Reasoning about Attribute Value Equivalence in Relational Data

Fengfeng Fan #1, Qun Chen #2, Zhanhuai Li #3, Lei Chen #4

# School of Computing, Northwestern Polytechnical University
Xi’an, China
{fanfengfeng@mail.nwpu, chenbenben@nwpu, lizhh@nwpu}.edu.cn
* Dept of Computer Science & Engineering, Hong Kong University of Science and Technology
Hong Kong, China
4 leichen@cse.ust.hk

Abstract—Identifying the attribute values that describe the same real-world entity but have different representations in relational data is an essential task for many data cleaning and mining applications (e.g., duplicate record detection and functional dependency mining). The state-of-the-art approaches for attribute value matching are mainly based on string similarity of attribute values. The effectiveness of these approaches depends on the assumption that equivalent attribute values appear similar while in comparison, non-matchers appear less similar. However, these approaches may not perform well in the circumstances where the specified string similarity metric is not a reliable indicator for attribute value equivalence.

To alleviate the limitation of the existing string-based approaches, in this paper, we propose a new framework for attribute value matching in relational data. We first propose a novel probabilistic approach to reason about attribute value equivalence by value correlation analysis. Unlike the existing string-based approaches, our approach does not directly compare the contents of two targeted attribute values but reasons about their equivalence by analysing their correlation with other attribute values. We also propose effective methods for probabilistic equivalence reasoning with multiple attributes. Next, we present a unified framework which takes both string similarity measurement and value correlation analysis into consideration. Specifically, it exploits both analytical approaches and provides with a unified estimation by evidential reasoning. To make our approach scalable to huge amount of the data, we then develop efficient and easily parallelizable computation algorithms and implement them on the popular MapReduce programming paradigm. Finally, we demonstrate the effectiveness of the unified framework empirically on real datasets. Through extensive experiments, we show that our framework outperforms the string-based approaches by considerable margins on matching accuracy and achieves the desired efficiency.

Index Terms—Relational Data, Attribute Value Equivalence, String Similarity Measurement, Relational Value Correlation Analysis.

1 INTRODUCTION

An attribute value may have different representations in a relational database. Variations in representation can arise from differences in storage formats, typographical errors, aliases and abbreviations. Determining attribute value equivalence is an essential task for many relational data cleaning and mining applications [5], [51]. For instance, most techniques for duplicate record detection in relational data [18], [50], [35] divide each record into fields (attributes) and identify duplicate records by comparing their values on fields. Effective attribute value matching can therefore improve the accuracy of duplicate record identification. Functional dependency and conditional functional dependency mining [11], [21] also requires attribute value matching to reduce noise: non-identical but equivalent attribute values could make a valid functional dependency elusive.

Since attribute values are usually stored as strings, the state-of-the-art techniques for attribute value matching [39], [40], [45], [29], [32], [13], [26], [25] are often based on string similarity of attribute values. They estimate the probability of two attribute values being equivalent by directly comparing their string contents. Depending on the value domain, the similarity patterns of equivalent strings can vary significantly. Therefore, effective matching usually requires a metric to accommodate the value representation variations specific to a domain. The existing string similarity metrics have been shown to be effective in various applications. However, they also have the fundamental limitation: a metric tuned and tested on previous matching problems can perform poorly on new and different matching problems. Even though researchers have proposed adaptive algorithms [9], [46] that can learn similarity metrics automatically, the difficulty of using these methods cannot be overlooked: they require significant training data and intensive human intervention. Provided with a new matching problem, it remains challenging to design a string similarity metric that can effectively capture the value representation variations present in the problem.

We illustrate the limitation of the string-based approach by the example as shown in Table 1. The relational records refer to research papers and each paper has four attributes, title, author, journal, year, which describe the title, the authors, the publication venue and the publication year of the paper respectively. It can be observed that the journal values “Computers” and “Computer” look very similar but actually represent different publication venues. In contrast, the journal values “Journal on Very Large Data Bases” and “VLDB J” appear much less similar but actually refer to the same research journal. To alleviate the
limitation of the string-based approach, we propose to reason about attribute value equivalence by value correlation analysis. In Table 1, we can observe that the two title values, which are correlated with “Journal on Very Large Data Bases” and “VLDB J” respectively, have a common keyword “database”, and similarly, the author values correlated with them share a common author “K. Ramamritham”. Generally, we have the observation that the papers published in the same journal have the higher probability to be in a same research area than those published in different journals. Accordingly, with a higher probability, they share some author names and their titles share some common keywords. As a result, the correlation analysis between two journal values and their corresponding author and title values can provide us with useful clues for equivalence reasoning. More specifically, if two journal values are correlated with many common author values and many highly similar title values, it can be reasoned that they have a high probability to be equivalent.

Note that a simple type of correlation among attribute values can be described by functional dependency, which specifies that the value of one attribute uniquely determines the value at another attribute. Obviously, functional dependency, if it exists, can be exploited to match two attribute values. In the example shown in Table 1, suppose that each paper has a unique title. Accordingly, we have the functional dependency, title → journal. As a result, two attribute values on journal can be determined to be equivalent if their corresponding records have the same value at the attribute title. Unfortunately, in practice, it is challenging to detect a clear-cut functional dependency in the presence of non-identical but equivalent attribute values, and even if it can be successfully detected, it may be of limited use in determining equivalence due to lack of relevant data. Again, in the example shown in Table 1, if each record refers to a unique paper, the functional dependency, title → journal, is then powerless in determining attribute value equivalence at journal because there do not exist two papers sharing a common title.

We observe that besides string similarity measurement, value correlation analysis can also be effective in reasoning about equivalence among attribute values. Therefore, in this paper, we propose a new framework that can leverage both of them. It consists of two components, which correspond to these two analytical approaches respectively. Our major contributions can be summarized as follows:

1) We present a novel probabilistic approach to estimate the probability of attribute value equivalence by value correlation analysis. Unlike the existing string-based approach, our approach does not directly compare the contents of two attribute values but reasons about their equivalence by analysing their correlation with other attribute values. It represents a new perspective on matching attribute values in relational data.

2) We propose effective methods for probabilistic equivalence reasoning by multiple attributes. We model the problem of probabilistic equivalence reasoning as a classification problem, and propose an analytical approach for attribute selection and their estimation combination based on the ensemble theory.

3) We propose a unified framework for attribute value matching in relational data. Building on both string similarity measurement and value correlation analysis, it provides with a unified equivalence estimation by evidential reasoning. The proposed framework is a unified one in the sense that it can be simplified into a pure string similarity metric by setting the evidence weight of value correlation analysis to be 0.

4) We develop efficient and easily parallelizable algorithms for the probabilistic estimation approach by value correlation analysis.

5) We experimentally evaluate the performance of the proposed framework on real datasets available in open-source. Our extensive experiments validate the effectiveness of the unified framework, demonstrating that it outperforms the string-based approaches by considerable margins on matching accuracy and achieves the desired efficiency.

The rest of this paper is organized as follows: Section 2 defines the problem of attribute value matching and briefly presents the existing string-based approaches. Section 3 proposes the probabilistic models to estimate equivalence by value correlation analysis and presents the methods for probabilistic equivalence reasoning with multiple attributes. Section 4 describes the unified framework. Section 5 presents the algorithms for value correlation analysis. Section 6 empirically evaluates the performance of the proposed framework. Section 7 reviews related work. Finally, Section 8 concludes this paper with some thoughts on future work.

### 2 Background

#### 2.1 Problem Definition

Given a relational table $R$, we denote its attributes by capital letters, $\{A, B, \ldots\}$, and represent their corresponding attribute values by Greek letters with subscripts, $\{\alpha, \beta, \ldots\}$. For simplicity of presentation, we summarize the used symbols and their meanings in Table 2.

<table>
<thead>
<tr>
<th>title</th>
<th>author</th>
<th>journal</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority assignment in real-time active databases</td>
<td>R.M. Sivasankaran, J.A. Stankovic, D. Towsley, B. Purimeta, K. Ramamritham</td>
<td>Journal on Very Large Data Bases</td>
<td>2003</td>
</tr>
<tr>
<td>A taxonomy of correctness criteria in database applications</td>
<td>K. Ramamritham, P.K. Chrysanthis</td>
<td>VLDB J</td>
<td>2002</td>
</tr>
<tr>
<td>Precise and efficient groundness analysis for logic programs</td>
<td>K. Marriott, H. Søndergaard</td>
<td>acm letters on programming languages and systems</td>
<td>2005</td>
</tr>
<tr>
<td>Semantics of constraint logic programs with optimization</td>
<td>K. Marriott, P.J. Stuckey</td>
<td>loplas</td>
<td>2008</td>
</tr>
</tbody>
</table>
Each non-null value $\alpha_i$ on the attribute $A$ refers to a member of some set of real-world entities. Two attribute values are deemed to be equivalent, denoted by $\alpha_i \equiv \alpha_j$, iff they refer to a same entity. For instance, in Table 1, the values “VLDB J” and “Journal on Very Large Data Bases” are equivalent because they refer to the same publication venue.

The problem of attribute value matching in relational data is defined as follows:

**Definition 1.** Given two attributes, $A_1 \in R_1$ and $A_2 \in R_2$, of relations $R_1$ and $R_2$, and two sets of attribute values, $\alpha_1 \in A_1$ and $\alpha_2 \in A_2$, of attributes $A_1$ and $A_2$. The problem of attribute value matching is to identify all correspondences between attribute values in $\alpha_1 \times \alpha_2$ representing the same real-world entity. The definition includes the special case of finding pairs of equivalent attribute values within a single relation ($A_1 = A_2$, $R_1 = R_2$).

Attribute value matching is usually performed by pairwise comparisons. It ranks the pairs of attribute values by their equivalence probabilities. Therefore, the core challenge is to compute the equivalence probability between two given attribute values.

## 2.2 String-based Approach

The state-of-the-art techniques for attribute value matching directly measure string similarity between two attribute values. The higher the similarity, the higher the possibility two attribute values are equivalent. Since our unified framework incorporates string similarity measurement as one of its key components, here we briefly examine several effective and widely used metrics. In general, string similarity metrics can be broadly classified into three categories: character-based, token-based and hybrid.

### 2.2.1 Character-based Metrics

The character-based metrics treat an attribute value as a long string. The most typical one is **edit distance**. The edit distance between two strings $\alpha_1$ and $\alpha_2$ is defined to be the minimum number of edit operations of single character needed to transform the string $\alpha_1$ into $\alpha_2$. There are three types of edit operations: (1) insert a character into the string; (2) delete a character from the string; and (3) replace one character with a different character. In the simplest form, each edit operation costs 1. This version of edit distance is also referred to as the Levenshtein distance [34].

The Needleman-Wunsch distance [41] is a natural extension to Levenshtein distance that introduces additional parameters defining each possible character replacement cost and the cost of insertions and deletions. The affine gap cost [49] is another widely used variation that introduces two costs for insertion: one for inserting the first character and a second (usually lower) for inserting additional characters.

