Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2009 Proceedings

Americas Conference on Information Systems (AMCIS)

2009

Assessing Accuracy with Locality-Sensitive Hashing in Multiple Source Environment

Jingyu Han
Nanjing University, hjysky@gmail.com

Dawei Jiang
National University of Singapore, jangdw@comp.nus.edu.cn

Lingjuan Li
Nanjing University, lilj@njupt.edu.cn

Zhiming Ding
Chinese Academy of Science, zhiming@iscas.ac.cn

Assessing Accuracy with Locality-Sensitive Hashing in Multiple-Source Environment

Jingyu Han
School of computer,Nanjing University of Posts and Telecommunications, P.R.China
hjysky@gmail.com

Dawei Jiang
School of computing, National University of Singapore, Singapore
jangdw@comp.nus.edu.cn

Lingjuan Li
School of computer,Nanjing University of Posts and Telecommunications, P.R.China
lilj@njupt.edu.cn

Zhiming Ding
Institute of Software, Chinese Academy of Science, P.R.China
zhiming@iscas.ac.cn

ABSTRACT

Accuracy assessment is a key issue in data quality management. Most of current studies focus on how to qualitatively analyze *accuracy* dimension and the analysis depends heavily on experts' knowledge. Seldom work is given on how to *automatically* quantify *accuracy* dimension. Based on Jensen-Shannon Divergence (JSD) measure, we propose a*ccuracy* of data can be automatically quantified by comparing data with its entity's *most approximation* in available *context*. To quickly identify *most approximation* in large scale data sources, Locality-Sensitive Hashing (LSH) is employed to extract *most approximation* at multiple levels, namely column, record and field level. Our approach can not only give each data source an objective *accuracy* score very quickly as long as *context* member is available but also avoid human's laborious interaction. Theory and experiment show our approach performs well in achieving metadata on *accuracy* dimension.

Keywords

Data quality, Accuracy, Jensen-Shannon Divergence (JSD) , Locality-Sensitive Hashing (LSH), Context

INTRODUCTION

Data quality assessment is as follows. Given a set of data sources which describe the same set of entities, we are to give each data source a score indicating how *good* that data source is in terms of quality dimensions including *accuracy*, *completeness*, *consistency*, etc (Aebi and Perrochon, 2000; Wand, 1996). This is a pressing concern in multiple-source environment, especially in CIS (Scannapieco, 2004). The *goodness* of data can be measured in terms of many dimensions but in this paper we only focus on *accuracy* dimension as many other quality dimensions are related to it (Motro, 2006; Parssian, 2002). Intuitively, *accuracy* gives to what extent correctness of data is satisfied.

A possible way to solve above problem is to invite a human expert to identify *all* the *perfect representations* of the entities. Then, *accuracy* can be measured by comparing data with their *perfect representations* based on some measures (Kukich, 1992). Unfortunately, this approach does not scale well for large multiple-source environment since it introduces *huge* human efforts. In addition, people usually use ad hoc ways to compare different types of data to quantify *accuracy* and it becomes a necessity to find a universal *accuracy* measure for different types of data.

We propose a novel approach to Quantify *Accuracy* within Context at Multiple levels (QACM). Our QACM approach can *automatically* and *quickly* assess data source in terms of *accuracy* dimension with *little* human efforts, which is the greatest advantage of our approach. Our solution is based on this observation. In a domain specific multiple-source environment, especially in CIS system, the same entity is often redundantly stored in many data sources and the same syntax notation usually denotes the same entity. Although each data source in CIS is independently maintained and unintentional errors often occur, it is not likely that all the copies of the same entity are simultaneously wrong. Naturally, we propose that *most approximation* can be extracted among all overlapped data sources by vote-fusion （Bergamaschi，1999） and *accuracy* can be figured out by comparing data with *most approximation*. Specifically, *accuracy* is assessed with three phases as follows.

Phase 1: Identifying Context

By mapping each data source into a q-gram metric space (Kukich, 1992), *context* of one data source is quickly identified, which provides highly relevant data sources to be compared with .

Phase 2: Extracting Most Approximation

It consists of two steps: preprocessing and extraction. In preprocessing, objects (column, record or field) are mapped onto buckets based on LSH signatures (Gionis, 1999; Cohen, 2001; Indyk, 1998) such that the probability of collision is much higher for objects that are close to each other than for those that are far apart. In extraction, we first find every object's top K Nearest Neighbors (KNN). Then, *most approximation* of each object is chosen by vote-fusion among all similar ones.

Phase 3: Measuring Accuracy

Accuracy is quantified by comparing data source with its *most approximation* based on Jensen-Shannon Divergence (JSD), which gives birth to a universal *accuracy* measure for different types of data.

In particular, contributions of this paper are made as follows.

- We propose data source's *most approximation* can be derived from its available *context* and thus *accuracy* can be *automatically* quantified by comparing data source with its *most approximation* with little human's interaction.
- *Most approximation* of column, record or field is extracted by vote-fusion among all its KNNs from contextual data sources. This guarantees that a reasonable *most approximation* can be found by searching a very small portion of data.
- LSH technique is exploited to retrieve approximate KNNs when *most approximation* of data source is extracted at different levels. It scales well with size of each data source and total context data source number.

RELATED WORK

Quality evaluation can be divided into two categories. The first category focuses on *qualitatively* analyzing data quality dimensions (Aebi, 1993; Bouzeghoub, 2004; Orr, 1998; Wang, 1993, 1996). These research mainly gives what data quality means and how to measure data quality qualitatively and it is needed to ask people to give the quality rating or quality tags. This is too laborious and even impossible for large scale data sources. Literature (Chiang, 2008) proposes to discover conditional functional dependencies to identify conformant and non-conformant records. It aims at *consistency* dimension and does not touch on *accuracy* dimension.

