Learning Rules for Effective Almost-parameter-free Instance matching

ABSTRACT

Instance matching is an important step in data integration where the goal is to find instance representations referring to the same entity. In this paper, we propose an efficient approach to learn attributes, similarity functions, and thresholds, called instance-matching rules, for finding matches. Existing rule-based approaches calculate similarity of each attribute separately, and identify an instance pair as a match if each of the similarities is high enough. They may fail to identify matching instance pairs if there are errors occur in a single attribute. Besides, these approach cannot effectively learn the rules without the fine-tuning of parameters. At mean while, these approaches are also expensive in learning, because they learn the best rule from a large number of candidates whose number depends on the number of attributes, similarity functions, and especially training examples. In this paper, we address these three problems. We measure two instances as a whole by calculating the average similarity of a set of attributes to balance the errors in single one. The approach we proposed in this paper is almost free of parameters, which can easily estimate the value of the parameters from the training data and require not fine-tuning of them. We then propose an efficient algorithm to learn the instance-matching rules from a significantly smaller set of candidates whose size only depends on the number of attributes and similarity functions. The experiments on both real and synthetic datasets show that our solution greatly improves the effectiveness as well as the efficiency by up to 87% reduction of learning time. Moreover, the approach is also effective in the way that it can achieve stable results when the parameters are set with a large range of different values.

1. INTRODUCTION

Structured data is abundantly available in enterprises and also largely increasing in the Web setting. Generally speaking, it can be conceived as structured descriptions of real-world entities. A popular standard for making such descriptions available on the Web is RDF. It is a graph-structured data model that can be flexibly used to compose entity descriptions as sets of $\langle subject, predicate, object\rangle$ triples, or in other words, edges of an RDF data graph. Every triple captures either (1) an entity's *attribute* value, (2) its *type*, (3) or a *relation* between an entity and another. In the last few years, large number of structured data (formerly stored in relational and XML databases) have been converted to RDF and published on the Web. Prominent examples include DBpedia[1] (the structured data counterpart of Wikipedia), as well as entity descriptions embedded in Web pages in the form of microformats[2] and RDFa[3]. The semantics captured by this data have been exploited in various tasks such as Web search: RDFa and mircoformats are used by Google and Yahoo! to provide rich snippets [4] for Web search results, which are based on structured descriptions of the entities embedded in the results.

One main problem towards the effective usage of structured data is integration. Besides schema matching, the other main problem in this regard is to resolve differences in the data representation of the same real-world object. It is called *instance matching* (also known as entity resolution, entity co-reference, or record linkage), which is about finding instances (e.g. `Publication`) that refer to the same object [1, 2, 3].

State-of-the-art approaches employ instance-matching rules to find instances that are the same. For example, n_3 and n_4 in Tbl. 1 may form a match, if we consider a instance-matching rule "two publications are identified as the same if they have similar `Title` and similar `Authors`". Typical approaches for learning instance-matching rules involve selecting a set of attributes, and for each attribute determining the best similarity function (e.g. Jaccard, Cosine, QGram etc. [4]) and threshold. For example, the rule above can be specified as "two publications are the same, if the *Jaccard* similarity of `Title` is greater than 0.8 and the *Cosine* similarity of `Author` is greater than 0.4". We call this type of rules *attribute-threshold instance-matching rule* (*aIR*), because it is satisfied only when the similarity of each attribute is higher than the threshold separately.

However there are three problems of aIR-based approaches. First, aIR-based approaches may fail to identify matching instance pairs if there are errors that occur in a sin-

[1]`http://dbpedia.org`
[2]`http://microformats.org/`
[3]`http://www.w3.org/TR/xhtml-rdfa-primer/`
[4]`http://www.google.com/webmasters/tools/richsnippets`

gle attribute leading to the similarity being incorrectly low. Second, these approach cannot effectively learn the correct rules without fine-tuning of various parameters. And finally, the learning cost is expensive because these methods are designed to search a large number of candidates for the best rule. The number of the candidates is decided by the number of attributes, similarity functions, and especially training examples. While more training data may lead to higher effectiveness of the learned rules, it also result in much more learning time.