The Jaro metric [29] is another effective character-based similarity metric. It is based on the number and order of common characters between two strings. Given two strings, $\alpha_1 = b_1b_2\ldots b_m$ and $\alpha_2 = c_1c_2\ldots c_n$, a character $b_i$ in $\alpha_1$ is defined to be in common” with $\alpha_2$ iff there is a $c_j = b_i$ in $\alpha_2$ such that $(i - L) \leq j \leq (i + L)$, where $L = \min(m, n) / 2$. Let $\alpha'_1 = b'_1b'_2\ldots b'_p$ be the string consisting of the characters in $\alpha_1$ that are common with $\alpha_2$ (in the same order they appear in $\alpha_1$), and let $\alpha'_2$ be analogous. A transposition for $\alpha'_1$ and $\alpha'_2$ is defined to be a position $i$ such that $b'_i = c'_i$. Let $t$ be the number of transpositions for $\alpha'_1$ and $\alpha'_2$. The Jaro distance metric for $\alpha_1$ and $\alpha_2$ is defined to be:

$$Jaro(\alpha_1, \alpha_2) = \frac{1}{3} \left[ \frac{|\alpha'_1|}{|\alpha_1|} + \frac{|\alpha'_2|}{|\alpha_2|} + \frac{|\alpha'_1 \setminus \alpha'_2|}{|\alpha_1| \setminus |\alpha_2| / 2} \right]$$

### 2.2.2 Token-based Metrics

The token-based metrics convert the strings to token multisets and consider similarity metrics on these multisets. They are usually more effective than the character-based ones in the circumstances where word order is unimportant. One simple token-based metric is **Jaccard similarity**. Between two word sets $\alpha_1$ and $\alpha_2$, Jaccard similarity is simply $(|\alpha_1 \cap \alpha_2| / |\alpha_1 \cup \alpha_2|)$.

[13] described a system named WHIRL that adopts from information retrieval cosine similarity combined with the tf-idf weighting scheme to compute similarity between two word sets. Each word $w$ in the attribute value $\alpha_i$ is assigned a weight $V(w, \alpha_i) = \log(tf_w + 1) \cdot \log(idf_w)$, where $tf_w$ is the number of times that $w$ appears in $\alpha_i$ and $idf_w$ is $\log \frac{|R|}{|\alpha_i|}$, where $|\alpha_i|$ is the number of attribute values in $R$ that contain $w$. The cosine similarity of two strings, $\alpha_1$ and $\alpha_2$, is defined as

$$sim(\alpha_1, \alpha_2) = \frac{\sum_{w} V(w, \alpha_1) \cdot V(w, \alpha_2)}{\sqrt{\sum_{w} V(w, \alpha_1)^2} \cdot \sqrt{\sum_{w} V(w, \alpha_2)^2}}$$

### 2.2.3 Hybrid Metrics

Similar to the token-based metrics, the hybrid metrics convert the strings to token multisets. But it also considers token similarity in measuring the similarity between these multisets. [14] proposed the following recursive matching scheme for comparing two long strings $\alpha_1$ and $\alpha_2$. First, two strings $\alpha_1$ and $\alpha_2$ are broken into tokens, $\alpha_1 = b_1 \ldots b_n$ and $\alpha_2 = c_1 \ldots c_m$. Then, their similarity is defined as

$$sim(\alpha_1, \alpha_2) = \frac{1}{n} \sum_{i=1}^{n} \max_{j=1}^{m} sim'(b_i, c_j)$$

where $sim'()$ is some secondary character-based similarity metric between tokens.

A “soft” version of the TF-IDF metric [14], which considers similar tokens, is also a hybrid metric. Again let $sim'$ be a secondary similarity metric between tokens. Let $C(\theta, \alpha_1, \alpha_2)$ be the set of tokens $w \in \alpha_1$ such that there is some token $v \in \alpha_2$ such that $sim'(w, v) \geq \theta$. For $w \in C(\theta, \alpha_1, \alpha_2)$, let $D(w, \alpha_2) = \max_{v \in \alpha_2} sim'(w, v)$. Then, the soft TF-IDF metric is defined as

$$sim(\alpha_1, \alpha_2) = \sum_{w \in C(\theta, \alpha_1, \alpha_2)} V(w, \alpha_1) \cdot V(w, \alpha_2) \cdot D(w, \alpha_2)$$

## 3 Probabilistic Correlation Analysis

In this section, we first present a basic model in Subsection 3.1 which exploits only identical values and reasons about value equivalence on a target attribute based on another attribute. Next,
in Subsection 3.2, we propose an extended model that incorporates string similarity metrics into the process of equivalence reasoning. Finally, in Subsection 3.3, we discuss how to reason about value equivalence on a target attribute based on multiple attributes.

3.1 Basic Model

Given two non-identical values \( \alpha_1 \) and \( \alpha_2 \) at the target attribute \( A \), the basic model uses their correlated values at another attribute \( B \) to estimate their equivalence probability. As shown in Table 2, \( B(\alpha_i) \) denotes the set of attribute values at \( B \) of the tuples in \( R(\alpha_i) \). For convenience of presentation, we assume that each value in the set denoted in Table 2 corresponds to a tuple in \( R \). Therefore, the sets (e.g., \( B(\alpha_i) \)) may contain duplicate attribute values.

Based on the Bayes’ theorem, the probability of \( \alpha_1 \) and \( \alpha_2 \) being equivalent can be computed by

\[
p(\alpha_1 \equiv \alpha_2) = \frac{p(\alpha_1 \equiv \alpha_2|\beta_1 \equiv \beta_2)}{p(\beta_1 \equiv \beta_2|\alpha_1 \equiv \alpha_2)} \cdot p(\beta_1 \equiv \beta_2)
\]

where \( \beta_1 \in B(\alpha_1), \beta_2 \in B(\alpha_2), p(\beta_1 \equiv \beta_2) \) represents the probability that a value in \( B(\alpha_1) \) is equivalent to a value in \( B(\alpha_2) \), and \( p(\alpha_1 \equiv \alpha_2|\beta_1 \equiv \beta_2) \) and \( p(\beta_1 \equiv \beta_2|\alpha_1 \equiv \alpha_2) \) are conditional probabilities representing the correlation between two attributes \( A \) and \( B \) with respect to \( \alpha_1 \) and \( \alpha_2 \), respectively.

Therefore, the correlation factor between \( A \) and \( B \), \( f(A, B) \), can be computed by

\[
f(A, B) = \frac{p(\alpha_1 \equiv \alpha_2)}{p(\beta_1 \equiv \beta_2)}
\]

where \( p(\alpha_1 \equiv \alpha_2|\beta_1 \equiv \beta_2) \) represents the probability of two values in \( A(R) \) being equivalent, and \( p(\beta_1 \equiv \beta_2) \) represents the probability of two values in \( B(R) \) being equivalent. Denoting \( p(\alpha_1 \equiv \alpha_2) \) and \( p(\beta_1 \equiv \beta_2) \) by \( p(A(R)) \) and \( p(B(R)) \) respectively, we have

\[
f(A, B) = \frac{p(A(R))}{p(B(R))}
\]

The basic model only considers identical values for correlation analysis. It assumes that two attribute values are equivalent if they are identical. Therefore, it estimates \( p(\beta_1 \equiv \beta_2) \) in Equation 1 by \( p(\beta_1 \equiv \beta_2) \). Similarly, it estimates the correlation factor, \( f(A, B) \), by

\[
f(A, B) \approx \frac{p(\alpha_1 \equiv \alpha_2)}{p(\beta_1 \equiv \beta_2)}
\]

Summarizing the above estimations, we have

\[
p(\alpha_1 \equiv \alpha_2) \approx f(A, B) \cdot p(\beta_1 \equiv \beta_2)
\]

Note that the computation of \( f(A, B) \) as shown in Equation 4 requires to retrieve all the \( A \) and \( B \) attribute values in \( R \). Here we propose a scheme, estimation by reference, to make the estimation of \( f(A, B) \) unnecessary for the estimation of \( p(\alpha_1 \equiv \alpha_2) \). It only involves the values in the two sets, \( B(\alpha_1) \) and \( B(\alpha_2) \).

Estimation by Reference. Given two randomly selected tuples from \( R(\alpha_i) \), \( t_i \) and \( t_j \), the probability of their attribute values at \( A \) being equivalent can be estimated by Equation 7 as follows:

\[
p(\alpha_{1i} \equiv \alpha_{1j}) \approx f(A, B) \cdot p(\beta_{1i} = \beta_{1j})
\]

where \( \alpha_{1i} \) and \( \alpha_{1j} \) represent \( t_i \) and \( t_j \)’s attribute values at \( A \), and \( \beta_{1i} \) and \( \beta_{1j} \) represent their attribute values at \( B \). Since \( \alpha_{1i} = \alpha_{1j} = \alpha_1 \), \( p(\alpha_{1i} = \alpha_{1j}) = 1 \). Combining Equations 7 and 8 results in

\[
p(\alpha_1 \equiv \alpha_2) \approx \frac{p(\beta_1 \equiv \beta_2)}{p(\beta_{1i} = \beta_{1j})}
\]

where \( p(\beta_1 \equiv \beta_2) \) represents the probability of a value in \( B(\alpha_1) \) and a value in \( B(\alpha_2) \) being identical, and \( p(\beta_{1i} = \beta_{1j}) \) represents the probability of two values in \( B(\alpha_1) \) being identical.

In Equation 9, the denominator serves as the reference to determine the equivalence between \( \alpha_1 \) and \( \alpha_2 \). The closer \( p(\beta_1 = \beta_2) \) is to \( p(\beta_{1i} = \beta_{1j}) \), the more probably \( \alpha_1 \) and \( \alpha_2 \) are equivalent. Suppose that in the running example, \( \alpha_1 = “VLDB J” \) and \( \alpha_2 = “Journal on Very Large Data Bases” \). Then, \( p(\beta_1 = \beta_2) \) captures the title similarity between the papers published at “VLDB J” and those published at “Journal on Very Large Data Bases”, and \( p(\beta_{1i} = \beta_{1j}) \) captures the title similarity within the papers published at “VLDB J”.