The second category deals with how to *quantitatively* assess quality of relational data (Motro, 1996; Parssian, 1999, 2002). Literature (Parssian, 1999, 2002) gives methods to estimate *accuracy* and *completeness* in terms of record level. Literature (Motro, 1996) gives an approach to assess *soundness* and *completeness* of relational data. Main drawback of these approaches is that manual verification is very laborious. In fact, it is not a good policy and even infeasible for large data sources. Literature (Ballou, 2006) proposes to estimate query result quality by sampling from base tables. Of course, it can not precisely quantify data source's quality.

The work closely relevant to ours is how to assess data quality in CIS (Bertolazzi, 2001; Majkic, 2004; Mecella, 2002; Scannapieco, 2004). In literature (Bertolazzi, 2001), a methodological framework for data quality in cooperative systems has been proposed, consisting of five phases (i.e., definition, measurement, exchange, analysis and improvement). DaQuinCIS(Majkic, 2004; Mecella, 2004; Scannapieco, 2004) is also a platform to address interoperability in CIS. But they do not give how to determine the reference against which data quality can be gauged and how to effectively quantify quality dimensions in CIS.

PROBLEM SETTING AND BASIC CONCEPTS INVOLVED

Definition 1*:* Given a set of entities $E = \{e_1,\ldots,e_n\}$, a Cooperative Information System (CIS) is N data sources $D = \{T_1,\ldots, T_N\}$ which store E as follows. Each data source $T_i \in D$ contains a lot of records, i.e., $T_i = \{r_1,\ldots,r_n\}$ and each record $r_i \in T_i$ is a copy of e_i with unexpected modifications such as missing data, wrong data or non-standard representations , etc.

Our aim is to give each data source Ti a score to indicate its *accuracy* with regard to E. Note schema mapping can be automatically obtained by many schema mapping techniques(Bergamaschi, 1999; Palopoli, 2000) or data profiling tools (Dasu, 2002; Evoke).

Unintentional error of the data is due to random reason to some extent, which further determines how much information is conveyed by data. Hence, it is intuitive to borrow some concepts from information theory to quantify *accuracy*.

Definition 2*:* Given two distributions $P(X)$ and $Q(X)$, their Jensen-Shannon Divergence (JSD) is defined as

$$\mathrm{JSD}(P||Q) = \frac{1}{2} D(P||M) + \frac{1}{2} D(Q||M),$$

where $M = (P + Q)/2$, $D(P||M)$ is differential entropy between P and M and $D(Q||M)$ is differential entropy between Q and M.

QUANTIFYING DATA SOURCE'S *ACCURACY* IN CONTEXT

Given a data source T_i, although its *perfect representation* E is not available, a number of copies of E (including T_i) are stored in D and we can approximate E from its redundant copies.

Identifying data source's context

Definition 3*:* Given a data source T_i, similarity measure sim(.,.) and similarity threshold θ_{con} , context of a data source T_i is

$con(T_i) = \{T_j \,|sim(T_i, T_j) \geq \theta_{con}, j = 1,\ldots,N\}$.

Namely, $con(T_i)$ consists of all the syntax-similar data sources. Similarity between T_i and T_j is calculated with two steps. *First*, we compute the similarity between every corresponding columns $T_i[k]$ and T_j $[k]$ for $1 \leq k \leq m$ (m is total number of attributes of data source). In this paper we adopt *Jaccard coefficient*, which belongs to *LSH* family (Indyk, 1998), as such

$sim(T_i[k], T_j[k]) = Jacc(T_i[k], T_j[k]) = |G_q(T_i[k]) \cap G_q(T_j[k])|/|G_q(T_i[k]) \cup G_q(T_j[k])|$,

where $G_q(T_i[k])$ is q-gram multiset (Kukich, 1992). For numerical value, we can also embed it into q-gram metric space by dividing it into a sequence of intervals.

Second, similarity between two data sources T_i, T_j with m common columns is calculated as

$$sim(T_i, T_j) = \frac{1}{m} \sum_{k=1}^{m} \omega_k * sim(T_i[k], T_j[k]) \ .$$

We collect *similar* data sources of T_i and regard all the similar data sources (including T_i) as *context* of T_i, which is achieved by Alg.1. We can see time of scanning all the data sources is dominant in Alg.1.

Algorithm 1: identifyDataSourceContext

Input: T_i , $D=\{T_1, T_2, \ldots, T_N\}$, column weights $\omega_1, \ldots, \omega_m$ and similarity threshold θ_{con}

Output: $con(T_i)$
$con(T_i) \leftarrow \varnothing$;
Each column of T_i is mapped into q-gram metric space,denoted as $G_q(T_i[k])(1 \leq k \leq m)$;
for $j \leftarrow 1; j \leq N$ **do**
  accSim ← 0;
  for $k \leftarrow 1$ **to** m **do**
   map T_j $[k]$ into q-gram space,denoted as $G_q(T_j[k])$;
  accSim ← accSim + ω_k *$|G_q(T_i[k]) \cap G_q(T_j[k])|/|G_q(T_i[k]) \cup G_q(T_j[k])|$;
  end
 if accSim/m $\geq \theta_{con}$ **then** $con(T_i) \leftarrow con(T_i) \cup \{T_j\}$;
end
return $con(T_i)$;

Furthermore, top K context of T_i, denoted as $con^K(T_i)$, is defined as follows.

Definition 4*:* $con^K(T_i)$ is the set constituted by top K similar data sources in $con(T_i)$.

$Con^K(T_i)$ can be similarly obtained as Alg.1 except that it should return the top K ones.