In this paper, we propose an efficient approach of learning instance-matching rules to solve these three problems. To deal with the errors in attributes, we observe that although there may be errors in a single attribute, it is unlikely that the errors happens on every attribute. Therefore, we consider the similarity of two instances as the average similarity of a set of attributes. The two instances are considered as the same only when the average similarity is greater than the threshold. For example, instances n_3 and n_4 in Tbl. 1 are identified as a match, if we consider a instance-matching rule "two publications are the same, if the average of the *Jaccard* similarity of `Title` and the *Cosine* similarity of `Author` is greater than 0.6". We call this type of instance-matching rule as *mapping-threshold instance-matching rule* (*mIR*). Besides, the approach we proposed in this paper is almost free of parameters, which can easily estimate the value of the parameters from the training data and require not fine-tuning of them. Finally, we propose an efficient algorithm to learn instance-matching rules, which search the best mIR from a significantly smaller set of candidates. The number of candidates depends only on the number of attributes and similarity functions, while it is irrelevant to the number of training examples.

We observe that the average similarities of matches (and non-matches), that are calculated according to different attributes and similarity functions, subject to different probability distributions. We propose to calculate the matching and non-matching certainties, as the certainties to assign an instance pair to either a match or a non-match, from the cumulative probability of (dis)similarities of (non-)matches. Then a pair of instances will be considered as the same if its certainty to be a match is greater than that to be a non-match. Then the decision boundary can be calculated as the only one threshold, such that when a pair of instances has similarity that is greater than the threshold, its matching certainty is always greater than the non-matching certainty. Since there is only one mIR candidate for a specific combination of attributes and similarity functions (because we calculate only one threshold for the candidate), the number of mIR candidates is only relevant to the number of attributes and similarity functions. Based on this observation, we further propose an efficient algorithm that searches the best rule from all mIR candidates, and an efficient algorithm to execute the mIR.

Comparing to the state-of-the-art aIR learning approaches, our solution greatly improves the effectiveness as well as efficiency by up to 87% reduction of learning time. Moreover, the approach is also effective in the way that it can achieve stable results when the parameters are set with a large range of different values.

We organize the paper as follows. We formally define the problem of learning mapping-threshold instance-matching rule in Section 2. The algorithm for rule learning and execution are provided in Section 3 and Section 4 respectively. We present experiments in Section 5, related works in Section 6, and conclusions in Section 7.

2. INSTANCE MATCHING

We consider the kinds of data where instances are captured as sets of attribute values.

DEFINITION 1 (DATA). *The* data $N[a_1, a_2, \cdots]$ *comprises a set of instances. Each* instance $n \in N$ *is represented by a set* $\{a_1, a_2, \cdots\}$ *of attributes. The* value *of an* attribute a *of the instance* n *is denoted by* $n[a]$.

The problem tackled in this work is instance matching. Given a set of instances N, the goal is to assign a mapping $(n, n') \in N \times N$ to one of the two classes $\mathbf{M}^+$ and $\mathbf{M}^-$. The class $\mathbf{M}^+$ contains all the mappings that refer to the same real-world entities (matches), and the class $\mathbf{M}^-$ contains all the mappings that refer to different entities (non-matches).

2.1 Attribute-thresholded Instance Matching

Let $\{g_1, g_2, \cdots\}$ be a set of similarity functions such that $0 \leq g(n[a], n'[a]) \leq 1$. A higher score indicates a higher similarity between $n[a]$ and $n'[a]$. For the convenience of expression, we loosely use $g(a)$ to denote an association of a similarity function g with an attribute a. For example, we use $Jaccard(\texttt{Title})$ to denote the similarity of attribute `Title` that is calculated by *Jaccard* similarity function.

Existing aIR-based approaches are designed according to *attribute monotonicity*, which requires that "any pair of matching records have a higher similarity value than a non-matching pair on at least one attribute" [5]. If an instance matching problem is attribute monotonic, these approaches are then able to learn *attribute-threshold instance-matching rule* (*aIR*) to correctly identify all the matching instances.

