A Method of Deduplication based on Inconsistency

Wu Yunpeng[1, a *], Lv Yiting[1] ,Sun Yuping[1], Hu Qiang[1], Wu Mingfei[1], Guo Yu[1] and Wang Decai[1]

[1]Institute of NBC Defence of the PLA, Beijing, 102205, P.R. China

[a]ypwu_cn@163.com

Keywords: Data cleaning, Deduplication, Inconsistency, Functional Dependency, Index.

Abstract. Reducing the number of comparisons is the most common way to improve the effectiveness of data cleaning. We investigate the problem by using inconsistency. We split redundant data into three categories. For each category, we give an algorithm and analyze its complexity, and combine them together finally. In particular, we address the chasing problem for the method under functional dependency. At the last, we experimentally verify that these algorithms effective and scale well, and that the method helps us more efficiently detecting duplications.

Introduction

"Real-world data is dirty" [1], noise in data exists in the form of redundancies and errors. Therefore, data usually needs to be cleaned before application. Deduplication is one of the most important branches of data cleaning. In practice, reducing the number of comparisons is the common way to solve the problem[2,3]. Files(or tables) firstly are blocked into different blocks, and comparisons are made only within block [4,5]. Determining appropriate blocks is important, since it directly influence the results. A good blocking method can dramatically reduce comparisons while have small impact on the results. Traditional methods [1,5] usually block records depending on their key values. There are two disadvantages for methods blocking based on key, (1) redundant records with error key can not be well detected, and (2) the key is often chose by experts according to their domain-specific knowledge, which is time-consuming and error-prone.

Dependencies are introduced to capture more semantics and understand construction of relational model [7]. However, redundancies and errors bring inconsistency. Recently, it has been noticed that the inconsistency can be used for data cleaning. Constructing matching rule based on inconsistency was firstly proposed by W. Fan et al.,[8,9,10]. They proposed a notion of "CFD" [8] to capture the inconsistency in database, and gave rules to help to choose key [9]. Inconsistency occurs when two records violate a dependency. In this paper, we propose a novel approach that improves the effectiveness of deduplication based on inconsistency.

Reduction based on inconsistency

Preliminaries: Notions and Example. Let 1 be two records that violate a functional dependency(FD) , then (1) values of r and under LHS attributes of are equivalent, while (2) values of them under RHS attributes of are different. For convenience, We use $LHS_{\sigma}(RH$ to

denote LHS(RHS) attributes of a FD , or simple LHS(RHS) if is understood from the context. If A is a set of attributes, then we use A(r) to denote the value of r under attributes of A. In particular, if T is a set of records, then A(T) represents the set that unites all A(, where r_i €.

Proposition 1: Let ı be two records that violate a FD , then there exists a set A ⊆ RH of attributes such that A(r) ≠ A(, and error value occurs in at least one of them.

Definition 1: A violation is a record pair <r, r'> where r, r' are two records that violate a FD.

Obviously, inconsistency occurs if there exists violation in a database table, and all violations under a given FD can be detected by the following SQL query Q_{vio}:

SELECT ∗
FROM t_1
WHERE $t_1.X_1 = t2.X1$ AND … $t_1.X_k = t_2.X_k$
AND (t_1:$Y_1 \neq t_2.Y_1$ OR … $t_1.Y_m \neq t_2.Y_m$)
GROUP BY $t_1.X_1$, …, $t_1.X_k$

where, t_1 and t_2 are same talbe. X_1, …, X_k and Y_1, …, Y_m are LHS and RHS of the FD, respectively.

Example 1: Consider a relation which contains 6 attributes: "NAM", "CIT", "ZIP", "COU", "STR", "PHO". Every record contains information that identifies an unique house. House holder's name is stored in "NAM","ZIP", "STR", "CIT", "COU" represent post code, street, city, country of house, respectively. "PHO" stores the telephone number of house. Four FDs defined on the relation are shown as following:

fd_1: "ZIP", "COU" → "CIT"
fd_2: "STR", "CIT" → "ZIP"
fd_3: "PHO" → "COU"
fd_4: "PHO" → "CIT"

An example database is shown as in table 1. Records r_1, r_2 and r_5 identify same entity whose holder's name is "Peter Ludwig Berger". However, errors occur in values of r_2 and r_5 under different attributes ("CIT" and "PHO"of r_2, "CIT" and "ZIP" of r_5). Records r_3 and r_4 identify other two different entities whose holders' names are "Allan R. Robinson" and "Angle Yang", respectively. Errors occur in values under "COU" of r_3 and "ZIP" of r_4. In particular, all these houses located in the same street of the same city of the same country. Therefore, records were supposed to have same values under attributes "CIT", "ZIP", "COU", "STR". Obviously, there are total 5 violations can be detected by using query like Q_{vio} under each FD, <r_1, r_2>, <r_4, r_5>, <r_1, r_4>, <r_3, r_4>, and <r_1, r_5>. Therefore, two pairs of duplicate records r_1, r_5 and r_2, r_5 can be discovered through only five comparisons by using suitable matching rules.