Extracting Data Source's Most Approximation

We resort to LSH technique to extract data source's *most approximation* by regarding each column (record or field) as a point in q-gram metric space. The principle behind LSH is that collision probability of two points is closely related to the similarity between them. Specially, the smaller the similarity is , the smaller the collision probability is. This can be summarized as

theorem 1.

Theorem 1*:* Let sim(.) be set resemblance measure (such as Jaccard coefficient) of space Γ. Then, for $1 > \rho_1 > \rho_2 > 0$, there exists a family H of hash functions such that the hash family is $(\rho_1, \rho_2, \rho_1, \rho_2)$-sensitive. That is to say, for any two point u, v

(1) if $u \in B(v, \rho_1)$, then $\Pr[h(u) = h(v)] \geq \rho_1$,

(2) if $u \notin B(v, \rho_2)$, then $\Pr[h(u) = h(v)] \leq \rho_2$

where $B(v, \rho) = \{p : \text{sim}(p, v) \geq \rho\}$.

In order for a locality-sensitive family to be useful, it has to satisfy the inequality $\rho_1 > \rho_2$. Family of hash function used by LSH can be extended naturally from set to multiset (Haveliwala, 2000). Alg.2 gives how to map a set of points into hash tables.

Algorithm 2: LSHHash
Input: Parameter κ and ℓ, a family of LSH functions $\{h_1(.), ..., h_\pi(.)\}$, a set of points in Γ
Output: Hash tables T^j , $j = 1, .., \ell$
foreach $j = 1, ..., \ell$ **do**
  Initialize hash table T^j by generating a random hash function $g_j(.) = (h_{(j-1)*\kappa+1}(.), ..., h_{j*\kappa}(.))$;
end
foreach $j = 1, ..., \ell$ **do**
  foreach point $p_i \in \Gamma$ **do**
    store identifier on bucket $g_j(h_{(j-1)*\kappa+1}(p_i), ..., h_{j*\kappa}(p_i))$ of hash table T^j;
  end
end

Second, to identify the similar ones of query point u (a column, a record or a field value), u is mapped onto buckets with the same hash functions as used in Alg.2 and all the points encountered are regarded as candidates. Note we only need to search similar points stored in $\text{con}^K(T_{father}(u))$ (here $u \in T_{father}$) because similar ones are more likely stored in these data sources and we do not need to search whole context. This is achieved by Alg.3 and its total times of accessing hash tables is bounded by $O(K\ell)$. The following three sections address how to efficiently extract *most approximation* at column, record and field level respectively by virtue of LSH technique.

Algorithm 3: findSimilar

Input: A point $u \in \Gamma$, $\text{con}^K(T_{father}(u))$

Output: similar ones Φ

$\Phi \leftarrow \varphi$;

foreach $T_i \in \text{con}^K(T_{father})$ **do**

  for $j \leftarrow 1$ **to** ℓ **do**

    map u based on $g_j(.)$;

    $\Phi \leftarrow \Phi \cup$ points found in buckets $g_j(u)$ of hash table T_i^j ;

  end

end

return Φ;

Extracting Most Approximation at Column Level

In this policy, similarity between two data sources, say T_i and T_j , is calculated at column level. *Most approximation* of T_i is the concatenation of its every column $T_i[k]$'s *most approximation*s (here $1 \leq k \leq m$). Specifically, frequency of a certain q-gram is total times of occurrences in one column. *Most approximation* is achieved by two steps as follows.

Step 1: Preprocessing to Store Each Column Onto Buckets Based on the Column's LSH ***Signatures***

Each column that belongs to *context* is regarded as a point and mapped onto some buckets with Alg.2. Based on theorem 1, we know that if two columns are similar, there is a high probability that they are mapped onto the same bucket. At the end, ℓ hash tables are constructed in *context* and the buckets in $T^j[k](1 \le j \le \ell, 1 \le k \le m)$ store the k-th columns that agree with $g_j(.)$.

Step 2: Extracting Most Approximation

T_i's *most approximation* with regard to E, denoted as T_i', is composed of ($T_i[1]'$, $T_i[2]'$, ..., $T_i[k]'$, ..., $T_i[m]'$), where $T_i[k]'(1 \le k \le m)$ is *most approximation* of column $T_i[k]$. Vote-fusion policy is used to determine $T_i[k]$'s *most approximation*. Specifically, the column whose number of similar ones is largest is selected as the *best* one. Concrete steps are described in Alg.4. As *findSimilar* algorithm's running time is bounded by $O(K\ell)$, time complexity of Alg.4 is $O(mK\eta\ell)$, where $\eta = |\text{con}(T_i)|$.

Algorithm 4: mostAtColumn

Input: con(T_i), hash tables $T^1[1]$, ..., $T^\ell[1]$, ..., $T^1[m]$, ..., $T^\ell[m]$

Output: *most approximation* T_i'

simColIndex[m] = {i, ..., i}; /* Initialize */

for $k \leftarrow 1$ **to** m **do**

simColmnNumArray[|con(T_i)|] = {1, ..., 1};

maxNum ← 1;

for $j \leftarrow 1$ **to** |con(T_i)| **do**

simColumSet ← ∅ ;

simColumnSet ←findSimilar($G_q(T_j[k])$, con$^K(T_j)$);

merge(simColumnNumArray, simColumnSet);

end

simColIndex[k] ← index of simColumnNumArray entry with largest value ;

end

for $k \leftarrow 1$ **to** m **do**

$T_i'[k] \leftarrow T_{\text{simColIndex[k]}}[k]$;

end

return T_i';

This policy can extract a *most approximation* with regard to E quickly.