DEFINITION 2 (ATTR.-THRESH. INST.-MATCH. RULE). *Given a set* $\{a_1, a_2, \cdots, a_d\}$ *of attributes, and a set* $\{g_1, g_2, \cdots, g_d\}$ *of similarity functions, let* $g_i(a_i) \geq \theta_i$ *denote a* similarity function predicate *where* $0 \leq \theta_i \leq 1$. *For any two instances* n *and* n', *the similarity function predicate returns true if* $g_i(n[a_i], n'[a_i]) \geq \theta_i$. *An attribute-threshold instance matching rule is a conjunction of similarity function predicates as* $\bigwedge_{i=1}^{d} g_i(a_i) \geq \theta_i$. *A pair of instances* n *and* n' *are considered as a match if they satisfy all the similarity function predicates in the rule.*

EXAMPLE 1. *Suppose an aIR* $Jaccard(\texttt{Title}) \geq 0.80 \wedge Cosine(\texttt{Author}) \geq 0.40$ *that is designed for the data in Tbl. 1. Because* $Jaccard(n_3[\texttt{Title}], n_4[\texttt{Title}]) = 1.00 > 0.80$ *and* $Cosine(n_3[\texttt{Authors}], n_4[\texttt{Authros}]) = 0.43 > 0.40$, *the instance pair* (n_3, n_4) *is identified as a match.*

Table 2: Similarities calculated according to $Jaccard(\texttt{Title})$, $Cosine(\texttt{Authors})$, and rule function $f = \frac{Jaccard(\texttt{Title}) + Cosine(\texttt{Authors})}{2}$ for the data in Tbl.1.

Mapping	(Non-)Matching	$Jaccard(\texttt{Title})$	$Cosine(\texttt{Authors})$	f	$1-f$
(n_1, n_2)	$\mathbf{M}^+$	0.28	1.00	0.64	0.36
(n_3, n_4)	$\mathbf{M}^+$	1.00	0.43	0.72	0.28
(n_5, n_6)	$\mathbf{M}^-$	0.33	0.50	0.42	0.58
(n_6, n_7)	$\mathbf{M}^-$	0.20	0	0.1	0.90

Table 1: A sample of publications taken from a real bibliographic database; correct mappings are (n_1, n_2) and (n_3, n_4).

ID	Title	Authors	Venue	Year
n_1	Environmental information systems	Oliver Guenther	SIGMOD	1997
n_2	Environment Information Systems - Guest Editor's Foreword	Oliver Guenther	SIGMOD Record	1997
n_3	Toward autonomic computing with DB2 universal database	S. S. Lightstone, G. M. Lohman, D. C. Zilio	SIGMOD Record	2002
n_4	Toward autonomic computing with DB2 universal database	Sam Lightstone, Guy Lohman, Danny Zilio	ACM SIGMOD Record	2002
n_5	An Overview of Data Warehousing and OLAP Technology	Surajit Chaudhuri, Umeshwar Dayal	SIGMOD Record	1997
n_6	Data Warehousing and OLAP for Decision Support (Tutorial)	S. Chaudhuri, U. Dayal	SIGMOD Conference	1997
n_7	Tracing the lineage of view data in a warehousing environment	Y. Cui, J. Widom, J. L. Wiener	TODS	2000

Existing approaches for learning aIR involve determining a set of attribute, and for each attribute the best similarity function and threshold [5, 1, 2, 6]. Comparing to the general machine learning techniques (e.g. SVM and decision tree [5, 6]), aIR can be more efficiently executed resulting from less calculation of similarity function predicates. Especially, it does not have to test all the similarity function predicates, if any one of the predicates in the rule return false.

- **aIR approaches is sensitive to errors in attributes.** Even if most attributes are the same, an aIR might still fails to identify a matching instance pair, due to a variety of errors (e.g. spelling errors, missing values etc.) that occur in a single attribute result in one of the similarity function predicate incorrectly returns false.

- **aIR approaches depends on fine-tuning of parameters.** Existing approaches requires fine-tuning of various parameters. For example, the method porposed by Chaudhuri et al. [5] needs the number of similarity function predicates and the number of rules to learn as parameters. And the approach proposed by Jiannan et al. [6] requires experts to manually select attributes, and for some of the attributes the similarity functions and thresholds, so that it can learn similarity functions and thresholds for the other attributes. Without the fine-tuning of such parameters, these approaches cannot learn effective rules.

- **The cost for learning aIR is expensive**: Existing approaches [5, 6] are designed to search a large number of candidates for the best rule. The number of the candidates is decided by the number of attributes, similarity functions, and especially training examples (see details in Appendix A). While more training data results in higher quality of the learned rules, it also leads to much more time for learning. Even though various methods being proposed to boost efficiency (e.g. greedy algorithm [5] or removing candidates composed of redundant thresholds and similarity functions [6] etc.), the cost for learning is still expensive.

