trivially unique.*

**Lemma 3.** *If a Property  $P$  is unique in a dataset  $D$ , then any specialization of  $P$  must be trivially unique.*

The last lemma follows from the first lemma. If all specializations of  $P$  are unique, then these specializations together (which are all mutual siblings) will result in all records in the dataset being unique.

Due to these lemmas, we can follow a level-wise strategy of mining unique properties, starting with properties which are based on a single attribute (referred to as singletons) and then moving on to combinations of these singletons of length 2, 3, etc. At each level, we need not consider any properties that are known to be unique from the previous level, as by the second and third lemmas above, these will only result in trivially unique properties. The reader will surely notice that this processing is similar to that of the Apriori [1] algorithm of frequent itemset mining.

### 3.1 Preprocessing: Identifying Singletons

In order to allow an interactive user experience while mining uniqueness of records in a dataset, we do most of the time-consuming work in a preprocessing phase. The pseudo-code for this phase is shown in Figure 1 and explained below.

```

Preprocess( $D, \mu, O$ )
Input: Dataset  $D$ , Threshold  $\mu$ , Ontology  $O$ 
1.   for each categorical attribute  $A$ :
2.       if  $|DistinctValues(A)| > \mu |D|$ 
3.           generalize( $A, O, D$ )
4.           compute and store frequency of values in  $A$ 
5.       else:
6.           Prune( $A$ )//ignore  $A$  from now on
7.   For each numerical attribute  $A$ :
8.       cluster the values of attribute  $A$ 
9.       Compute and store the frequency of each cluster.

```

**Fig. 1.** Preprocessing Phase

During the preprocessing stage, we do the following:

1. We apply ideas from Attribute Oriented Induction (AOI) [6] to do the following: Generalize categorical attributes that have too many distinct values (e.g. addresses of people) using an ontology if available (lines 2-3 of Figure 1), or else to remove such attributes (e.g. names of people) that have too many distinct values (lines 5-6).

2. Determine ranges of numerical attributes for the purpose of defining boolean properties. We do this by applying any standard clustering algorithm on each such attribute (lines 7-9).

At the end of preprocessing, each value of unpruned categorical attributes and each range of numerical attributes correspond to singletons. After preprocessing, we identify combinations of these singletons that are unique. This is explained in the next subsection.

### 3.2 Determining Uniqueness of Records

In this phase, the user provides an input record  $x$  and desires to see all the ways in which  $x$  is unique. The algorithm then proceeds to first identify if  $x$  is unique with respect to singleton properties extracted during the preprocessing phase. If that fails, then combinations of singleton properties are considered. The detailed pseudo-code is shown in figure 2 .

<i>Uniqueness(x, D, <math>\sigma</math>, <math>\mu</math>)</i>	
<b>Input</b> : Record $x$ , Dataset $D$ , Thresholds $\sigma, \mu$	
1	k=1
2	Candidates = { $\{A\}$ : for each unpruned attribute $A$ }
3	while  Candidates  > 0
4	NonUniqProp = $\phi$
5	for each candidate $P \in$ Candidates :
6	result = IsUnique( $x, P$ )
7	If result $\neq$ 1/not unique
8	NonUniqProp = NonUniqProp $\cup$ { $P$ }
9	k=k+1
10	Candidates = Combine(NonUniqProp,k)

**Fig. 2.** Levelwise Uniqueness Mining

The above algorithm works in an iterative manner, starting with a set of candidate properties. Note that by "property", here we mean an attribute or a set of attributes. Implicitly, the values or ranges corresponding to these attributes are obtained from the attribute values of  $x$ . The initial candidates correspond to all single attributes that have not been pruned in the preprocessing phase (lines 1-2 of figure 2). The given record  $x$  has a value for each categorical attribute and lies in a range of values for each numerical attribute. Hence every attribute corresponds to a unique singleton property extracted during the preprocessing phase.

The algorithm proceeds to check for each attribute, if  $x$  is unique with respect to that attribute (lines 5-6). The details of the IsUnique function used here is shown later in Figure 3 . If the attribute is not found to be unique, it is added to a list NonUniqProp (lines 7-8). This list is used to generate candidates for the next iteration (line 10).

The Combine function used to generate candidates operates as follows: It extends each entry in NonUniqProp with one more attribute, such that all immediate subsets of the new entry are already present in NonUniqProp. In a sense, it works similar to the AprioriGen function of [1]. We now describe the details of the IsUnique function used in line 6 above. Its pseudo-code is shown in Figure 3 .

<i>IsUnique</i> ( $x, P$ )	
<b>Input:</b> Record $x$ , Property $P$	
1	$count =   \{y : y \in D \wedge P(y) = P(x)\}  $
2	$\eta = \sigma / N_P$
3	if $count < \eta \mid D$  :
4	if no.of unique records w.r.t $P < \mu \mid D$  :
5	Print $x$ is unique w.r.t $P$ //non-trivially unique.
6	Return 1
7	else:
8	return 0

**Fig. 3.** Is record unique w.r.t. Property?

In this Pseudo-code ,note that the property  $P$  represents a set of attributes. Implicitly, the required values or ranges corresponding to these attributes are obtained from the attribute values of  $x$ . The function initially determines the number of records in  $D$  that have the same attribute values and ranges as  $x$  (line 1 of Figure 3 ). Next, the function determines if the property  $P$  is unique or not by comparing this number with a threshold (line 2).