Extracting Most Approximation at Record Level

To extract a more precise *most approximation*, vote-fusion is performed at record level. That is to say, *most approximation* of T_i consists of all its records' *most approximation*s, which are extracted from con(T_i).

Step 1: Preprocessing to Store Each Record in Buckets Basedon its LSH ***Signatures***

Each data source, say T_l ($1 \le l \le \eta$), is scanned record by record and LSHHash algorithm is applied to map its every record onto buckets of hash tables. Note here every record's q-gram multiset $G_q(r)$ is weighted union of all its field's q-gram multisets. At the end of this step, the bucket of T_l^j ($1 \le j \le \ell$) stores the records from data source T_l that agree with $g_j(.)$.

Step 2: Extracting Most Approximation

The key problem is to extract each record's *most approximation*, which further forms *most approximation* of T_i. The task falls into two core procedures. The first procedure is to identify each record's micro-context $\text{con}_{\text{micr}}(r)$ defined as definition 5, which contains at most $K\eta$ similar records. The second procedure is to vote record's *most approximation* in its micro-context.

Definition 5*:* $\text{con}_{\text{micr}}(r) = \Psi_1 \cup \Psi_2, ..., \cup \Psi_j, ..., \cup \Psi_{|\text{con}(Ti)|}(r \in T_i)$, where Ψ_j is r's top K Nearest Neighbor(KNN) in context

data source T_j .

Note here we choose top K nearest neighbors, rather than nearest neighbor, in each data source as members of micro-context. This is due to that r's nearest neighbor in each data source is not necessarily the best candidate of *most approximation*. We should not exclude potential candidates so early that they have no chance to be fairly selected in later vote-fusion. To quickly identify micro-context members, KNN search problem can be relaxed to approximate KNN search problem, namely (ρ_1, ρ_2, λ, γ) problem, as follows.

Definition 6*:* Given a set of points in set space Γ, a pair of points (u, v) satisfying sim(u, v) $\geq \rho_1$ is mapped onto the same bucket with probability at least λ, and a pair of points (u, v) satisfying sim(u, v) $< \rho_2$ is mapped onto the same bucket with probability at most γ.

In above definition four user-defined parameters are introduced:ρ_1, ρ_2, λ, γ. ρ_1 and ρ_2 are similarity thresholds. λ and γ are used to control the precisions that similar points are mapped onto the same bucket. During mapping, we need to set parameters κ and ℓ to satisfy user-defined precision requirement λ and γ. Parameters κ and ℓ can be determined by theorem 2 and 3 and the proof is not described due to space.

Theorem 2*:* For (ρ_1, ρ_2, ρ_1, ρ_2) family, a pair of points (u,v) satisfying sim(u, v) $\geq \rho_1$ is mapped onto the same bucket with probability at least λ by setting $\ell \geq \lceil \ln(1-\lambda)/\ln(1-\rho_1^{\,k}) \rceil$.

Theorem 3*:* For (ρ_1, ρ_2, ρ_1, ρ_2) family, there exists k_0. A pair of points (u, v) satisfying sim(u, v) $\geq \rho_1$ is mapped onto the same bucket with probability at least λ, and a pair of points (u, v) having sim(u, v) $< \rho_2$ is mapped onto the same bucket with probability at most γ by setting $\kappa \geq \kappa_0$ and $\ell = \lceil \ln(1-\lambda)/\ln(1-\rho_1^{\,k}) \rceil$.

By retrieval of hash tables, a point u's approximate KNN can be obtained with Alg.5. Clearly time complexity of Alg.5 is

Algorithm 5: findApprKNN

Input: A query point $u \in \Gamma, T^1, ..., T^\ell$

Output: K (or less) approximate nearest neighbors Ψ

$\Psi \leftarrow \varnothing$;

for $j \leftarrow 1$ **to** ℓ **do**

map u based on $g_j(.)$;

$\Psi \leftarrow \Psi \cup$ points found in buckets $g_j(u)$ of hash table T^j ;

end

return K nearest neighbors of u in set Ψ; /*By linear search in main memory*/

Algorithm 6: recordMicroContext

Input: $G_q(r)$, hash tables $T_1^{\,1}$, ..., $T_1^{\,\ell}$, ..., $T_\eta^{\,1}$, .., $T_\eta^{\,\ell}$ (here $\eta = |\mathrm{con}(T_i)|$)

Output: $\mathrm{con}_{micr}(r)$

$\mathrm{con}_{micr}(r) \leftarrow \varnothing$;

foreach $i = 1, ..\eta$ **do** $\mathrm{con}_{micr}(r) \leftarrow \mathrm{con}_{micr}(r) \cup$ findApprKNN($G_q(r)$, $T_i^{\,1}$, ..., $T_i^{\,\ell}$);

end

return $\mathrm{con}_{micr}(r)$;

bounded by O(ℓ). Further, record's micro-context is identified with Alg.6. We can see that the worst case of its computational cost is bounded by O($\eta\ell$). After record's micro-context is identified, *most approximation* of record is selected by vote in its micro-context with Alg.7. The intuition behind Alg.7 is that among all the members in $\mathrm{con}_{micr}(r)$, the member whose number of similar ones is largest, is chosen as *most approximation*. Thus, data source's *most approximation* can be extracted at record level as Alg.8. Suppose size of one data source is N. As recordMicroContext algorithm requires O($\eta\ell$) time and voteAtRecord