Example 2. *Consider an aIR Jaccard*(`Title`) $\geq \theta_1 \wedge$ *Cosine*(`Authors`) $\geq \theta_2$ *for the data in Tbl. 1. As the similarities that are shown in Tbl 2, θ_1 and θ_2 must be set with small values that are not greater than 0.28 and 0.43 respectively, if the mIR can identify both matches (n_1, n_2) and (n_3, n_4) that are with spelling errors in* `Title` *and* `Authors` *respectively. However, the aIR designed in this way is incorrect, because it is unavoidable to assign the non-matching instance pair (n_5, n_6) to the class of match. In this circumstance, it is impossible to define an aIR that can correctly distinguish all matches and non-matches.*

Due to these drawbacks, aIR learning may not yield to good results in terms of both effectiveness and efficiency. In this paper, we adopt a different form of instance-matching rule that performs better when there are errors in attributes, and can be efficiently learned by an approach that is almost free of parameters.

2.2 Mapping-threshold Instance Matching

Instead of exploiting *attribute monotonicity*, we take use of the *mapping monotonicity* to solve the instance matching problem. We say an instance matching problem is *monotonic* if, for any matches (n_1, n'_1) and non-matches (n_2, n'_2), there exist a set of attributes such that the average similarity of these attributes of (n_1, n'_1) is greater than that of (n_2, n'_2). We observe that this mapping monotonicity exists more generally, because even though errors might occasionally occur in a single attribute, it is unlikely that they occur in every attribute of a match. Based on the mapping monotonicity, we formally define the *mapping-threshold instance-matching rule* (*mIR*) as follows:

Definition 3 (Map.-thresh. Inst- Match. Rule). *Given a set $\{a_1, a_2, \cdots, a_d\}$ of attributes and correspondingly a set $\{g_1, g_2, \cdots, g_d\}$ of similarity functions, an* rule function $f : N \times N \to [0, 1]$ *calculates the similarity of two instances as the average similarity of all the attributes, i.e. $f(n, n') = \frac{\sum_{i=1}^{d} g_i(n[a_i], n'[a_i])}{d}, n, n' \in N$. Given a threshold θ, an mapping-threshold instance-matching rule (mIR) is defined as a tuple $\lambda(f, \theta)$, such that two instance $n \in N$ and $n' \in N$ are considered as a match if $f(n, n') \geq \theta$, where $0 \leq \theta \leq 1$.*

Example 3. *Consider the average similarities that are calculated according to the rule function $f = \frac{Jaccard(Title)+Cosine(Authors)}{2}$ in Tbl. 2. Since both matching instance pairs (n_1, n_2) and (n_3, n_4) have greater similarities than the non-matching instance pairs (n_5, n_6) and (n_6, n_7), we can correctly identify all matches using the mIR $\frac{Jaccard(Title)+Cosine(Authors)}{2} \geq 0.6$.*

2.3 mIR Learning Problem

Consider a set of mappings M as training examples, which are composed of positive examples $M^+ \subseteq \mathbf{M}^+$, i.e. instance pairs that are known to be correct, and negative examples $M^- \subseteq \mathbf{M}^-$, i.e. instance pairs that are known to be incorrect. Obviously there is $M = M^+ \cup M^-$. Let Ψ be a set of mIRs that are learned from M, and let $M_\Psi \subseteq M$ be the instance pairs in M that satisfies any of the mIR in Ψ. Ideally we hope M_Ψ is exactly equal to M^+. However, in reality, M_Ψ may not only contain non-matching instance pairs, but also miss matching instance pairs that are failed to be identified by Ψ. To evaluate the quality of Ψ, we consider a general objective function $Q(\Psi, M^+, M^-)$. The less false positives, i.e. the less non-matching instance pairs that are included in M_Ψ, the higher $Q(\Psi, M^+, M^-)$; the less false negatives, i.e. the less matching instance pairs that are failed to be identified by Ψ, the higher $Q(\Psi, M^+, M^-)$. For example, F-measure is used as an general objective function $\frac{2pr}{p+r}$, where $p = \frac{|M_\Psi \cap M^+|}{|M_\Psi|}$ is precision and $r = \frac{|M_\Psi \cap M^+|}{M^+}$ is recall.