Table 1. Tuples of example

Records	NAM	CIT	ZIP	COU	ST R	P HO
r_1	PeterLudwigBerger	EDINBURGH	EH8 9AB	UnitedKingdom	BRIT ON	07882374999
r_2	P eterL.Berger	EDINBUG	EH8 9AB	UnitedKingdom	BRIT ON	07882374899
r_3	AllanR:Robinson	EDINBURGH	EH8 9AB	UnitedLingdom	BRIT ON	07365874938
r_4	AngleY ang	EDINBURGH	EHu 9AB	UnitedKingdom	BRIT ON	07962851774
r_5	P eterL.Berger	EDINBUG	EHu 9AB	UnitedKingdom	BRIT ON	07882374999

Blocking based on Inconsistency Detecting. Given any pair of records $\langle r, r$, it can be classified in to three categories according to their values on different sides of some FD , (1) $LHS_\sigma(r) = LHS_\sigma($ and $RHS_\sigma(r) \neq RHS_\sigma($, (2) $LHS_\sigma(r) \neq LHS_\sigma($ and (3) $LHS_\sigma(r) = LHS_\sigma($ and $RHS_\sigma(r) = RHS_\sigma($. We analyze each of the categories in the following of the section.

1) Detecting Inconsistency in Static: We have seen that violations under a FD can be detected by a simple SQL query. However, executing the query costs $O($ time, where n is the number of records. It is unacceptable when n is large.

Proposition 2: Given a set of records and a FD, detecting violations under the FD is solvable in liner time.

Actually, for a given FD , the the detecting process can be divided into two phases(TP). In the first phase, records are divided into different partitions according to their values under LH. Records in same partition share common value under LH. In the second phase, each partition obtained from the first phase is further divided into different sub-partitions according to their values under RH. Records in same sub-partition share common value under RH. Obviously, for any two records , if they belongs to same partition but different sub-partition, then they satisfy $LHS(r) = LHS($ and $RHS(r) \neq RHS($, thus record pair $\langle r, r$ is a violation under . For convenience, the two phases together is simply referred to as the two phases(TP), and the result of TP is a set of blocks of I under , referred to as partition and represented as $PAR(I$.

Given a FD , we construct two hash tables for the partition. In the first hash table, the key is hash value of each record value under LH, and each row contains a set of record values under RH. In the second hash table, the key is hash value of each record value under RH, and each row contains a set of records. Obviously, the time and space costs of constructing these two hash tables are in O(n), where n is the number of records.

Proposition 3: The violations detected by these two phases are identical with those obtained by executing query like Q_{vio}.

It is clear, for a given instance and a FD, either the first phase or the second phase can be done in liner time. In addition, if given a set of FDs, the time cost will be in $O(n$, where n is the number of records and m is the number of FDs.

2) Detecting Inconsistency by Chasing: Two records violate a FD means that there exists an attribute such that . On the other side, a FD defined on any two records requires that if their values under LHS equivalent then their values under RHS should also equivalent. That is to say, values A(r) and A(r') were highly possible same value, if no error occurs. If satisfy , then they can not be detected by the TP. A suitable way to reduce the impact of this situation is to find out these values, and assume these values are equivalent, then detect violations based on the TP.

Example 2: Recall the example 1, although records r_2 and r_5 are duplicate, they can not be detected by the method depicted in section 1), since there does not exist any FD such that . Therefore, duplicate record pair can never be found, no matter using what matching rule.

On the other side, according to the FD fd_2, the violation can be detected, since STR(r_3) = STR(r_4) and CIT (r_3) = CIT (r_4) and ZIP (r_3) ≠ ZIP (r_4). If ZIP (r_3) and ZIP (r_4) are considered to be same value, i.e., let values "EHu 9AB" and "EH8 9AB" equivalent. Then, reconsider the FD fd_1, violation can be detected.

Example 2 reveals that violations that have already been detected can be used to find more potential violations. It is obviously that the process is iterative. We introduce a notion of chase [8] to capture the properties of the process. Given an instance set of records, a set of FDs and a collection of sets of values, the chase is based on successively applying the following rule

Rule 1: Let all values in same set of the collection be equivalent, detect violations under all the FDs, update the collection.

A chasing sequence is a sequence of partitions, each partition is obtained by applying TP under a FD together with a collection of equivalent sets. The sequence is terminal when there is no FD in can be applied on the instance. Note, a FD can be applied on an instance means there exist new violations can be detected by TP under the FD.

Definition 2: Given a set of records S and a set of FDs, the chase of S under , denoted by chase(S,), is the set of violations that can be obtained from a terminal chasing sequence.

Given a partition P of I obtained based on the TP under, values that potentially equivalent can be obtained from P. Suppose is a block that contains more than one divisions, for each attribute , all values in set A(B) should be considered as equivalent. Suppose B' is another block that different from B, if there exists a value appears in A(B) and A(B') simultaneously, then values in should be considered as equivalent. If we put all

values that potentially equivalent into one set, a collection C of such sets is obtained. Calculating such collection is solvable in liner time by constructing two hash tables and two graphs.