Either of \( B(\alpha_1) \) and \( B(\alpha_2) \) can serve the reference purpose. In practical implementation, we suggest to choose the one with the higher value of \( p(\beta_{1i} = \beta_{1j}) \). We denote the probability of a value in \( B(\alpha_1) \) and a value in \( B(\alpha_2) \) being identical by \( p(B(\alpha_1), B(\alpha_2)) \), and similarly the probability of two values in \( B(\alpha_1) \) being identical by \( p(B(\alpha_1)) \). By reference, the equivalence probability of \( \alpha_1 \) and \( \alpha_2 \) is estimated by

\[
p(\alpha_1 \equiv \alpha_2) \approx \frac{p(B(\alpha_1), B(\alpha_2))}{\max_{i=1,2} p(B(\alpha_i))}
\]
3.2 An Extension of Basic Model

The basic model as presented in Equations 7 and 9 has limited applicability because it only considers identical attribute values. Consider the running example shown in Table 1. To determine the equivalence between two non-identical journal names, we compare their corresponding paper titles. In case that their papers do not share common titles, estimating equivalence probability by the basic model would result in the value of \(\beta_1 = \beta_2\) being equal to 0. However, it can be observed that if two papers are published in the same journal, they may be in the same research area with high probability; accordingly, their titles may also be similar. Therefore, it is important for the probabilistic model to be able to capture the correlation between journal and title as shown in Table 1. To this end, we extend the basic model by incorporating string similarity metrics into the process of value correlation analysis.

The extended model considers not only identical values, but also similar values in computing the probabilities in Equation 1. Previous work on attribute value matching assumes that the probability of two attribute values being equivalent equals to their string similarity. Under such assumption, \(p(\beta_1 \equiv \beta_2)\) in Equation 1 can be estimated by the average string similarity between \(B(\alpha_1)\) and \(B(\alpha_2)\), which is computed by

\[
p(\beta_1 \equiv \beta_2) \approx \frac{\sum \text{sim}(\beta_{1i}, \beta_{2j})}{|B(\alpha_1)| \cdot |B(\alpha_2)|}
\]  

(11)

in which \(\beta_{1i} \in B(\alpha_1)\), \(\beta_{2j} \in B(\alpha_2)\) and \(\text{sim}(\beta_{1i}, \beta_{2j})\) specifies string similarity between \(\beta_{1i}\) and \(\beta_{2j}\). Note that in Equation 11, the numerator exhaustively aggregates the similarity of every value pair between \(B(\alpha_1)\) and \(B(\alpha_2)\) and the denominator represents the total number of computed value pairs.

Similarly, the correlation factor between \(A\) and \(B\), \(f(A, B)\), can be estimated by

\[
f(A, B) \approx \frac{\sum_{\alpha_i, \alpha_j \in A(R)} \text{sim}(\alpha_i, \alpha_j)}{\sum_{\beta_i, \beta_j \in B(R)} \text{sim}(\beta_i, \beta_j)}
\]  

(12)

in which the numerator aggregates the similarity of every value pair in \(A(R)\), and the denominator aggregates the similarity of every value pair in \(B(R)\).

If estimated by reference, the numerator in Equation 10, \(p(B(\alpha_1), B(\alpha_2))\) can be similarly computed by Equation 11. The denominator, \(p(B(\alpha_i))\), can be estimated by the average string similarity of the values in \(B(\alpha_i)\), which is computed by

\[
p(B(\alpha_i)) \approx \frac{\sum_{j,k} \text{sim}(\beta_{1j}, \beta_{2k})}{|B(\alpha_i)| \cdot (|B(\alpha_i)| - 1)}
\]  

(13)

where \(\beta_{1j} \in B(\alpha_i)\) and \(\beta_{2k} \in B(\alpha_i)\) \((j \neq k)\), the numerator exhaustively aggregates the similarity of every value pair in \(B(\alpha_i)\), and the denominator denotes the total number of computed value pairs.

3.3 Estimation by Multiple Attributes

For probabilistic equivalence reasoning based on multiple attributes, a straightforward solution is to combine multiple attribute values into a long field and then treat the resulting string as a single attribute in correlation analysis. However, this approach does not differentiate the uneven importance of different attributes. Using a single string similarity metric, it cannot incorporate specific domain knowledge on individual attributes into the reasoning process either.

To address the limitation of the straightforward solution, we model the problem of probabilistic equivalence reasoning as a classification problem. It classifies each value pair, \(\alpha_i\) and \(\alpha_j\), at a target attribute \(A\) into one of two groups, equivalent and inequivalent. Other attributes in \(R\) (except \(A\)) are considered as the feature sets of the objects (value pairs). Probabilistic reasoning by an individual attribute works as a classifier, which computes for its corresponding feature. Probabilistic reasoning by multiple attributes then corresponds to the problem of building an ensemble of classifiers and combining their classification results.

According to the ensemble theory [36], the performance of an ensemble depends on the performance of its component classifiers and their performance diversity. Therefore, the process of attribute selection for probabilistic reasoning consists of two steps. The first step of candidate filtering chooses the candidate attributes whose corresponding classifiers can achieve good performance. The following step of combination generation selects a combination of attributes from the candidate set with the aim to boost the performance diversity among the chosen attributes.

3.3.1 Candidate Filtering

Given an attribute value \(\alpha_1\) at \(A\), consider two randomly selected tuples in \(R(\alpha_1)\). According to the probabilistic models as presented in Equation 4 and 8, the probability of their attribute values at \(A(\alpha_{1i}\) and \(\alpha_{1j}\)) being equivalent can be estimated by

\[
p(\alpha_{1i} \equiv \alpha_{1j}) \approx \frac{p(\alpha_{1i} \equiv \alpha_{1j})}{p(\beta_i \equiv \beta_j) \cdot p(B(\alpha_i))}
\]  

(14)

where \(\alpha_{1i} = \alpha_{1j} \equiv \alpha_1\), and therefore \(p(\alpha_{1i} \equiv \alpha_{1j})=1\). Note that \(p(\alpha_i \equiv \alpha_j) = p(B(R))\). We define a conditional correlation factor between an attribute \(B\) and an \(A\) value \(\alpha_1\) as

\[
ccf(B, \alpha_1) = \frac{p(B(\alpha_1))}{p(B(R))}
\]  

(15)

where \(p(B(\alpha_1))\) denotes the probability of two values in \(B(\alpha_1)\) being equivalent, and \(p(B(R))\) denotes the probability of two values in \(B(R)\) being equivalent. Therefore, we have

\[
ccf(B, \alpha_1) \cdot p(\alpha_i \equiv \alpha_j) \approx 1
\]  

(16)

In Equation 16, the value of \(p(\alpha_i \equiv \alpha_j)\) is independent of the chosen \(B\). We observe that the higher the value of \(ccf(B, \alpha_1)\) is, the more accurate estimation the probabilistic models can achieve. Therefore, the attributes with strong conditional correlation with \(A\) values should be selected for equivalence reasoning at \(A\). Based on Equation 15, the value of \(ccf(B, \alpha_1)\) is high in the circumstance where two attribute values in \(B(R)\) have low probability of being equivalent, but if their corresponding tuples have the same value at \(A\), their probability of being equivalent becomes much higher. For instance, in the running example shown in Table 1, the probability that two randomly chosen papers have similar titles is low because of the variety of research areas. However, if two papers were published in a same research venue, the probability that their titles are similar would become dramatically higher. In comparison, conditional correlation between journal and year is much weaker. Therefore, the attribute title is a better candidate than year for equivalence reasoning at journal.
Given a target attribute $A$ and another attribute $B$, we estimate the general conditional correlation factor between them by a weighted sum of $ccf(B, \alpha_i)$ as follows:

$$ccf(B, A) = \sum_{\alpha_i \in A(R)} (w_i \cdot ccf(B, \alpha_i))$$  \hfill (17)

in which $\alpha_i$ denotes a distinct value in $A(R)$, $w_i$ denotes the weight and its value is set to be proportional to the occurrence frequency of $\alpha_i$ in $A(R)$.

The process of candidate filtering sets a threshold $\theta_c$ (e.g., $\theta_c = 5$) on the value of conditional correlation factor. An attribute $B$ is included into the candidate set if and only if its conditional correlation factor with regard to $A$ is no smaller than $\theta_c$, $ccf(B, A) \geq \theta_c$. Note that if the $A$ attribute values are independent of their corresponding $B$ values, theoretically the value of $ccf(B, A)$ should be equal to 1. Therefore, $\theta_c$ should be set to be a value larger than 1.

### 3.3.2 Combination Generation

The process of combination generation involves measuring performance diversity among candidate attributes. Given two candidate attributes, $B_1$ and $B_2$, their performance difference on a pair of target $A$ attribute values, $(\alpha_i, \alpha_j)$, $D_{(\alpha_i, \alpha_j)}(B_1, B_2)$ can be computed by

$$|f(A, B_1) \cdot p(\beta_{1i} \equiv \beta_{1j}) - f(A, B_2) \cdot p(\beta_{2i} \equiv \beta_{2j})|$$  \hfill (18)

in which $\beta_{1i} \in B_1(\alpha_i)$, $\beta_{1j} \in B_1(\alpha_j)$, $\beta_{2i} \in B_2(\alpha_i)$ and $\beta_{2j} \in B_2(\alpha_j)$.

According to Equation 4, we have

$$f(A, B_1) = \frac{p(A(R))}{p(B_1(R))}$$  \hfill (19)

and

$$f(A, B_2) = \frac{p(A(R))}{p(B_2(R))}$$  \hfill (20)

Therefore, we have

$$f(A, B_2) = f(A, B_1) \cdot f(B_1, B_2)$$  \hfill (21)

Normalized by the estimation result by $B_1$, the performance difference can be represented by

$$D_{(\alpha_i, \alpha_j)}(B_1, B_2) = |1 - f(B_1, B_2) \cdot \frac{p(\beta_{2i} \equiv \beta_{2j})}{p(\beta_{1i} \equiv \beta_{1j})}|$$  \hfill (22)

We represent the correlation between $B_1(\alpha_i)$ and $B_2(\alpha_j)$ by

$$f(B_1(\alpha_i), B_2(\alpha_j)) = \frac{p(\beta_{1i} \equiv \beta_{1j})}{p(\beta_{2i} \equiv \beta_{2j})}$$  \hfill (23)

Equation 22 can therefore be rewritten by

$$D_{(\alpha_i, \alpha_j)}(B_1, B_2) = |1 - \frac{f(B_1, B_2)}{f(B_1(\alpha_i), B_2(\alpha_j))}|$$  \hfill (24)

Since probabilistic reasoning aims to identify equivalent attribute values, the classifier ensemble is designed to boost the performance diversity of the classifiers on these value pairs. Therefore, we estimate the performance diversity between $B_1$ and $B_2$ by considering the identical value pairs, $(\alpha_i, \alpha_i)$. Denoting $D_{(\alpha_i, \alpha_i)}(B_1, B_2)$ by $D_{\alpha_i}(B_1, B_2)$, we estimate performance diversity between $B_1$ and $B_2$ by

$$D_A(B_1, B_2) \approx \sum_{\alpha_i \in A(R)} w_i \cdot D_{\alpha_i}(B_1, B_2)$$  \hfill (25)

in which $w_i$ represents the weight and its value is proportional to the occurrence frequency of $\alpha_i$ in $A(R)$.