Now we formalize the problem of learning mIR for effective instance matching problem as follows:

Definition 4 (mIR-learning problem). *Given a set of positive examples M^+ and a set of negative examples M^-, the goal in the mIR learning problem is to learn Ψ, a set of mIRs, to maximize a pre-defined objective function $Q(\Psi, M^+, M^-)$.*

3. ALGORITHM FOR LEARNING MIR

In this section, we describe an efficient algorithm for learning a set of mIR. We first introduce an approach for the restricted version of the problem in which, only one mIR is learned. For a given rule function, we propose to calculate the matching and non-matching certainties, as the certainties to assign an instance pair to either a match or a non-match, from the cumulative probability of (dis)similarities of (non-)matches (Section 3.1). A pair of instances would be classified to matches when its matching certainty is greater than the non-matching certainty. Based on the estimation of (non-)matching certainty (Section 3.2), we learn the only threshold as the decision boundary, that if the similarity of an instance pair is greater than the threshold, its matching certainty is always greater than the non-matching certainty (Section 3.3). We then introduce the method for fast evaluation of the quality of a mIR candidate (Section 3.4), based on which an hill-climbing algorithm is proposed to learn the best mIR from a significantly smaller set candidates compared to the number of candidates in existing aIR-based approaches (Section 3.5). Finally we describe an approximation algorithm to learn a set of mIRs which maximize the pre-defined objective function (Section 3.6). The algorithm for the restricted version that learns the best mIR is one of our main contributions in this paper.

3.1 Certainty for Instance Matching

Let f be a rule function, henceforth, we refer the *similarity* of a pair of instances (n, n') as the value of the rule function $x = f(n, n')$, and the *dissimilarity* as $y = 1 - f(n, n')$. Obviously, there is $0 \leq x \leq 1$, $0 \leq y \leq 1$, and $x + y = 1$. For a pair of instances, the higher similarity x, the more likely it is assigned to $\mathbf{M}^+$; and the higher dissimilarity y, the more likely it is assigned to $\mathbf{M}^-$. However, even though the similarity (or dissimilarity) serves as evidences of how similar (or dissimilar) of two instances, we are still not certain to make a decision that how similar is similar. For examples, given a dataset with seldom errors, most matches may have very high similarity around 1. Therefore, when there are two instances with similarity 0.7, we may have low certainty to infer them as a match. On the other hand, given a dataset with various errors, we can have every high certainty to classify the instance pair as a match, because we know most matches in the dataset have similarity that are less than 0.7.

We exploit positive and negative examples to calculate the certainty of assigning a pair of instances to either $\mathbf{M}^+$ or $\mathbf{M}^-$. Intuitively, for an unlabeled instance pair with similarity value x (or dissimilarity value y), the more positive (or negative) examples we have observed that have similarities (or dissimilarities) less than x (or y), the more certainty we have to assign it to $\mathbf{M}^+$ (or $\mathbf{M}^-$). For example, when two instances have similarity value 0, there should be the lowest certainty to assign them to $\mathbf{M}^+$, because no (or few) positive examples have similarities that are less than or equal to 0. On the other hand, when they have similarity 1, the certainty of assigning it to $\mathbf{M}^+$ is the highest because all positive examples have similarities that are not greater than 1.

Definition 5 (Matching Certainty). *Let $X = \{x | 0 \leq x \leq 1\}$ be a random variable of similarities calculated by a rule function f. The matching certainty is the certainty of assigning an instance pair with similarity x to $\mathbf{M}^+$, which is calculated from the cumulative distribution function (CDF) of X as $F(x, \mathbf{M}^+) = Pr(X \leq x, \mathbf{M}^+)$, i.e. the cumulative probability of a pair of instances belongs to $\mathbf{M}^+$, and has similarity that is less than or equal to x.*

Obviously, the higher value of x, the higher $F(x, \mathbf{M}^+)$, and hence the higher matching certainty. The value of $F(x, \mathbf{M}^+)$ can also be viewed as the probability of a pair of instance is a matche, and has similarity that does not exceed x. Then if the similarity of an instance pair exceeds x, we can derive that the matching certainty of the instance pair should be also greater than $F(x, \mathbf{M}^+)$.

Similarly, we can define the non-matching certainty as follows:

Definition 6 (Non-matching Certainty). *Let $Y = \{y | 0 \leq y \leq 1\}$ be a random variable of dissimilarities calculated by $1-f$. The non-matching certainty is the certainty of assigning an instance pair with dissimilarity y to $\mathbf{M}^-$, which is calculated from the CDF of Y as $F(y, \mathbf{M}^-) = Pr(Y \leq y, \mathbf{M}^-)$, i.e. the cumulative probability of a pair of instances belongs to $\mathbf{M}^-$, and has similarity that is less than or equal to y.*

We can classify a pair of instances to either $\mathbf{M}^+$ or $\mathbf{M}^-$ by comparing the (non-)matching certainties. The mapping is classified to $\mathbf{M}^+$ if the matching certainty is greater than the non-matching certainty, otherwise it is classified to $\mathbf{M}^-$.