Algorithm 7: voteAtRecord

Input: $r \in T_i$, θ_{micr}, $\mathrm{con}_{micr}(r)$

Output: r'

simNumArray[|$\text{con}_{micr}(r)$|]={ 0,..,0}; /* Initialize*/

foreach $r_i \in \text{con}_{micr}(r)$ **do**

foreach $r_j \in \text{con}(T_{father}(r_i))$ **do**

if $\text{sim}(r_i, r_j) \geq \theta_{micr}$ **then** simNumArray[i]++;

end

end

$r' \leftarrow$ the record in $\text{con}_{micr}(r)$ with the largest counter in simNumArray;

return r';

Algorithm 8: mostAtRecord

Input: T_i, θ_{micr}, hash tables T_1^1 , ..., T_1^ℓ , ..., T_η^1 , .. , T_η^ℓ

Output: T_i'

$rid \leftarrow 0$;

while *not eof of* T_i **do**

rid + +;

$\text{con}_{micr}(r_{rid}) \leftarrow$recordMicroContext($G_q(r_{rid})$, T_1^1 , ..., T_l^ℓ , ..., T_η^1 , .. , T_η^ℓ);

$r'_{rid} \leftarrow$ voteAtRecord(r_{rid}, θ_{micr}, $\text{con}_{micr}(r_{rid})$);

insert r'_{rid} into T_i' ;

end

return T_i';

algorithm requires $O(\eta K^2)$ time, time complexity of algorithm Alg.8 is $O(N\eta(K^2 + \ell))$. Compared with algorithm mostAtColumn, *most approximation* extracted at record level is much more precise because it can distinguish the values from different records while the former can not.

Extracting Most Approximation at Field Level:

In this extraction policy, *most approximation* of every record is a synthetic one, rather than an existing record. In other words, every field of a record's *most approximation* is chosen from all corresponding field values in the record's micro-context. Extraction consists of two core steps as follows.

Step 1: Preprocessing to Store Each Record in Buckets Based on Its LSH ***Signature***

This is the same as the corresponding step when *most approximation* is extracted at record level.

Step 2: Extracting Most Approximation

Most approximation of each record is synthesized as Alg.9. As total number of members in $\text{con}_{micr}(r)$ is at most $K\eta$, time

Algorithm 9: voteAtField

Input: $r \in T_i$, $\theta_{fieldsim}$, $\text{con}_{micr}(r)$

Output: r'

$r'[1..m]$ = {null, ..., null}; /* Initialize */

for $k \leftarrow 1$ **to** m **do**

simNumArray[|$\text{con}_{micr}(r)$|]={0,..,0};/* Initialize */

foreach $r_i[k] \in \text{con}_{micr}(r)$ **do**

foreach $r_j[k] \in \text{con}(T_{father}(r_i))$ **do**

if $\text{sim}(r_i[k], r_j[k]) \geq \theta_{fieldsim}$ **then** simNumArray[i]++;

end

end

$r'[k] \leftarrow$ field value that corresponds to the largest entry of simNumArray;

end

return r'

Algorithm 10: mostAtField

Input: T_i, $\theta_{fieldsim}$, hash tables, T_1^1 , ..., T_l^ℓ , ..., T_η^1 , ..., T_η^ℓ

Output: T_i'

$rid \leftarrow 0$;

while *not eof of* T_i **do**

rid + +;

$\text{con}_{micr}(r_{rid}) \leftarrow$recordMicroContext($G_q(r_{rid})$, T_1^1 , ..., T_1^ℓ , ..., T_η^1 , .. , T_η^ℓ);

$r'_{rid} \leftarrow$ voteAtField(r_{rid}, $\theta_{fieldsim}$, $\text{con}_{micr}(r_{rid})$);

insert r'_{rid} into T_i' ;

end

return T_i' ;

complexity of alg.9 is bounded by $O(mK^2\eta)$. Following this, *most approximation* of data source can be extracted as Alg.10. As the q-grams from different field values of the same record are distinguished, a very precise *most approximation* can be achieved. Time complexity of this algorithm is $O(N\,\eta(mK^2 + \ell))$. As m, K are constants and ℓ is set by precision requirements, running time mainly depends on N and η, namely size of data source and total number of context data sources.

Measuring Data Source's Accuracy

Accuracy of T_i is measured based on JSD between T_i and T_i'. JSD is also computed at column, record and field level respectively and we do not describe them in details.

EXPERIMENTAL EVALUTION

We ran experiments on real and synthetic data sources to evaluate QACM approach.

Real Data Sources. As there is no standard benchmark to evaluate data quality in CIS environment, we collected two groups of real data sources from National University of Singapore. The first group describes a certain year's undergraduates majoring in information system (denoted as E^{is}) and the second group describes a certain year's undergraduates majoring in computer science (denoted as E^{cs}). Specifically, data are from six sources including T_{FC} ,T_{AC}, T_{SSC}, T_{RSC}, T_{CC} and T_{SO}.

Synthetic Data Sources. Synthetic data sources were populated using real data describing community entities, which was extracted from globalcomputing[1]. It acts as E. Twelve synthetic data sources $\{T_1, T_2, \ldots, T_{12}\}$ were produced according to some rules (Kim, 2003).

We have done experiments in terms of effectiveness, running time and effects of LSH technique respectively but here the effects of LSH technique is not reported due to space. The state of the art about quantifying data quality is sampling

[1] http://www.globalcomputing.com

combined with human interaction (Ballou, 2006; Motro, 1996; Parssian, 1999, 2002), while ours is an *automatic* approach to quantify *accuracy* dimension. Hence, we do not compare our approach with them due to its unfairness.

Effectiveness of QACM Approach

Real *accuracy* by comparing T_i with E is denoted as $acc_{real}(T_i)$ and *accuracy* by comparing T_i with T_i' is denoted as $acc(T_i)$.