Recall from Section 2 that in the computation of the threshold,  $N_P$  is the number of siblings of the property  $P$  being considered. To illustrate: For a singleton numeric attribute,  $N_P$  would be the number of ranges that this attribute has been divided into during the preprocessing phase. For a combination of numeric attributes,  $N_P$  would be the product of number of ranges of each attribute.

Next, the function verifies whether  $P$  is trivially unique.It does this by computing the total number of records that are unique with respect to  $P$  and checking if this number is less than a threshold (line 3). If the property is found to be non-trivially unique, the property is output as being so (line 4). If the property is determined not to be unique, the function returns 0 (line 6-7).

Also,if the number of attributes are too high, we incorporate the IsUnique() function in figure 3 and Combine() function in figure 2 into preprocessing stage. Thus, we store each and every unique properties of all records in preprocessing stage itself.So,The uniqueness of any record can be obtained instantaneously irrespective of the number of attributes.

## 4 Related Work

The uniqueness mining problem introduced in this paper is related to several other areas including outlier mining,subspace clustering and frequent itemset

mining. At first glance it may be tempting to dismiss this problem as another avatar of outlier mining [2, 3]. However, in the latter, we are interested in extracting records that are significantly different from the majority; here, we are interested in extracting the properties that make any given record seem like an outlier. It is possible to extract outliers using our approach in the following way: If a record is an outlier its unique properties would be very "dominant". It is likely to have unique properties that are very general. It is also likely that these properties can be extracted in our approach using very lenient thresholds. It is not possible to use general-clustering based outlier detection methods to determine unique records. This is because the clusters obtained using general clustering algorithms on  $n$ -dimensional data objects is hard to characterize. In our approach, we apply clustering algorithms separately on each dimension - this is used only to identify good ranges for each dimension. Recent and upcoming works in subspace clustering [4] may eventually resolve the above problem. These studies aim to identify clusters in sub-spaces of dimensions, rather than on all dimensions at once. If this is possible efficiently, then records present in very small subspace clusters can be treated as unique. However, the work in this paper is different in the sense that it aims to output the unique properties of *any* input record.

## 5 Experiments

In this section, we evaluate the Uniqueness Mining algorithm using synthetic relational datasets. In these experiments we attempt to demonstrate that the algorithm and framework are useful in that the unique properties of records are output. We do this by adding handmade records to the dataset whose properties are significantly different from the remaining records. We then show that our algorithm is able to identify the unique properties of the hand-made records accurately. An important point is that the output of our algorithm is small enough for a human to parse. This is due to the pruning of properties that are trivially unique. Next, we also demonstrate the scalability of our approach by measuring its response times on two synthetic datasets of very different sizes and a real dataset. In all cases, the response times of our approach were instantaneous.

The experiments were run on a computer with a 2.6 GHz Intel Pentium IV Processor and a RAM of 512 MB and the data was stored in a 40 GB DDR HDD. All the algorithms have been implemented in Python (version 2.4) and the data was stored in a MySQL relational database server (version 5.0.22).

We generated datasets that replicate a student information database. The attributes used were name, gender, major, Date of birth (DOB), telephone, city of residence, GPA, major course Grade. In our experiments we fix  $\mu = 10\%$  and  $\sigma = 10\%$ .

During the preprocessing stage, the numerical attribute GPA was clustered using the CURE clustering algorithm [5]. Similarly, other attributes were clustered as described in the algorithm and the frequencies of clusters were stored in a database.

### Experiment 1

In this experiment we verified the accuracy of the algorithm in finding already known hand-made unique tuples. We introduced 30 hand-made records in a 1000 record dataset. We illustrate our output using one such record (Gender:"F", Major:"ECE", Residence:"City X, Country Y", GPA:6.13, Major course grade: A). This record was created uniquely in the combination (GPA, major course grade)-usually there are very few people with the given combination of values for GPA and major course grade.

In the preprocessing phase, the attributes 'name', 'phone' and 'DOB' were pruned by keeping  $\mu = 10\%$ . There were 1000, 874 and 1000 distinct values in these attributes, respectively. This clearly shows that even though a record may be unique in these attributes it would be trivially unique and uninteresting. Thus the attribute elimination methods in the preprocessing stage prove to be very useful.

When we run the algorithm with the  $\sigma = 10\%$ , we obtained the tuple as unique in the subspace ('GPA', 'Major course grade'), as expected. We also obtained that the record is unique in the subspace ('GPA', 'Residence'). The record did not contain any special properties based on a single attribute.

Further details of the statistics of the database with respect to this record are as follows:

The cluster that contains GPA=6.13 contained 46 records. The cluster with Grade=A consists of 95 records which are too many when compared to the threshold values. But the combination contains only 1 point while the minimum threshold value evaluates to approximately 2 in this case. This shows that although there are many average and intelligent students, there are a few people that have an average overall GPA and also have a very good grade in their major course.