Example 4. *We calculate the matching and non-matching certainty for an instance pair with similarity* 0.7 *(and dissimilarity* 0.3*), according to training examples that are listed in Tab 2. Because only the positive example*

(n_1, n_2) among all four examples has similarity lower than 0.7, the matching certainty is 0.25. Similarly, because none of negative examples has dissimilarity lower than 0.3, the non-matching certainty is 0. As a result, we classify the instance pair to $\mathbf{M}^+$ because the matching certainty is greater than the non-matching certainty.

3.2 Estimate (Non-)Matching Certainty

In this section, we describe the approach to estimate matching certainty $F(x, \mathbf{M}^+)$ and non-matching certainty $F(y, \mathbf{M}^-)$. Since these two types of certainty can be estimated in the very similar way, we focus on introducing the estimation of matching certainty.

Let $q(x, \mathbf{M}^+)$ be the density distribution function of $F(x, \mathbf{M}^+)$ and $q(\mathbf{M}^+)$ be the prior probability of the class $\mathbf{M}^+$, we can rewrite $F(x, \mathbf{M}^+)$ as follows:

$$\begin{aligned} F(x \cap \mathbf{M}^+) &= \int_0^x q(t, \mathbf{M}^+)dt \\ &= \int_0^x q(t|\mathbf{M}^+)q(\mathbf{M}^+)dt \\ &= q(\mathbf{M}^+)\int_0^x q(t|\mathbf{M}^+)dt \\ &= q(\mathbf{M}^+)F(x|\mathbf{M}^+) \end{aligned} \tag{1}$$

where $F(x|\mathbf{M}^+)$ is the class-conditional cumulative probability function (CCF) over similarity. Note that the class prior $q(\mathbf{M}^+)$ can often be estimated simply from the fraction of positive examples in the training data. Eq. 1 converts the matching certainty estimation problem to estimate the CCF $F(x|\mathbf{M}^+)$. Obviously there are $F(0|\mathbf{M}^+) = 0$ and $F(1|\mathbf{M}^+) = 1$

Observation 1. *Suppose $X = \{x | 0 \leq x \leq 1\}$ be the random variable of similarities calculated from matches, X subjects to different CCF if the observed data of X is calculated according to different rule functions.*

Example 5. *We sampled positive and negative examples from a real bibliographic database provided by the benchmark [8]. The database has the same schema as the data in Tbl 1. As illustrated in Fig 1, consider the dissimilarities of negative examples, that are calculated according to two different rule functions $f_1 = \frac{Jaccard(\mathtt{Title})+Cosine(\mathtt{Authors})}{2}$ and $f_2 = QGram(\mathtt{Authors})$, as an example. Different rule functions refer to different distributions. Compared to the dissimilarities calculated according f_1, those calculated according to f_2 distribute more widely in the interval [0.9, 1], as shown in Fig. 1(b) and Fig. 1(d) respectively.*

Given a rule function f, we can obtain a set of observed similarities $\{x_1, x_2, \cdots x_k\}$ on the variable X, which are calculated from positive examples. Then the value of $F(x_i|\mathbf{M}^+)$ at the point x_i can be simply estimated as follows:

$$F(x_i|\mathbf{M}^+) = \frac{|\{x|x \leq x_i\}|}{k}, \text{for } x \in \{x_1, x_2, \cdots\ x_k\} \tag{2}$$

where $|\{x|x \leq x_i\}|$ is the amount of the observed data that is less than or equal to x_i.