As in cooperative multi-source environment people are more interested in top n data sources ranked by *accuracy*, we adopt p@n to measure effectiveness of our approach (Ricardo, 1999).

Effectiveness on Real Data Sources:

We evaluated effectiveness on two groups of data. It showed that for each group the former five data sources are highly

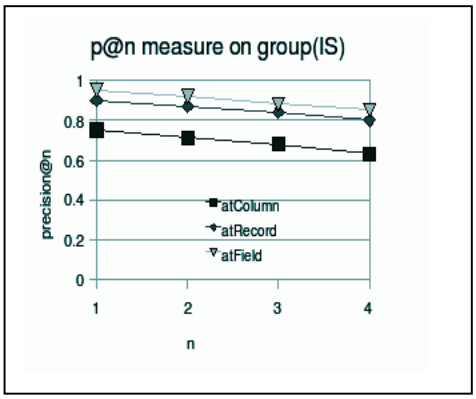

Figure1. P@n measure on group(is)

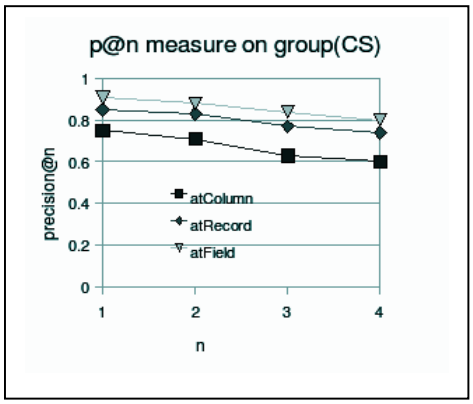

Figure. 2. P@n measure on group(cs)

overlapped and the last data source has a low relevance to other data sources. Thus, we chose former five data sources as *context*. *Most approximation* of each data source is extracted by mostAtColumn, mostAtRecord and mostAtField within *context* respectively. We varied n from 1 to 4. Fig.1 plots the p@n measure for E^{is} group and Fig.2 plots the p@n measure for E^{cs} group. They show p@n measure obtained at more refined level is always more precise. This is due to its more precise *most approximation* extracted.

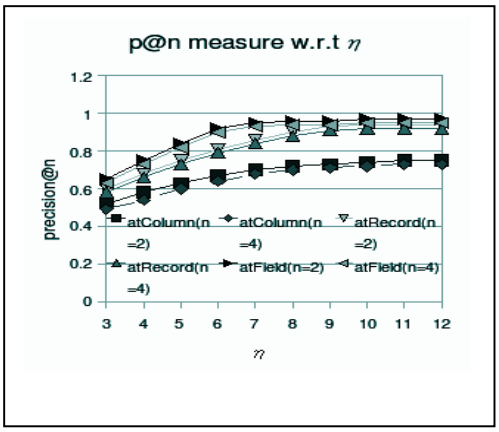

Figure. 3. P@n measure w.r.t total context number

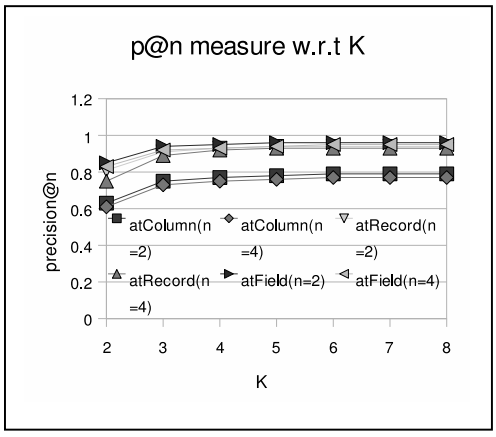

Figure. 4. P@n measure w.r.t K

Effectiveness on Synthetic Data Sources:

We varied parameters of interest on synthetic data sources to test its effect.

Effect of Total Context Data Source Number. We evaluate how total context data source number affects p@n measure. We fixed $\ell = 10$, $\kappa = 12$ and varied total context data source number from 3 to 12 by adjusting parameter θ_{con}. Size of every synthetic data source is fixed to 30 000. We tested how p@n measure changed with respect to context data source number by

exploring different values of n from 1 to 4. We only report the result in Fig.3 by setting $n = 2$ and $n = 4$ due to space. For each n, p@n measure increases first rapidly then slowly with the increase of total context data source number. Furthermore, marginal improvement of p@n measure is approaching zero when total context data source number is beyond a certain threshold. This is due to that when total context data source number is small, every newly added context data source can remarkably correct q-gram distribution while this effect is becoming less and less when total context data source number becomes larger. The figure also illustrates that the extraction policy at coarser level is less sensitive to total context data source number than the extraction policy at more refined level. This is mainly due to that different extraction policies behave differently in distinguishing data distributions.