The algorithm for generating attribute combination is sketched in Algorithm 1. It iteratively selects the attribute $B_j$ with the highest conditional correlation factor in the candidate attribute set $S_C$. If there exists an attribute $B_i$ in the result attribute set, $S_F$, such that the performance diversity between $B_i$ and $B_j$ with regard to $A$ as estimated by Equation 25 is below a threshold $\theta_d$ (e.g., 0.3), $D_A(B_i, B_j) < \theta_d$, $B_j$ is filtered out; otherwise, $B_j$ is inserted into $S_F$. The algorithm stops when the candidate set $S_C$ becomes empty.

The algorithms for computing conditional correlation factor and performance diversity between two attributes are presented in Subsection 5.3. Provided that they have been computed beforehand, Algorithm 1 runs in $O(k^2)$ time in worse case, in which $k$ denotes the number of attributes in $R$.

### 3.3.3 Combination Rule

As in other applications of the ensemble theory, the probabilistic estimations computed by the classifiers for individual attributes are combined using the voting method. It first estimates the equivalence probability of $\alpha_1$ and $\alpha_2$ by each chosen attribute and then computes a weighted sum of the estimation results. The weight for an attribute $B_i$ is proportional to the log of the conditional correlation factor between $B_i$ and $A$. Suppose that the chosen $k$ attributes are $\{B_1, \ldots, B_k\}$. The combined probability is estimated by

$$p(\alpha_1 \equiv \alpha_2) = \sum_{i \leq k} w_i \cdot p_i(\alpha_1 \equiv \alpha_2)$$  \hfill (26)

in which $p_i(\alpha_1 \equiv \alpha_2)$ denotes the equivalence probability estimated by the single attribute $B_i$, and $w_i$ represents the weight of the estimation result by $B_i$. The value of $w_i$ in Equation 26 is set by

$$w_i = \frac{\ln(ccf(B_i, A))}{\sum_{1 \leq i \leq k} \ln(ccf(B_i, A))}$$  \hfill (27)

in which $ccf(B_i, A)$ denotes the conditional correlation factor between $B_i$ and $A$, and $\ln()$ denotes the log function.
4 THE UNIFIED FRAMEWORK

The unified framework consists of two analytical components, string similarity measurement and value correlation analysis. We first describe a theory of evidence, the Dempster-Shafer theory, in Subsection 4.1, and then present how to use it to combine the estimation results obtained at the two components in Subsection 4.2.

4.1 The Dempster-Shafer Theory

The Dempster-Shafer (D-S) theory [43], [56] combines evidence from different sources and arrive at a degree of belief that takes into account all the available evidence. It is based on a frame of discernment composed of a set of propositions that are mutually exclusive and collectively exhaustive. In the D-S scheme, basic probabilities can be assigned to not only singleton propositions but also any of their subsets, thereby allowing a piece of evidence to be profiled by a Belief Distribution (BD) defined on the powerset of the frame of discernment. BD is regarded as the most natural and flexible generalisation of conventional probability distribution in the sense that the former allows inexact reasoning at whatever level of abstraction and on the other hand reduces to the latter if basic probabilities are assigned to singleton propositions only.

Formally, let \( X \) be the universal set representing all possible states of a system under consideration. By a function of Basic Belief Assignment (BBA), the D-S theory assigns a belief mass to each element of the power set. The mass of an element \( E_i, m(E_i) \), expresses the proportion of all relevant and available evidence that supports the claim that the actual state belongs to \( E_i \) but to no particular subset of \( E_i \). The masses of elements satisfy

\[
m(\emptyset) = 0 \text{ and } \sum_{E_i \in 2^X} m(E_i) = 1
\]

If only singleton propositions are assigned belief masses, a BBA function reduces to a classical probability function.

The kernel of D-S theory is Dempster’s rule, which is rooted in probability theory and constitutes a conjunctive probabilistic inference process. It generalises Bayes’ rule and was indeed promoted as the sole evidence combination rule to combine evidence in the D-S framework originally. The Dempster’s rule assumes that evidence is fully reliable: a proposition will not be supported at all if it is ruled out by an evidence. It adopts the orthogonal sum operation to combine evidence, which is rooted in probability theory and constitutes a conjunctive probabilistic function reduces to a classical probability function.

The BBA function of string similarity measurement assigns probability masses to singleton propositions in the following manner:

\[
m_s(\alpha_1 \equiv \alpha_2) = p_s(\alpha_1 \equiv \alpha_2) \text{ and } m_s(\alpha_1 \! \equiv \! \alpha_2) = 1 - p_s(\alpha_1 \equiv \alpha_2)
\]

Similarly, the BBA function of value correlation analysis assigns probability masses to singleton propositions in the following manner:

\[
m_v(\alpha_1 \equiv \alpha_2) = p_v(\alpha_1 \equiv \alpha_2) \text{ and } m_v(\alpha_1 \! \equiv \! \alpha_2) = 1 - p_v(\alpha_1 \equiv \alpha_2)
\]

Then, according to the Dempster’s rule, the combined probability of \( \alpha_1 \) and \( \alpha_2 \) being equivalent is computed by

\[
p_{s,v}(\alpha_1 \equiv \alpha_2) = \frac{p_s \cdot p_v}{p_s \cdot p_v + (1 - p_s) \cdot (1 - p_v)}
\]

in which \( p_s \) denotes \( p_s(\alpha_1 \equiv \alpha_2) \) and \( p_v \) denotes \( p_v(\alpha_1 \equiv \alpha_2) \).

For instance, suppose that the equivalence probabilities estimated by similarity measurement and correlation analysis are 0.8 and 0.6 respectively. Then, the combined probability is computed by \( p_{s,v}(\alpha_1 \equiv \alpha_2) = 0.857 \).

According to the above formula, the unified estimation would be 0 if any of the individual estimations is computed to be 0. To address this counter-intuitive issue, we propose to introduce evidence weights and translate them into simple support functions. We use the Shafer’s discounting method [43] to combine two weighted evidence. Suppose \( p(E_i) \) is the degree of belief to which a piece of evidence points to a proposition \( E_i \). Let \( w \) be a factor that is used to discount \( p(E_i) \), where \( w (0 \leq w \leq 1) \) is interpreted as evidence weight. The Shafer’s discounting method defines the BBA function for an evidence as follows:

\[
m(E_i) = \begin{cases} 0 & E_i = \emptyset \\ w \cdot p(E_i) & E_i \subset X, E_i \neq \emptyset \\ w \cdot p(E_i) + (1 - w) & E_i = X \end{cases}
\]

What Equation 29 means is that the degree of support for a proposition is proportional to both the weight of the evidence and the belief degree to which the evidence points to the proposition. It allocates the residual support left by the evidence due to its weight, as measured by \( (1 - w) \), to \( X \). The Shafer’s discounting method changes the specificity of the original evidence in the sense that as long as \( 1 - w > 0 \), global ignorance is introduced to a belief distribution even when the evidence points to a proposition \( E_i \) precisely and unambiguously, or \( p(E_i) = 1 \). In other words, if \( w < 1 \), there will always be \( m(X) \geq 1 - w > 0 \) even if \( p(X) = 0 \). We have noticed that there are other methods [56] for weighted evidence combination. The difference is on how to handle the residual support, as measured by \( 1 - w \). We propose to use the Shafer’s discounting method because assigning the residual support to \( X \) is most reasonable in this setting.

Generally, suppose that the evidence weights of similarity measurement and correlation analysis are \( w_s \) and \( w_c \) respectively. Then, we have

\[
m_s(\alpha_1 \equiv \alpha_2) = w_s \cdot p_s, m_v(\alpha_1 \equiv \alpha_2) = w_v \cdot (1 - p_s) \text{ and } m_s(X) = 1 - w_s
\]

Similarly, we also have

\[
m_v(\alpha_1 \equiv \alpha_2) = w_c \cdot p_c, m_v(\alpha_1 \equiv \alpha_2) = w_v \cdot (1 - p_c) \text{ and } m_v(X) = 1 - w_c
\]

Therefore, the unified equivalence probability, \( p_{s,v}(\alpha_1 \equiv \alpha_2) \), is computed by

\[
p_{s,v}(\alpha_1 \equiv \alpha_2) = \frac{w_s \cdot p_s \cdot w_c \cdot p_c + w_s \cdot p_s \cdot (1 - w_c) + w_c \cdot p_c \cdot (1 - w_c)}{1 - w_s \cdot p_s \cdot w_c \cdot p_c + w_s \cdot p_s \cdot (1 - w_c) + w_c \cdot p_c \cdot (1 - w_c)}
\]
Consider again the case that the equivalence probabilities estimated by similarity measurement and correlation analysis are 0.8 and 0.6 respectively; but both evidence weights are 0.5. Then, we have

\[ m_s(\alpha_1 \equiv \alpha_2) = 0.4, m_s(\alpha_1 ! \equiv \alpha_2) = 0.1, \text{and } m_s(X) = 0.5 \]

We also have

\[ m_c(\alpha_1 \equiv \alpha_2) = 0.3, m_c(\alpha_1 ! \equiv \alpha_2) = 0.2, \text{and } m_c(X) = 0.5 \]

The combined probability is therefore computed by

\[ m_{s,c}(\alpha_1 \equiv \alpha_2) = \frac{0.4 \cdot 0.3 + 0.4 \cdot 0.2 + 0.3 \cdot 0.5}{0.4 \cdot 0.2 + 0.3 \cdot 0.5} = 0.528. \]

We suggest to set the weights of string similarity measurement and value correlation analysis to be equal by default. Our experimental evaluation in Subsection 6.3 shows that evidence weighting can affect the performance of the unified framework and it performs best when the two weights are set to be equal. It is worthy to point out that the unified framework is simplified into string similarity measurement or probabilistic estimation by value correlation analysis if the weight of the other component is set to be 0. Suppose that the evidence weight of value correlation analysis is equal to 0, then the unified probability is reduced to:

\[ p_{s,c}(\alpha_1 \equiv \alpha_2) = m_{s,c}(\alpha_1 \equiv \alpha_2) = w_s \cdot p_s(\alpha_1 \equiv \alpha_2) \]

in which \( w_s \) represents the evidence weight of string similarity measurement.

5 Algorithms & Implementation

In this section, we develop the algorithms for the unified framework, focusing on the algorithms for value correlation analysis based on the basic and extended probabilistic models, and then describe their implementation on MapReduce. Note that attribute value matching based on string similarity measurement has been extensively studied in the literature [18].