This may be very useful for the hiring wing of a company who are looking for students who have got good grades in their major course but not based only on their GPA. Also, this might be very useful for those unique students who are normally rejected based on their GPAs.

Additionally, we come to know that there is only one person in 130 students from City X in the GPA range 6.04-6.28 showing that the student is unique in the subspace ('gpa', 'residence').

### Experiment 2

In this experiment we deal with the scalability of the algorithm. We created a synthetic data set of the same format as in the above experiment consisting of 1 Million records. The dataset in the previous experiment had only 1000 records.

The complexity of the preprocessing stage is proportional to the complexity of the clustering algorithms and the speed of query retrieval of the relational database servers. The algorithm we use for clustering numerical attributes is CURE. So the preprocessing stage would be of the order of  $O(n^2 \log(n))$ .

In this dataset, only 'Name' and 'Phone' are removed during preprocessing and the clusters of other attributes are stored in the database. We ran the

level-wise algorithm on the resulting database containing a million records and 6 attributes. The algorithm was executed in 26 seconds on an average.

One interesting result was obtained when the input record was (gender:'F', major:'CIVIL', DOB:1-11-1986, Residence:'City X', GPA:6.23, majorcourseGrade : 'A'). The uniqueness of this record was found in the subspaces (GPA, Major course grade); (DOB,residence,major);(DOB,residence,gender); (DOB,major, gender).

In this case out of a cluster of 100949 people having 'A' in Major course grade and 45981 people having GPA range of (6.34 - 6.04), only 2 people lie in the intersection of both the clusters making them very special indeed!

### Experiment 3

Unlike the earlier experiments, where we have dealt with the scalability and accuracy of the algorithm with respect to datasets with less number of attributes and fairly large number of records, this experiment verifies the accuracy and scalability of our algorithm using large number of attributes.

For this experiment, we have used a dataset containing the answers of students in a competitive exam. There are 50 attributes in the dataset each representing a question in the exam. The value of each attribute can be either 'yes' or 'no', 'yes' if the answer is correct, 'no' if it isn't correct. Totally, there are 1000 attributes in the dataset.

The intuitive approach of our algorithm is to find unique candidates in these exams. Normally, the candidates are assessed based on their final marks only, but this algorithm provides another scope of ranking the candidates.

We can find some unique students from the total set who have answered a set of answers correctly, while many other students have answered them incorrectly. These unique students can be given some bonus marks based on the toughness of the questions. This can ensure that talented students do not fail to qualify in competitive exams, just because they have wasted some extra time answering tough questions.

For this, The threshold value  $\sigma$  is set to 0.1.

For the given threshold value, we found a student having the following unique set of correct answers.

{'ques32': 'yes', 'ques36': 'yes', 'ques21': 'yes', 'ques4': 'yes', 'ques42': 'yes'}

Normally, the student got 28 marks out of 50. But, he was able to answer these questions which others were not able to answer, wasting some valuable time. So, he may be given some bonus marks as he had answered some relatively tough questions.

## 6 Conclusions

In this paper we introduced the problem of Uniqueness Mining. We are interested in determining what makes a given record unique or different from the majority of the records in a dataset. We have discussed several examples of applications

where this may be useful. Our solution to this problem uses ideas from clustering, attribute oriented induction (AOI) and frequent itemset mining. Most of the time-consuming work is done in a preprocessing stage and the online computation of the uniqueness of a given record is instantaneous. We demonstrated the accuracy and scalability of our approach using synthetic datasets.

Future work includes designing faster algorithms. This is necessary for applications where there are a large number of attributes. Also, in many applications, there may be a notion of different weights to attributes - some attributes may be more important in defining uniqueness. Further, numerical attributes such as GPA or salary usually have a bias - higher (or lower) values typically mean that the record is better in some sense. This may be useful in defining or ranking the unique properties of records.

## References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of Intl. Conf. on Very Large Databases (VLDB) (September 1994)
2. Agrawal, C.C., Yu, P.S.: Outlier detection for high dimensional data. In: Proc. of ACM SIGMOD Intl.Conf. on Management of Data 2001 (2001)
3. Ng, R.T., Breunig, M.M., Kriegel, H.-P., sander, J.: Identifying density based local outliers. In: Proc. of ACM SIGMOD Intl. Conf. on Management of Data (2000)
4. Knorr, E.M., Ng, R.T.: Finding Intensional Knowledge of Distance-Based Outliers. In: Proc. of VLDB 1999 (1999)
5. Gunopulos, D., Agrawal, R., Gehrke, J., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proc. of ACM SIGMOD Intl.Conf. on Management of Data (1998)
6. Sudipto Guha, R.R., Shim, K.: An efficient clustering algorithm for large databases. In: Proc. of ACM SIGMOD Intl. Conf. on Management of Data 1998 (1998)
7. Cercone, N., Cai, Y., Han, J.: Attribute-oriented induction in relational databases. In: Knowledge Discovery in Databases, AAAI/MIT Press (1991)