Theorem 1: Given a set of records and a set of FDs, the chasing process terminates, and the chase is identical.

Proof: A necessary condition keeps the chasing process continue is that new violations can be found. There exists a upper bound(n^2) for the number of violations, where n is the number of records. In addition, according to the chasing rule, the chasing process only adds new violations. Therefore, the number of violations is monotone increasing, and thus the chasing process terminates.

Suppose two terminal chasing sequences **P** =$P_1,\ldots,P_k$ and **P'**= $P'_1,\ldots,P'_m$ are obtained by applying TP under two different FD sequences s_1 and s_2, and S, S' are two different sets of violations obtained from **P** and **P'**, respectively. Without loosing generality, suppose $S \setminus S' =$ and $S_\Delta \neq$. Sequence **P*** = $P'_1,\ldots,P'_m$, $P_1,\ldots,P_k$ is the union of **P'**and **P**, it can be obtained by applying TP under FD sequence s_1 unites s_2. It is clear that the set of violations obtained from sequence **P*** is the union of S and S', that means sequence **P'** is not a terminal chasing sequence, which contradicts with the assumption.

Usually, the collection C is empty at the beginning of the chasing process, and updates itself after each partition is obtained. From the definition 2 and the chasing rule, we can infer that there are two necessary conditions which keep the chasing process continue. The first one is that there exist two FDs and such that $RHS_{\sigma_1} \cap LHS_{\sigma_2} \neq$, which referred to as influence relation from to . Given a set of FDs, the influence relations between them can be represented by an influence graph, where each node is labeled with a FD of , and for any two FDs , □ such that $RHS_{\sigma_1} \cap LHS_{\sigma_2} \neq$, there is a directed edge from the node labeled with to the node labeled with . The influence graph of the four FDs depicted in example II.1 is shown as in Fig. 2. Constructing the influence graph of a set FDs is solvable in $O(p^2)$ time, where p is the number of FDs. The other necessary condition is that C can still be changed, otherwise after all violations have been detected by TP under each FD of , the process will terminates.

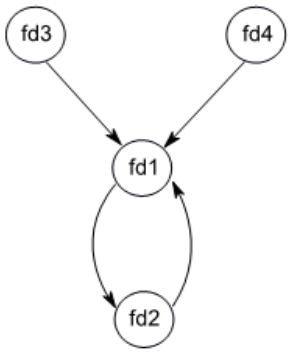

Figure 2. Influence graph of the four FDs in example 1

A chasing process can be illustrated by an influence graph. Suppose the influence graph of a set of FDs is G. for any partition P that obtained from the TP under a FD σ, the collection C changed based on P can only P′, where P′is obtained from a FD σ', such that the there is a directed path from the node labeled with to the node labeled with . In the next section, we will investigate algorithms according to different influence graphs.

3) Deduplication with Consistency: Until now, we only considered detecting violations with inconsistency. However, if given two records r, r' and a set of FDs, r and r' consistent with all FDs of , let set be the attribute set that contains all attributes appear in , then $A_{\Sigma}(r) = A_{\Sigma}($. Obviously, they potentially duplicate with each other and finding such comparisons is necessary.

Proposition 5: Given a set of records and a set of FDs, finding all pairs of records that not a violation is solvable in liner time.

It can be achieved by constructing a hash table. The key is the hash value of record value under , each row of the hash table is consist of a set of records. Constructing such an index for each record is solvable in O(n) time and O(n) space. where n is the number of records.

Summary

We have proposed a method to reduce the number of record comparisons based on inconsistency, and provided algorithms for the method. Our experimental results have verified the effectiveness and scalability of our methods. Future work in this field includes: (1) improve the performance of existing algorithm for BID when influence graph is SCG, a possible way is to develop more efficient index that trade space for time optimally; (2) we only consider the functional dependency in this paper, join dependency and inclusion dependency should be considered in our future work; (3) combine matching rule into the methods for real-life application.

References

[1] M. A. Hern´ andez and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," Data Min. Knowl. Discov., vol. 2, no. 1, pp. 9–37, 1998.

[2] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," Journal of the American Statistical Association, vol. 64, no. 328, pp. 1183–1210, 1969.

[3] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," IEEE Trans. Knowl. Data Eng., vol. 19, no. 1, pp. 1–16, 2007.

[4] R. Baxter and P. Christen, "A comparison of fast blocking methods for record linkage," 2003, pp. 25–27.

[5] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," Journal of the American Statistical Association, vol. 84, no. 406, pp. 414–420, 1989.

[7] S. Abiteboul, R. Hull, and V. Vianu, Foundations of Databases. Addison-Wesley, 1994.

[8] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, “Conditional functional dependencies for data cleaning,” in ICDE, 2007, pp. 746–755.

[9] W. Fan, X. Jia, J. Li, and S. Ma, “Reasoning about record matching rules,” PVLDB, vol. 2, no. 1, pp. 407–418, 2009.

[10] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, “Improving data quality: consistency and accuracy,” in VLDB ’07: Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007, pp. 315–326.