5.1 Basic Model

According to Equation 7, the estimation of \( p(\alpha_1 \equiv \alpha_2) \) can be reduced to a series of set probability computations. Suppose that the distinct attribute values in \( B(\alpha_1) \) and \( B(\alpha_2) \) are \( \{\beta_1, \beta_2, \ldots, \beta_k\} \). We denote the occurrence frequencies of an attribute value \( \beta_i \) in \( B(\alpha_1) \) and \( B(\alpha_2) \) by \( n_{i1} \) and \( n_{i2} \), respectively. Then, \( p(\beta_1 = \beta_2) \) can be computed by:

\[
p(\beta_1 = \beta_2) = \frac{\sum_{1 \leq i \leq k} n_{i1} \cdot n_{i2}}{\sum_{1 \leq i \leq k} n_{i1} \cdot n_{i2}}
\] (31)

Given a relational table \( R \) and an attribute value \( \alpha_1 \) in \( A(R) \), the distinct values in \( B(\alpha_1) \) and their occurrence frequencies can be computed in \( O(|R|) \) time. Since value matching between \( B(\alpha_1) \) and \( B(\alpha_2) \) can be performed by hash join, we have the following lemma on the time complexity of computing \( p(\beta_1 = \beta_2) \) in Equation 7:

**Lemma 1.** Given a relational table \( R \) and two attribute values, \( \alpha_1 \) and \( \alpha_2 \), at \( A \), computing \( p(\beta_1 = \beta_2) \) in Equation 7 takes \( O(|R|) \) time.

The correlation factor \( f(A, B) \) in Equation 7 can be computed in a similar way. Suppose that \( A(R) \) has \( k \) distinct values, which are \( \{\alpha_1, \ldots, \alpha_k\} \), and their occurrence frequencies in \( A(R) \) are represented by \( \{n_{11}, \ldots, n_{kk}\} \). The value of \( p(\alpha_i = \alpha_j) \) in Equation 4 can be computed by:

\[
p(\alpha_i = \alpha_j) = \frac{\sum_{1 \leq i \leq k} \left( n_{ii} \cdot (n_{ii} - 1) \right)}{\sum_{1 \leq i \leq k} n_{ii} \cdot \left( \sum_{1 \leq i \leq k} n_{ii} - 1 \right)}
\] (32)

Therefore, we have the following lemma on the time complexity of computing the correlation factor:

**Lemma 2.** Given two attributes \( A \) and \( B \) in \( R \), computing their correlation factor \( f(A, B) \) takes \( O(|R|) \) time.

In the case of estimation by reference, the value of \( p(B(\alpha_i)) \) in Equation 10 can be computed in a way similar to what is shown in Equation 31. Therefore, we have the following theorem:

**Theorem 1.** Given a relational table \( R \) and two attribute values, \( \alpha_1 \) and \( \alpha_2 \), at \( A \), estimating \( p(\alpha_1 \equiv \alpha_2) \) in the basic model, whether by the correlation factor or reference, takes \( O(|R|) \) time.

**Algorithm 2:** Algorithm for Basic Model

```
Group the tuples in \( R \) by their attribute values at \( A \);
for \( \alpha_i \in A(R) \) do
  List the distinct values in \( B(\alpha_i) \) and their occurrence frequencies;
Group the tuples in \( R \) by their attribute values at \( B \);
for \( \beta_i \in B(R) \) do
  List the distinct values in \( A(\beta_i) \) and their occurrence frequencies;
  for each non-identical value pair \( (\alpha_j, \alpha_k) \) in \( A(\beta_i) \) do
    Generate \( ((\alpha_j, \alpha_k), n_{ijk}) \), in which \( n_{ijk} = n_{ij} \cdot n_{ik} \), and \( n_{ij} \) and \( n_{ik} \) represent the occurrence frequencies of \( \alpha_j \) and \( \alpha_k \) in \( A(\beta_i) \) respectively;
    for \( ((\alpha_j, \alpha_k), n_{ijk}) \) do
      Aggregate the values of \( n_{ijk} \) by the key, \((\alpha_j, \alpha_k)\), to generate \( ((\alpha_j, \alpha_k), n_{j}) \), in which \( n_j = \sum_i n_{ijk} \);
    for \( ((\alpha_j, \alpha_k), n_{j}) \) do
      Compute \( p(\alpha_j \equiv \alpha_k) \);
```

A straightforward algorithm for the basic model requires quadratic pairwise comparisons, whose cost may be prohibitively expensive even for modestly sized tables. Therefore, there is a need to improve the matching efficiency by reducing the required number of comparisons. It is observed that in Equation 31, \( p(\beta_1 = \beta_2) \) is not equal to 0 if and only if there exists at least a common value between the sets \( B(\alpha_1) \) and \( B(\alpha_2) \). Therefore, the basic model only needs to compare two attribute values at \( A \) that share at least one correlated value at \( B \). It can be achieved by grouping the tuples in \( R \) by the attribute values at \( B \) and listing all the value pairs at \( A \) within each group.

The algorithm for the basic model is sketched in Algorithm 2. It generates all the attribute value pairs at \( A \), \((\alpha_j, \alpha_k)\), which have at least a common value in their corresponding \( B(\alpha_j) \) and \( B(\alpha_k) \) sets, and computes their equivalence probability. In the generated \( (\alpha_j, \alpha_k), n_{jk} \), \( n_{jk} \) corresponds to the numerator in Equation 31. On its time complexity, we have the following theorem:

**Theorem 2.** Algorithm 2 takes \( O(|R| + \sum_{\alpha_i, \alpha_j \in A(R)} n_{ij}) \) time, in which \( n_{ij} \) represents the number of common distinct values in \( B(\alpha_i) \) and \( B(\alpha_j) \) for the pair of non-identical values at \( A \), \((\alpha_i, \alpha_j)\).

5.2 Extended Model

The computation of \( p(\beta_i \equiv \beta_j) \) in the extended model, as shown in Equation 11, requires to measure pair-wise similarity between the values in \( B(\alpha_i) \) and \( B(\alpha_j) \). On its time complexity, we have the following Lemma:

**Lemma 3.** Given \( R \) and two attribute values in \( A(R) \), \( \alpha_1 \) and \( \alpha_2 \), \( p(\beta_i \equiv \beta_j) \) in Equation 11 can be computed in \( O(|R| + n_{1} \cdot n_{2} \cdot t_B) \) time, in which \( n_{1} \) and \( n_{2} \) represents the numbers of distinct values in \( B(\alpha_1) \) and \( B(\alpha_2) \) respectively, and \( t_B \) denotes the worst-case time to compute the similarity between two \( B \) attribute values.

The computation of the correlation factor \( f(A, B) \) can be performed similarly. We have the following lemma on its time complexity:

**Lemma 4.** Given \( R \), the computation of \( f(A, B) \), as shown in Equation 12, takes \( O(|R| + n_{1}^2 \cdot t_A + n_{2}^2 \cdot t_B) \) time, in which \( n_A \)
and \( n_D \) represent the numbers of distinct values in \( A(R) \) and \( B(R) \) respectively, and \( t_1 \) and \( t_2 \) denote the worst-case time to compute the similarity between two \( A \) attribute values and between two \( B \) attribute values respectively.

Therefore, on the time complexity of the extended model, we have the following theorems:

**Theorem 3.** Given a relational table \( R \) and two attribute values, \( \alpha_1 \) and \( \alpha_2 \), at \( A \), estimating \( p(\alpha_1 \equiv \alpha_2) \) by the correlation factor in the extended model takes \( O(|R| + n_A^2 \cdot t_A + n_B^2 \cdot t_B) \) time.

**Theorem 4.** Given a relational table \( R \) and two attribute values, \( \alpha_1 \) and \( \alpha_2 \), at \( A \), estimating \( p(\alpha_1 \equiv \alpha_2) \) by reference in the extended model takes \( O(|R| + (n_1^2 + n_2^2 + n_1 \cdot n_2) \cdot t) \) time, in which \( n_1 \) and \( n_2 \) denote the number of distinct values in \( B(\alpha_1) \) and \( B(\alpha_2) \) respectively.

In the extended model, the number of attribute value comparisons can be reduced by token-based filtering if the similarity metric between two attribute values at \( B \) is token-based. For \( \alpha_1 \) and \( \alpha_2 \) to be considered for equivalence, there should exist a value in \( B(\alpha_1) \) and a value in \( B(\alpha_2) \) such that they share at least one common token. Note that not all the tokens are semantically meaningful. The stopwords can be simply filtered out because they have no semantic meaning.

**Algorithm 3:** Algorithm for Extended Model

```plaintext
for each \( \alpha_i \in A(R) \)
  [Retrieve \( B(\alpha_i) \),]
  [for each tuple \( t_i \in R \)]
    [for each token \( o_j \in B(t_i) \)]
      [Generate \((id_i, o_j, \alpha_i, \beta_i)\), in which \( id_i \) denotes the unique id of the tuple \( t_i \), and \( \alpha_i \) and \( \beta_i \) denote the tuple’s attribute values at \( A \) and \( B \) respectively;]
      [Group \((id_i, o_j, \alpha_i, \beta_i)\) by \( o_j \);]
      [for each \( o_j \)]
        [List all the tuple pairs, \(((id_j, id_k), (\alpha_j, \alpha_k), (\beta_j, \beta_k))\);]
        [Remove the duplicates in the set of \(((id_j, id_k), (\alpha_j, \alpha_k), (\beta_j, \beta_k))\), simplify them into \(((\alpha_j, \alpha_k), (\beta_j, \beta_k))\), and group them by the key \((\alpha_j, \alpha_k)\);]
        [for each group \((\alpha_j, \alpha_k)\) do]
          [Aggregate the values of \( \text{sim}(\beta_j, \beta_k) \) to generate \( ((\alpha_j, \alpha_k), \sum \text{sim}(\beta_j, \beta_k)) \);]
          [for each \((\alpha_j, \alpha_k, \sum \text{sim}(\beta_j, \beta_k))\) do]
            [Estimate \( p(B(\alpha_j), B(\alpha_k)) \) by Equation 11;]
            [Estimate \( p(\alpha_j \equiv \alpha_k) \) by \( f(A, B) \cdot p(B(\alpha_j), B(\alpha_k)) \);]
```  

The algorithms for estimating by the correlation factor and reference are similar. We sketch the algorithm for estimation by the correlation factor in Algorithm 3. It supposes that the correlation factor \( f(A, B) \) has been computed beforehand. Its time complexity can be estimated by

\[
T_e(A, B) = O(|R| \cdot m_B + T_e(A, B)) \tag{33}
\]

in which \( m_B \) denotes the maximal number of tokens contained by a \( B \) attribute value and

\[
T_e(A, B) = \sum_{\alpha_i, \alpha_j \in A(R)} \beta_i \mid B(\alpha_i) \beta_j \in B(\alpha_j) (c_{ij} \cdot t + t_B) \tag{34}
\]

in which \( \alpha_i \) and \( \alpha_j \) denotes two non-identical values in \( A(R) \), \( \beta_i \) and \( \beta_j \) denote a value in \( B(\alpha_i) \) and \( B(\alpha_j) \) respectively, \( m_{ij} \) denotes the number of common tokens shared by \( \beta_i \) and \( \beta_j \), and \( c_{ij} = 1 \) if \( \beta_i \) and \( \beta_j \) share at least a common token, otherwise \( c_{ij} = 0 \). Therefore, we have the following theorem:

**Theorem 5.** Algorithm 3 takes \( T_e(A, B) \) time.

### 5.3 Attribute Selection

The conditional correlation factor, \( ccf(B, A) \), as shown in Equation 17, and the performance diversity, \( D_A(B_1, B_2) \) as shown in Equation 25, can be computed in a similar way to how the correlation factor, \( f(B, A) \), is computed. On time complexity of computing \( ccf(B, A) \), we have the following theorems:

**Theorem 6.** In the basic model, computing \( ccf(B, A) \) takes \( O(|R|) \) time.

**Theorem 7.** In the extended model, computing \( ccf(B, A) \) takes \( O(|R| + n_B^2 \cdot n_A + n_B^2 \cdot t_B) \) time, in which \( n_A \) and \( n_B \) denote the number of distinct values in \( A(R) \) and \( B(R) \) respectively, and \( t_B \) denotes the worst-case time to compute the similarity between two \( B \) attribute values.