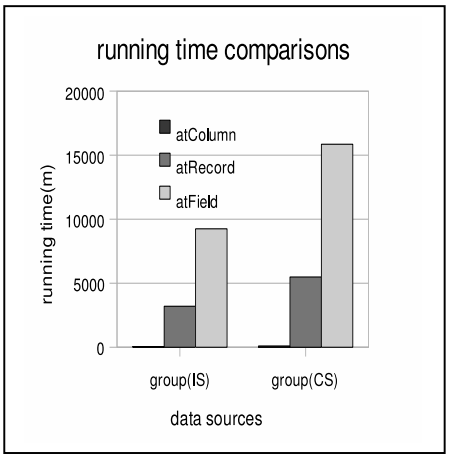

Figure. 5. Running time of different extraction policies

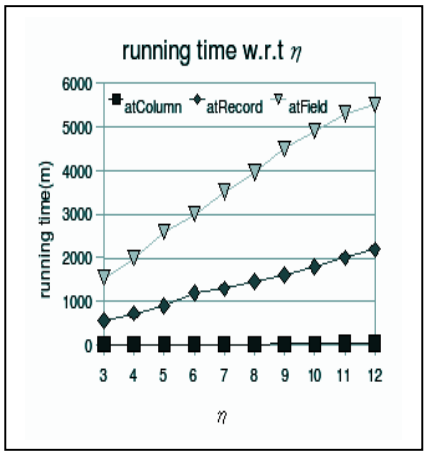

Figure. 6. Running time w.r.t total context data source number

Effect of K. Actually, parameter K serves two functions. First, in selecting candidates of *most approximation* in each data source, it bounds the number of candidates from one data source. Second, it also limits the number of members each object (column, record or field) is supported during vote. We fixed $\eta = 8$, $\ell = 10$ and $\kappa = 12$. We have explored the results of p@1,p@2,p@3 and p@4 by varying K from 2 to 8. We only report results of p@2 and p@4 measures in Fig.4 due to space. We can observe that the increase of p@n measure is rapidly becoming smaller with the increase of K. The increase is approaching zero when K exceeds 4. In fact, if K is too small, some potential best candidates may be excluded too early. Hence vote policy can not work well. However, if K is too large the marginal improvement of effectiveness is also negligible.

Running Time of QACM Approach

In QACM the core procedure is to extract *most approximation* among *context*. Therefore, we evaluate running time of different extraction policies on real and synthetic data sources. Every extraction algorithm was run 10 times to get its average to reduce the side effect resulting from hardware or software.

Running Time of Different Policies

We fixed $\eta = 8$, $\ell = 10$, $\kappa = 12$ and $K = 3$. Fig.5 shows the running time on E^{is} and E^{cs} respectively. Compared with the running time taken by mostAtRecord and mostAtField, the running time taken by mostAtColumn is almost negligible because total number of accessing hash tables in last policy is only a very small fraction of that in former two policies. Both mostAtField and mostAtRecord scan data source record by record, but total number of accessing hash tables in the former is $(m - 1)N\eta K^2$ times larger than that involved in the latter. This is verified in the graphs. In view of different p@n measures of three policies, users should consider both effectiveness and running time cost in choosing a policy.

Scalability Evaluation:

We evaluate scalability with respect to the parameters of interest.

Scalability in Total Context Data Source Number. We fixed K= 3, ℓ= 10 and size of each data source to be 30 000. We varied total context data source number from 3 to 12. As shown in Fig.6, running time scales almost linear with total number of context data sources. This would be better for user experience, since our QACM approach can scale well with the increase of total data source number.

Scalability in Size of Each Data Source. We evaluate on synthetic data sources to test how size of each data source affects running time. We fixed $K = 3$ and $\eta = 8$ and varied size of each data source from 10 000 to 300 000 to test scalability of our QACM approach. Fig.7 plots the running time with different size of each data source. We observe that *mostAtColumn* is not sensitive to size of each data source and running time of extraction policies at record and field level increases linearly with size of each data source. This is our desirable property.

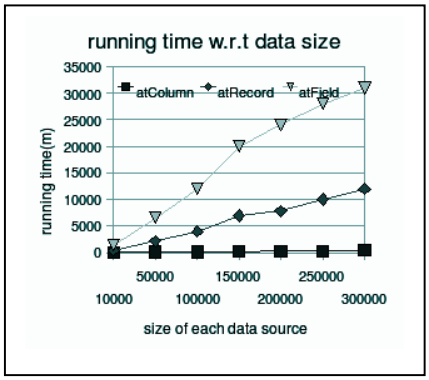

Figure. 7. Running time w.r.t size of each context data source

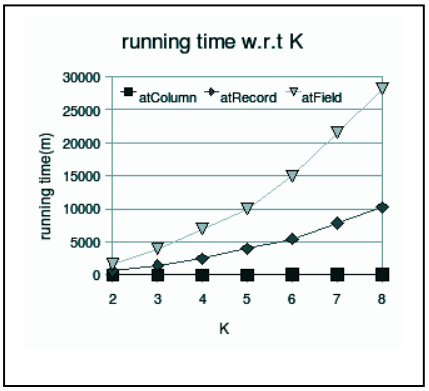

Figure. 8. Running time w.r.t K

Scalability in *K*. We also fixed $\eta = 8$, $\ell = 10$, $\kappa = 12$ and size of each data source to be 30 000. Fig.8 shows the result by varying K from 2 to 8. We observed that running time increases linearly with K for extraction policy at column level. However, running time increases almost quadratic with K for extraction polices at record or field level. Fortunately, considering that in practice a small K is enough. Actually, K is at most total context data source number and it can not be too large. Hence, we do not need to worry about its effect on running time.

CONCLUSION AND DISCUSSION

Accuracy dimension is a pressing care in many data quality scenarios such as data integration, information retrieval, etc. we propose a novel automatic approach to quantify *accuracy* dimension in multiple-sources environment. Note that although our QACM approach aims at multiple-source environment but it is not only limited to this kind of environment as long as context members can be obtained. For example, in world wide web our scheme can also be used to evaluate whether one web page or web site is accurate or not with the help of other techniques such as information extraction, information retrieval, etc. In sum, the contributions of this paper can be summarized as follows.

- As *accuracy* of data can be gauged by discrepancy between data and its entity's *perfect representation* and *perfect representation* is difficult to obtain, we propose that *accuracy* of data source can be feasibly gauged by comparing data source with its *most approximation*. This is the key of quantifying accuracy *automatically*.
- *Accuracy* measure is defined based on Jensen-Shannon Divergence (JSD). This is a universal measure to quantify *accuracy* for different types of data.
- Using LSH technique, our QACM approach can quickly extract *most approximation* of data source at different levels, namely column, record and field level, which scales well with total number of data sources and size of each data source.