Similarly, given a target attribute \( A \) and two candidate attributes \( B_1 \) and \( B_2 \) in \( R \), we have the following theorems on time complexity of computing \( D_A(B_1, B_2) \):

**Theorem 8.** In the basic model, computing \( D_A(B_1, B_2) \) takes \( O(|R|) \) time.

**Theorem 9.** In the extended model, computing \( D_A(B_1, B_2) \) takes \( O(|R| + n_B^2 \cdot (n_A + t_{B_1}) + n_B^2 \cdot (n_A + t_{B_2}) \cdot t_B) \) time, in which \( n_A \), \( n_{B_1} \), and \( n_{B_2} \) denote the number of distinct values in \( A(R) \), \( B_1(R) \) and \( B_2(R) \) respectively, and \( t_{B_1} \) and \( t_{B_2} \) denote the worst-case time to compute the similarity between two \( B \) attribute values and between two \( B \) attribute values respectively.

### 5.4 Implementation on MapReduce

#### 5.4.1 MapReduce

MapReduce [16] is popular shared-nothing parallel framework that provides an easy mechanism to implement iterative processing and fault tolerance. It processes distributed data across many nodes via three basic phases. In the Map phase, it takes an input and produces a list of intermediate key/value pairs without communication between nodes. Next, the Shuffle phase repartitions these intermediate pairs according to their keys across nodes. Finally, the Reduce phase aggregates the intermediate pairs it receives to produce final results. This process can be repeated by invoking an arbitrary number of additional Map-Reduce cycles as necessary. MapReduce is notable for having a more robust runtime system than most SQL DBMSs, in that it is extremely tolerant of node failures.

A MapReduce program is composed of a Map() procedure that performs sorting and filtering and a Reduce() procedure that performs a summary operation. We use Apache Spark [1], the popular open-source implementation of MapReduce, to implement the unified framework.

#### 5.4.2 Filtering Techniques

The basic algorithms for value correlation analysis requires quadratic pairwise comparisons, whose cost may be prohibitively expensive even for modestly sized tables. Therefore, there is a need to improve the matching efficiency by reducing the required number of comparisons.

In Equation 31, \( p(\beta_1 = \beta_2) \) is not equal to 0 if and only if the sets \( B(\alpha_1) \) and \( B(\alpha_2) \) share at least one value member. Therefore, the basic model only needs to compare two attribute values at \( A \) that share at least one correlated value at \( B \). It can be achieved by grouping the tuples in \( R \) by the attribute values at \( B \) and listing all the value pairs at \( A \) within each group.

In the extended model, the number of attribute value comparisons can be reduced by token-based filtering if the similarity metric between two attribute values at \( B \) is token-based. For \( \alpha_1 \) and \( \alpha_2 \) to be considered for equivalence, there should exist an attribute value \( \beta_{i1} \) in \( B(\alpha_1) \) and an attribute value \( \beta_{i2} \) in \( B(\alpha_2) \) such that \( \beta_{i1} \) and \( \beta_{i2} \) share at least one common token. Otherwise, given a token-based similarity metric, the numerator of Equation 11 would be computed to be 0. Note that not all the tokens are semantically meaningful. The
stopwords can be simply filtered out because they have no semantic meaning. According to the TF-IDF theory in information retrieval, the tokens that occur too frequently or too unfrequently in a corpus may have little semantic meaning. As a result, they can also be filtered out. The token-based filtering can set a lower bound and an upper bound on token frequency to filter out unmeaningful tokens.

5.4.3 Basic Model on MapReduce

We first describe the MapReduce process to compute the correlation factor \( f(A, B) \), and then explain how to compute \( p(\beta_1 \simeq \beta_2) \), as shown in Equation 7.

Since the numerator and denominator of \( f(A, B) \) are symmetric, they can be implemented in the same way on MapReduce. The computation of the denominator, \( p(\beta_1 = \beta_2|\alpha_i = \alpha_j) \), consists of two MapReduce cycles:

1) In the first cycle, the Map phase generates the key-value pairs with the format of \((\alpha_i, \beta_j, \text{count}_{ij})\), in which \text{count}_{ij} represents the number of tuples, whose values on \( A \) and \( B \) are \( \alpha_i \) and \( \beta_j \) respectively.

2) In the second cycle, the Map phase transforms the output of the first cycle into the key-value pairs with the format of \((\alpha_i, \beta_j, \text{count}_{ij}, \text{sim}_{ij})\). The following Reduce phase aggregates them on each \( \alpha_i \) into groups with the format of \((\alpha_i, \{< \beta_j, \text{count}_{ij}, \text{sim}_{ij} \}, ..., < \beta_k, \text{count}_{ik}, \text{sim}_{ik} \}), \) which \text{count}_{ij} corresponds to \( n_1 \) in Equation 32.

Now we consider the computation of \( p(\beta_1 = \beta_2) \) as shown in Equation 31. Its denominator requires the numbers of tuples in \( R(\alpha_i) \). They can be easily retrieved by a MapReduce cycle. A Map phase maps each tuple into a key-value pair with the format of \((\alpha_i, 1)\) and then the Reduce phase aggregates them into groups with the format of \((\alpha_i, \text{count}_{i}), \) in which \text{count}_{i} represents the number of tuples whose attribute value on \( A \) is \( \alpha_i \). The numerator of Equation 31 requires to compute the occurrence frequency of each attribute value \( \beta_j \) (\( \beta_j \in B(\alpha_i) \)) in \( R(\alpha_i) \). Its computation can be implemented by the following three MapReduce cycles:

1) In the first cycle, the Map phase generates the key-value pairs with the format of \((\alpha_i, \beta_j, 1)\). The Reduce phase aggregates them to produce the results of \((\alpha_i, \beta_j, \text{count}_{ij})\), in which \text{count}_{ij} represents the number of tuples, whose values on \( A \) and \( B \) are \( \alpha_i \) and \( \beta_j \) respectively.

2) In the second cycle, the Map phase first maps the output of the first cycle to generate the key-value pairs with the format of \((\beta_j, < \alpha_i, \text{count}_{ij}>)\). The Reduce phase, the key-value pairs are aggregated on each \( \beta_j \) to generate the output with the format of \((< \alpha_i, \beta_j, \text{count}_{ij}>)\), where \text{count}_{ij}=\text{count}_{ij}, \text{count}_{ij}. Note that the pair \( < \alpha_i, \beta_j > \) is order-insensitive: \( < \alpha_i, \beta_j > \) and \( < \alpha_j, \beta_i > \) refer to the same pair.

3) Finally, the third cycle reduces the output of the second cycle to produce the results of \((< \alpha_i, \beta_j, \sum \text{count}_{ij}>)\), in which \sum \text{count}_{ij} corresponds to the numerator in Equation 31 for the value pair \( \alpha_i \) and \( \alpha_j \).

5.4.4 Extended Model on MapReduce

The token-based filtering technique has the good property that it can be easily incorporated into the matching process based on MapReduce. The matching process with token-based filtering consists of two steps, token-based grouping and similarity aggregation, each of which can be implemented by a MapReduce cycle. The token-based grouping step assigns the tuples to blocks, each of which corresponds to a token. A block consists of all the tuples whose attribute value at \( B \) contains its corresponding token. Token-based grouping is implemented by the following MapReduce cycle:

1) The Map phase splits the attribute value at \( B \) of each tuple into tokens and emits the intermediate key-value pairs with the format of \((t, < \text{id}, \alpha_i, \beta_j >)\), in which \( t \) represents the token, \( \text{id} \) represents the tuple identity, \( \alpha_i \) and \( \beta_j \) represent the tuple’s attribute values on \( A \) and \( B \) respectively.

2) The Reduce phase aggregates the output of Map() on the token \( t \) into the groups with the format of \((t, \{ < \text{id}_1, \alpha_1, \beta_1 >, ..., < \text{id}_k, \alpha_k, \beta_k > \} \).

The second step of similarity aggregation is implemented by the following MapReduce cycle:

1) The Map phase takes each group of \((t, \{ < \text{id}_1, \alpha_1, \beta_1 >, ..., < \text{id}_k, \alpha_k, \beta_k > \}) \) and transforms it into the key-value pairs with the format of \((< \alpha_1, \alpha_j >, \text{sim}(\beta_1, \beta_j))\), in which \text{sim}() is a similarity metric between \( \beta_1 \) and \( \beta_j \). For each token \( t \), it iterates over every distinct pair of \((\text{id}_i, \text{id}_j)\), which is order-insensitive, and emits the corresponding key-value pairs.

2) The reduce phase aggregates the key-value pairs of \((< \alpha_1, \alpha_j >, \text{sim}(\beta_1, \beta_j))\) on the key to retrieve the summary similarity at \( B \) for each \( < \alpha_1, \alpha_j > \) pair, \((< \alpha_1, \alpha_j >, \sum \text{sim}(\beta_1, \beta_j))\). Note that \( \sum \text{sim}(\beta_1, \beta_j) \) corresponds to the numerator in Equation 11.

The computation of the reference probability, \( p(\beta_11 \simeq \beta_12) \), can be similarly implemented. We first randomly mark some \( \alpha_1 \) on \( A \) and then use the token-based MapReduce matching process to compute \((< \alpha_1, \alpha_1 >, \sum \text{sim}(\beta_1, \beta_1))\).

6 Experimental Study

In this section, we empirically evaluate the performance of the unified framework on real datasets. The details of the three test datasets are described as follows:

1) DBLP. The DBLP dataset [4] contains the information on the papers published in the research conferences and journals in various areas of computer science. The original dataset is clean, containing no dirty data. For experimental purpose, we generate dirty data by manipulating the attribute values at journal, which specifies the venue a paper is published in. The journal values are transformed by a series of edit operations, e.g. insert, delete and replace, as defined in the Levenstein distance [34].

2) HOTEL. The dataset contains the information of more than 138 thousands hotels around the world. Similar to the case of DBLP, we manually generate dirty data by manipulating the attribute values at city and aircode, which specify the city a hotel is located at and the code of its nearest airport respectively. The values are also transformed by a series of edit operations as defined in the Levenstein distance.

3) DBLP+CiteSeer (DLCS). The CiteSeer dataset [2] is similar to the DBLP data. It also records the published research papers in computer science. An author and a publication venue may have different representations in DBLP and CiteSeer. Our experiments aim to match the attribute values of journal and author between DBLP and CiteSeer. DBLP and CiteSeer contain around 960 thousand and 45 thousand tuples respectively. To facilitate repeated experimentation, we randomly choose 20% tuples in DBLP and match them with all the tuples in CiteSeer.

4) CORA. The CORA dataset [3] also contains the published research papers in computer science. Unlike the DBLP and CiteSeer datasets, it is pretty dirty by itself. In CORA, two records may refer to a same paper entity but their publication venues have different representations. The dataset contains totally 1,295 records and 474 attribute values at venue, which specifies publication venue. Our experiments aim to match the attribute values at venue. Since the venue values in CORA carry time information, two venues are considered to be different if they correspond to a same conference/journal series but have distinct publication time (e.g. “the 31th IEEE International Conference on Data Engineering” and “the 32nd IEEE International Conference on Data Engineering”).
On DBLP and HOTEL, we simulate two application scenarios for equivalence reasoning: integrating two tables from two data sources, and correcting value misrepresentations within a table. On DBLP, we first randomly select 400 journal values, and retrieve their corresponding tuples to generate the test data. In the first scenario, we randomly partition the test data into two equal-sized parts, simulating two data sources. Then, we randomly select 10% of the journal values in the test data, and change them in one of the parts. In the second scenario, we also randomly select 10% of the journal values in the test data. For each chosen value, we retrieve its corresponding tuples and randomly select a few (ranged from $n_l$ to $n_u$) from them for manipulation. By the range $[n_l, n_u]$, we vary the distribution of the numbers of value misrepresentations within a table. On HOTEL, the manipulative process is similar. There are totally 2359 distinct city values and 3497 distinct aircode values. In the first scenario, we manually change 10% of the city and aircode values in one of the partitioned data. In the second scenario, we set a range for the number of manipulated tuples for each chosen value in the test data.

Our experiments compare the performance of three approaches, based on string similarity measurement (SSM), value correlation analysis (VCA) and the unified framework (TUF) respectively. On the DBLP and HOTEL datasets, dirty attribute values are introduced by edit operations as defined in the Levenshtein distance. The string-based approach therefore uses the Levenshtein distance as its similarity metric, which can achieve the best performance. On the DLCS and CORA datasets, it uses a hybrid metric, which combines a TF-IDF weighting scheme with the Jaro-Winkler scheme. Our experiments showed that the hybrid metric performs better than the alternatives (e.g., the edit distance metrics and other token-based metrics). The hybrid metric has also been shown to achieve good performance in previous work [14].

We measure the performance of different approaches by the non-interpolated average precision (NAP), the maximum F-score (MaxF), and also the precisions at various cutoff levels. The non-interpolated average precision of a ranked list containing $n$ pairs for a task with $m$ correct matches is $\frac{1}{m} \cdot \sum_{i=1}^{n} c(i) \delta(i)$, where $c(i)$ is the number of correct pairs ranked before position $i$, and $\delta(i) = 1$ if the pair at rank $i$ is correct and 0 otherwise. Maximum F-score is $\max_{i} F_1(i)$, where $F_1(i)$ is the harmonic mean of recall and precision at rank $i$. Since each attribute value has at most one representation variation, we impose the 1-to-1 matching constraint that an attribute value should be matched to at most one other value. This practice has also been usually adopted in record matching.

All the algorithms presented in Section 5 can be efficiently parallelized on hash-partitioned data, and therefore can be easily implemented on popular parallel platforms. We have implemented them on Spark (version 1.1.1) [1], a open-source implementation of the MapReduce paradigm. The rest of this section is organized as follows: Subsection 6.1 compares the performance of different approaches on matching accuracy. Subsection 6.2 evaluates the effectiveness of probabilistic equivalence reasoning by multiple attributes. Subsection 6.3 evaluates how evidence weighting affects the performance of the unified framework. Finally, in Subsection 6.4, we evaluate efficiency of our parallel implementation of value correlation analysis.

### 6.1 Comparative Evaluation

The unified framework sets both weights of SSM and VCA to the default value of 0.5. On the DBLP dataset, the attributes title and author are chosen for equivalence reasoning on journal. On HOTEL, the attribute address is chosen for city, and the attributes address and city are chosen for aircode. On the DLCS dataset, the attributes title and author are chosen for equivalence reasoning on journal, and the attribute title and journal are chosen for equivalence reasoning on author.

#### 6.1.1 DBLP

On the DBLP dataset, we specify the numbers of edit operations executed on the attribute values at journal by a normal distribution, $N(\mu, \sigma^2)$. To evaluate the comparative performance in the circumstances where SSM has varying accuracies, we fix the value of standard deviation($\sigma$) at 2 but vary the value of mean(\mu) from 3 to 7.

The precisions measured at different cutoff levels are presented in Figure 1. We present the precision results at the cutoff levels up to 100, where TUF achieves the maximal recalls. In the case of $\mu=3$, it can be observed that even though SSM performs better than VCA, TUF performs clearly better than SSM. Close scrutiny reveals that VCA tends to match the journals in a same research area, whose papers

---

**Fig. 1: Precision-Cutoff Evaluation on DBLP:** (a), (b) and (c) are for the first scenario; (d) is for the second scenario.

**Fig. 2: Precision-Cutoff Evaluation on HOTEL:** (a) and (b) are for the first scenario; (c) and (d) are for the second scenario.
have similar titles and share some common authors. It effectively filters out the journal names that appear similar but are actually in different research areas from its top ranked list. As a result, the unified approach, which combines both analytical results obtained from SSM and VCA, achieves improved matching accuracy.

It can also be observed that as the value of $\mu$ increases, the performance of SSM deteriorates as expected while the performance of VCA remains stable and as a result, the performance improvement achieved by TUF over SSM becomes more considerable. For instance, with $\mu = 7$, at the cutoff levels of 10-50, TUF outperforms SSM by 30%-40% on precision. In comparison, its improvements over SSM in the cases of $\mu = 3$ and $\mu = 5$ at the same cutoff levels are around 10%-20% and 20%-30% respectively. As the value of $\mu$ increases, SSM becomes less reliable in determining value equivalence at journal. The performance of VCA is instead independent of the contents of the journal values. Since TUF leverages both SSM and VCA estimation results, its performance improvement over SSM increases as the performance of SSM deteriorates.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Max Recall</th>
<th>NAP</th>
<th>MaxF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM($\mu = 3$)</td>
<td>0.900</td>
<td>0.643</td>
<td>0.649</td>
</tr>
<tr>
<td>TUF($\mu = 3$)</td>
<td>0.975</td>
<td>0.858</td>
<td>0.810</td>
</tr>
<tr>
<td>SSM($\mu = 5$)</td>
<td>0.750</td>
<td>0.482</td>
<td>0.453</td>
</tr>
<tr>
<td>TUF($\mu = 5$)</td>
<td>0.925</td>
<td>0.777</td>
<td>0.724</td>
</tr>
<tr>
<td>SSM($\mu = 7$)</td>
<td>0.700</td>
<td>0.358</td>
<td>0.364</td>
</tr>
<tr>
<td>TUF($\mu = 7$)</td>
<td>0.925</td>
<td>0.753</td>
<td>0.716</td>
</tr>
</tbody>
</table>

In the second scenario, the values of $\mu$ and $\sigma$ of the normal distribution are set to be 5 and 2 respectively. We set three ranges, [2, 4], [4, 8] and [8, 12] for the number of manipulated tuples for each chosen journal value. The performance comparison between SSM and TUF is presented in Figure 1 (d). It can be observed that with the range [2, 4], which means that only 2-4 tuples exist for each manipulated value, the unified approach achieves clearly better performance than SSM. As the values of the range increase, the performance of VCA improve accordingly. As a result, the performance improvement achieved by TUF over SSM becomes more considerable.

The performance comparisons between SSM and TUF on NAP and MaxF are also presented in Table 3. We present the maximal recalls of different approaches as well. Note that the maximal recall may be less than 100% because of the imposed 1-to-1 matching constraint. It can be observed that TUF consistently outperforms SSM and the performance improvement increases with the value of $\mu$.

### 6.1.2 HOTEL

On the HOTEL dataset, we specify the numbers of edit operations performed on the city values by a normal distribution of $\mathcal{N}(\mu, \sigma^2)$ with $\mu = 3$ and $\sigma = 2$. The airport values have only three characters. Therefore, we randomly replace 1 or 2 characters at the chosen values.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Max Recall</th>
<th>NAP</th>
<th>MaxF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM(city)</td>
<td>0.805</td>
<td>0.402</td>
<td>0.411</td>
</tr>
<tr>
<td>TUF(city)</td>
<td>0.950</td>
<td>0.857</td>
<td>0.792</td>
</tr>
<tr>
<td>SSM(aircode)</td>
<td>0.352</td>
<td>0.078</td>
<td>0.136</td>
</tr>
<tr>
<td>TUF(aircode)</td>
<td>0.764</td>
<td>0.703</td>
<td>0.630</td>
</tr>
</tbody>
</table>

The comparative precision-cutoff evaluation results are presented in Figure 2. We report the precision results up to top-1000. After the cutoff level of 1000, every approach generates few (≤ 5) correct matches.

In the first scenario, it can be observed that on city, the performance difference between SSM and VCA is not significant, but TUF outperforms them by considerable margins. Even though neither SSM nor VCA achieves the precision above 50%, TUF achieves a precision above 90% at the cutoff level of 100. On aircode, the evaluation results are similar except that TUF outperforms SSM by even bigger margins. It can be observed that SSM performs very poorly in this case. With only three characters, many aircode value pairs have the same string similarity but most of them actually represent different airports. There are a lot of false positive matches in the top-k results. As a result, the performance of SSM remains low (around 10%) at various cutoff levels. In comparison, by analysing the correlated city and address values, VCA achieves much higher matching accuracy. It is interesting to point out that even though SSM performs very poorly, TUF performs well (it achieves the precisions of around 80% and 70% at the cutoff levels of 100 and 200 respectively).

The precision-cutoff evaluation results for the second scenario are similar to what were observed on the DBLP dataset. With the smallest range of [2, 4], TUF achieves considerably better performance than SSM. The performance improvement also becomes more considerable as the range values increases.

The performance comparisons between SSM and TUF on NAP and MaxF are also presented in Table 4. It can be observed that TUF consistently outperforms SSM by considerable margins.

#### 6.1.3 DLCS & CORA

On the DLCS and CORA datasets, manually checking the true recall values is a daunting task. Therefore, we only evaluate the precision-cutoff performance. The experimental results on DLCS are shown in Figure 3 (a) and (b). Similar to what were observed on DBLP and HOTEL, TUF consistently outperforms SSM on matching precision. On journal, SSM performs well at the cutoff levels of 40 and 80. It achieves the precisions of 97.5% and 87.5% respectively. At these two cutoff levels, TUF achieves slightly better performance than SSM. After that, TUF outperforms SSM by considerable margins: at the cutoff levels between 160 and 400, TUF achieves around 20% improvement over SSM.