To sum up, automatic data quality assessment is a challenging problem until recently and in this paper we address how to quantify *accuracy* dimension. There is still much work to do in future. For example, how to extend our approach to other data quality dimensions and how to assess *accuracy* in terms of semantics are also our focus.

ACKNOWLEDGMENTS

We thank the foundation given by National Natural Foundation of China under Grant. No 60573164 and NY207136 of Nanjing University of Posts and Telecommunications. We also sincerely thanks the members of database research group in Southeast University, China for their valuable work and the useful suggestions given by Divesh Srivastava from AT&T.

REFERENCES

1. D.Aebi, L.Perrochon.（1993） Towards improving data quality, in:Proceedings of the international conference on information systems and management of data, October, pp.273-281.

2. S.Bergamaschi, S.Castano, M.Vincini.（1999） Semantic Integration of Semistructured and Structured Data Sources, SIGMOD record 28(1) 54-49.
3. D.P.Ballou, I.N.Chengalur-Smith, R.Y.Wang.（2006） Sample-based quality estimation of query results in relational database environments, IEEE transactions on knowledge and data engineering 18(5) 639-650.
4. M.Bouzeghoub, V.Peralta.（2004） A framework for analysis of data freshness, in: Proceedings of the 2004 international workshop on Information quality in information systems, pp.59-67.
5. P.Bertolazzi, M.Scannapieco.（2001）introducing data quality in a cooperative context, in:Proceedings of the 6th international conference on information quality, pp.431-444.
6. C.Cappiello, C.Francalanci, B.Pernici.（2004） data quality assessment from the user's perspective, in: Proceedings of the 2004 international workshop on Information quality in information systems, pp.68-73.
7. T.Dasu, T.Johnson, S.Muthukrishnan, V.Shkapenyuk.（2002） Mining database structure; or, how to build a data quality Browser, in:Proceedings of the 2002 ACM SIGMOD international conference on Management of data,pp.240-251.
8. Evoke,Available from http://www.evokesoft.com.
9. K.Kukich.（1992）Technique for automatically correcting words in text, ACM Computing Surveys 24(4) 377-439.
10. Z.Majkic.（2004） A general framework for query answering in data qualitybased cooperative information systems, in:Proceedings of the 2004 international workshop on Information quality in information systems, pp.44-50.
11. P.Missier, S.Embury, M.Greenwood.（2006） Quality views: capturing and exploiting the user perspective on data quality, in:Proceedings of the 32nd International Conference on Very Large Data Bases, pp.977-988.
12. A.Motro, I.Rakov. （1996）Estimating the quality of data in relational databases, in: Proceedings of the 1996 Conference on Information Quality, pp.94-106.
13. M.Mecella, M.Scannapieco, A.Virgillito, R.Baldoni, T.Catarci, C.Batini.（2002）Managing data quality in cooperative information systems.Lecture Notes In Computer Science 2519, pp.486-502
14. K.Orr.（1998） data quality and systems theory, Communications of the ACM 41(2) pp.66-71.
15. A.Parssian, S.Sarkar, V.S.Jacob.（1999） assessing data quality for information products, in:Proceeding of the 20th international conference on Information Systems, pp.428-433.
16. A.Parssian, S.Sarkar, V.S.Jacob.（2002） Assessing information quality for the composite relational operation joins, in:Proceedings of the seventh international conference on information quality, pp.225-236.
17. L.Palopoli, G.Terracina, D.Ursino.（2000） The system dike: Towards the semiautomatic synthesis of cooperative information systems and data warehouses, In:Proceedings of the 2000 ADBIS-DASFAA Symposium on Advances in Databases and Information Systems, pp.108-117.
18. M.Scannapieco, A.Virgillito, C.Marchetetti,M.Mecella, R.Baldoni.（2004） The DaQuinCIS architecture: a platform for exchange and improving data quality in cooperative information systems, Information systems 29 pp.551-582.
19. R.Y.Wang.（1998） A product perspective on total data quality management, Communications of the ACM 41(2) 58-65.
20. R.Y.Wang, H.B.Kon, S.E.Madnick.（1993）Data quality requirements analysis and modeling, in:Proceedings of the Ninth International Conference on Data Engineering, pp.670-677.
21. Y.Wand, R.Y.Wang.（1996） Anchoring data quality dimensions in ontological foundations, Communications of the ACM 39(11) pp.86-95.
22. Aristides Gionis,Piotr Indyk, Rajeev Motwani.（1999） similarity search in high dimensions via hashing. in: Proceedings of the 25th international conference on very large databases, pp.518-529.
23. Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk.（2001） Finding interesting associations without support pruning. IEEE transactions on knowledge and data engineering 13(1) pp.64-78.
24. P.Indyk,R.Motwani.（1998） approximate nearest paper neighbor:towards removing the curse of dimensionality. in: Proceedings of the thirtieth annual ACM symposium on theory of computing, pp.604-613.
25. Taher H.Haveliwala, Aristides Gionis, Piotr Indyk.（2000） Scalable techniques for clustering the web. in: Proceedings of the 3rd international workshop on web and databases, pp.129-134
26. Won Kim.（2003） A taxonomy of dirty data. data mining and knowledge discovery,7(1),pp.81-99.
27. Ricardo Baeza-Yates, Berthier Ribeiro-Neto.（1999） Modern information retrieval.Addison Wesley Inc.
28. Fei Chiang, Renee J.Miller.（2008） Discovering data quality rules. in: proceedingof 34th international conference on verr large databases.