On author, SSM does not perform as well as on journal. In DLCS, there are many author names that appear similar but actually represent different researchers. As a result, SSM returns many false positive matches in its top-k results. It can be observed that even though VCA performs worse than SSM, TUF performs much better than SSM. After the cutoff level of 80, TUF achieves at least 30% improvement over SSM on precision.

The experimental results on CORA are presented in Figure 5. It can be observed that SSM performs poorly. CORA contains many venue values, which correspond to a same conference/journal series but have different publication time (e.g. “ICDE, 1998” and “ICDE, 1999”). Ranking by SSM results in many false positive matchings in top-k pairs. In contrast, VCA performs well on CORA, achieving precision between 70% and 80% at various cutoff levels. It can also be observed that TUF achieves the best performance among them.

### 6.1.4 Summary

Our comparative experiments on real datasets demonstrate that the unified approach can effectively leverage the analytical results obtained from string similarity measurement and value correlation analysis to improve matching accuracy. The achieved improvement on performance tends to increase with the deteriorating performance of the string-based approach. It can achieve good performance even in the circumstance where string similarity metrics perform very poorly.
Our experiments also show that the unified approach can achieve better performance than the string-based approach in the circumstance that there exists only a few tuples for misrepresented values. These observations bode well for its practical effectiveness, especially in the circumstance where string similarity metrics can not provide with reliable estimation on attribute value equivalence.

### 6.2 Attribute Combination

This subsection evaluates the effectiveness of attribute selection and estimation combination in value correlation analysis, as presented in Subsection 3.3. On all the three datasets, the threshold of conditional correlation factor($\theta_c$) for candidate filtering is set to be 5; the threshold of the performance diversity between two candidate attributes($\theta_d$) for combination generation is set to be 0.3. The conditional correlation factor of chosen attributes and their performance diversity with regard to a target attribute are detailed in Table 5.

<table>
<thead>
<tr>
<th>Dataset(A)</th>
<th>$ccf(B_1, A)$</th>
<th>$ccf(B_2, A)$</th>
<th>$D_A(B_1, B_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLPjournal</td>
<td>24.70</td>
<td>14.84</td>
<td>0.48</td>
</tr>
<tr>
<td>B1=author</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2=title</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLCSjournal</td>
<td>332.50</td>
<td>69.60</td>
<td>0.79</td>
</tr>
<tr>
<td>B1=author</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2=title</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOTELaircode</td>
<td>440.14</td>
<td>92.15</td>
<td>0.81</td>
</tr>
<tr>
<td>B1=city</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2=address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The evaluation results on DBLP are presented in Figure 4 (a) and (b). On DBLP, we set the value of $\mu$ to be 5. The evaluation results for other values of $\mu$ are similar, thus omitted here. Value correlation analysis are executed on author, title and the combination of author and title respectively. Note that the value of $ccf(\text{author}, \text{journal})$ is larger than the value of $ccf(\text{title}, \text{journal})$. In Figure 4 (a), it can be observed that VCA on author performs better than VCA on title. Similarly, in Figure 4 (b), it can be observed that TUF with VCA on author performs better than TUF with VCA on title. We also observe that VCA on the attribute combination clearly outperforms VCA on single attributes. The comparative performance of their corresponding TUF are similar. The evaluation results on DLCS are also presented in Figure 3 (c) and (d). They are similar to what were observed on DBLP.

On HOTEL, the value of $ccf(\text{city}, \text{aircode})$ is larger than the value of $ccf(\text{address}, \text{aircode})$. As a result, VCA on city performs better than VCA on address. VCA on the attribute combination outperforms both of them. TUF with VCA on the attribute combination also achieves the best performance.

Our experiments show that the higher conditional correlation factor a candidate attribute has, the better performance its corresponding VCA and TUF can achieve. Moreover, the performance of VCA and TUF is better on attribute combination than on any single attribute. They validate the effectiveness of our approach for value correlation analysis based on multiple attributes proposed in Subsection 3.3.

### 6.3 Evidence Weighting

This subsection evaluates how evidence weighting can affect the performance of the unified framework. We present the evaluation

Fig. 6: Evidence Weighting
results on matching journal values on the DBLP dataset. The results on other datasets are similar, thus omitted here due to space limit. The $\mu$ and $\sigma$ values of the normal function are set to be 5 and 2 respectively. Our experiments showed that if the weights of SSM and VCA are set to be equal ($0 < w < 1$), the performance of TUF remains stable regardless of the value of $w$. Therefore, we set the weight of a component to the default value of 0.5 and vary the weight of the other component from 0.1 and 0.5.

The detailed results are presented in Figure 6. From Figure 6 (a), it can be observed that if the weight of SSM is set to be 0.5, TUF achieves the best performance when the weight of VCA is also set to be 0.5. TUF with any weight setting outperforms SSM and its performance consistently improves as the weight of VCA increases from 0.1 to 0.5. The experimental results for the case of varying the SSM weight, as presented in Figure 6 (b), are similar. TUF with both weights set to be the default value of 0.5 achieves the best performance.

### 6.4 Efficiency Evaluation

![Fig. 7: Scalability Evaluation](image)

We evaluate both the performance variation of the proposed value correlation analysis algorithms on a single worker with data size and their scalability on multiple workers. The algorithms were evaluated on DBLP. Value correlation analysis is performed at the attribute author for equivalence reasoning at journal. In the first case, we vary data size (measured by the total number of tuples) from 200,000 to 700,000. In the second case, the whole DBLP dataset is used and the number of workers vary from 2 to 20.

The experimental results are presented in Figure 7. From Figure 7 (a), we observe that the consumed time of VCA increases nearly linearly with data size. From Figure 7 (b), we observe that when the number of workers is small ($\leq 6$), the speedup is very close to the number of workers. As the number of workers continues to increase, the synchronization and communication cost becomes increasingly considerable. Finally, the line of speedups flattens out as expected. Our experiments show that value correlation analysis can be efficiently executed, and the process can easily parallelized across a distributed cluster based on the MapReduce framework.

### 7 RELATED WORK

#### Attribute Value Matching in Relational Data

The state-of-the-art approaches consider the contents of attribute values for equivalence reasoning. Different similarity metrics usually have different suitable application domains. For instance, the metric of edit distance [34], [41] does well in detecting typographical errors. The Jaro metric [29] is intended primarily for short strings. The token-based metrics [13], [26] are usually used in the circumstance where word order is unimportant (e.g., personal names). Since string similarity metrics can be designed to accommodate the representation variations specific to a domain, they are usually able to achieve good performance in their target applications. However, a similarity metric tuned and tested on previous match problems may perform poorly on new problems. By incorporating string similarity metrics into the proposed unified framework, our work complements, but does not replace, the existing work based on string similarity measurement.

#### Duplicate Record Detection

Duplicate record detection, which studies whether two records consisting of multiple attributes refer to a same real-world entity, have been extensively studied in the literature [18]. Most effective and widely used approaches (e.g., probabilistic [23], rule-based [48], [28], supervised and semisupervised learning [12], [15]) were built on value matching on individual attribute fields. They solve the additional problem of how to merge the matching results on attribute fields. Obviously, our work on attribute value matching is complementary to them in that determining attribute value equivalence can effectively improve the accuracy of record matching. For efficient entity resolution, blocking algorithms [31], [42] are usually used to separate records into blocks likely to contain matching pairs such that unnecessary pair-wise comparisons can be avoided. It can be observed that attribute value equivalence reasoning can also effectively improve the accuracy of record blocking.

The need for attribute dependencies to decide matching keys has long been recognized [37], [53]. It can be achieved by probabilistic models using expectation maximization (EM) algorithms [38], [54] or reasoning about matching dependencies [22]. However, these techniques can only reason about the uniqueness and reliability of attribute fields for the purpose of designing effective record matching rules. They cannot be applied to reason about the equivalence between two specific attribute values.

#### Schema Matching

Schema matching [19], [7], [20], [6], [24], [17], [44], [30] studies the problem of generating correspondences between elements of two schemas. In the relational context, it identifies columns that represent the same concepts in two tables. Effective techniques usually take a hybrid approach, leveraging different criteria (e.g., linguistic matching, instance-based matching, structured-based matching and rule-based matching) to arrive at suggested correspondences. According to the criteria of instance-based matching, two columns are regarded as similar if their instances are similar. Our work on attribute value equivalence reasoning can be used to improve the accuracy of instance-based matching. It can therefore be leveraged by the existing schema matching techniques.

#### Entity Name Disambiguation

There also exist work on entity name disambiguation in relational data [8] or on Web [57], [55]. They assumed that identical names may refer to different real-world entities and focused on distinguishing these names. [8] solved name disambiguation in relational data by measuring two names’ neighborhood similarity besides their string similarity. It also contended that collective entity resolution, in which related entities are determined jointly, rather than independently, can effectively improve matching accuracy. [57] and [55] addressed name disambiguation on Web. [57] proposed to iteratively label data and then train type classifiers using the newly labeled data. [55] introduced a pairwise factor graph model for person name disambiguation, and proposed an active algorithm aiming to maximizing the utility of user correction.

In the circumstance that two identical values may refer to two different values, equivalence reasoning is very challenging because every appearance of value should be assumed to correspond to a distinct entity at the outset. An alternative solution is to first cluster distinct values into equivalent groups, which is the topic of this paper, and then distinguish the values in the same group. It is worthy to point out that our proposed approach can be similarly used to distinguish two identical values. In the case that two identical values refer to two distinct entities, it can be supposed that they are correlated with largely different other values. The application of the proposed approach to value disambiguation is however beyond the scope of this paper.

#### Entity Resolution on Web
The work on entity resolution on the Web [10], [27], [33], [52], [47] leveraged the knowledge acquired from external sources for domain-independent entity resolution. [10], [27], [33] studied how to match two web scale taxonomies. [10], [27] extracted entities’ surface forms (names or aliases) from Wikipedia and built a synonym dictionary for entities. [33] explored negative evidence in the structure of the taxonomy, as well as on the web, to improve matching accuracy. [52], [47] studied how to link named entity mentions detected from the unstructured Web text with their corresponding entities existing in a knowledge base. However, our task of attribute value matching in relational data is different from the problem of entity resolution on Web and no existing methods for this problem can be applied to address it.

8 CONCLUSION
This paper presents a novel probabilistic approach to reason about attribute value equivalence in relation data by value correlation analysis. Representing a new perspective for equivalence reasoning, it is complementary to the existing string-based approach. It also proposes a unified framework for attribute value matching. It builds on both analytical approaches and provides with a unified estimation by evidential reasoning. Our extensive experiments on real datasets have validated the efficacy of the new framework.

REFERENCES