The Eq. 2 can be calculated more efficient by ranking the observed similarities. When the observed similarities from

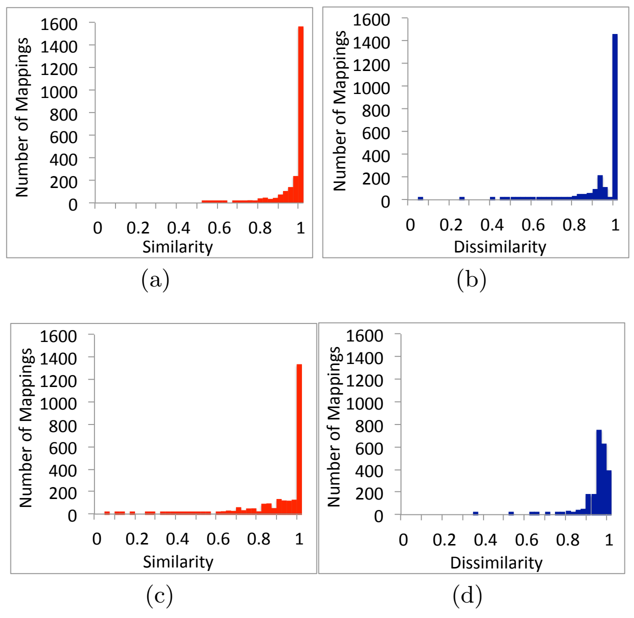


Figure 1: Consider two rule functions $f_1 = \frac{Jaccard(\mathtt{Title})+Cosine(\mathtt{Authors})}{2}$ and $f_2 = QGram(\mathtt{Authors})$. Figure (a) and (c) show the histograms for similarities of positive examples calculated according to f_1 and f_2 respectively. And Fig. (b) and (d) show the histograms for dissimilarities of negative examples calculated according to f_1 and f_2 respectively.

.

X are sorted in an ascending order, and let O_i is the rank number of x_i, the value of $F(x_i|\mathbf{M}^+)$ is estimated as follows:

$$F(x_i|\mathbf{M}^+) = \frac{O_i}{k} \tag{3}$$

we are then able to estimate $F(x|\mathbf{M}^+)$ for any unobserved similarity x in $[0, 1]$ by using Eq. 3. Let x_i and x_{i+1} be two adjacent observed similarities, for any unobserved similarity $x_u \in [x_i, x_{i+1}]$, $F(x_u|\mathbf{M}^+)$ can be estimated via linear interpolation as follows:

$$F(x_u|\mathbf{M}^+) = F(x_i|\mathbf{M}^+) + \frac{(F(x_{i+1}|\mathbf{M}^+) - F(x_i|\mathbf{M}^+))(x_u - x_i)}{x_{i+1} - x_i} \tag{4}$$

We call the CCF estimation by Eq. 4 as *interCCF*. The time complexity of interCCF is $O(\log k)$, where k is the number of training examples (see analysis in Appendix C). We observe that when there are a number of training examples available, the interCCF can reach high accuracy. Otherwise, because the real CCF usually does not simply follow a linear function, this estimation by the linear interpolation may largely deviate from the true value.

As a solution, we can also fit $F(x|\mathbf{M}^+)$ to the known cumulative probability distribution functions. Because the range of similarity is $[0, 1]$, we only consider the cumulative distributions that are defined on the same interval, such as Beta distribution, power-law distribution. Among all these distributions, we find power-law distribution is the most suitable for the instance matching problem in experiments, which is generally defined as follows:

$$F(x|\mathbf{M}^+) = ax^b, x \in [0, 1] \tag{5}$$

Where a and b are two parameters that need to be estimated from the training data. Since there is $F(1|\mathbf{M}^+) = 1$, we can directly figure out $a = 1$ by substituting $x = 1$ into Eq. (5). Then provided with the observed similarity set $\{x_1, x_2, \cdots, x_k\}$, we can estimate $b = \frac{\sum_1^k x_i F(x_i|\mathbf{M}^+)}{\sum_1^k x_i^2}$ via the least square method, where $F(x_i|\mathbf{M}^+)$ is calculated by Eq. (3). We call the CCF estimation by Eq. 5 as *powerCCF*. The time complexity of powerCCF is $O(k)$, where k is the number of training examples (see analysis in Appendix C).

3.3 Learn Threshold

Given a rule function f and an unlabeled instance pair (n, n') with similarity x and dissimilarity $y = 1 - x$, we can now classify (n, n') using a instance-matching decision rule that is defined based on the comparison of the matching certainty $F(x, \mathbf{M}^+)$ and the non-matching certainty $F(1 - x, \mathbf{M}^-)$, as follows:

$$(n, n') \in \begin{cases} \mathbf{M}^+ \text{ if } F(x, \mathbf{M}^+) \geq F(1-x, \mathbf{M}^-) \\ \mathbf{M}^- \text{ else} \end{cases} \quad (6)$$

This decision rule indicates that, if the matching certainty is greater than the non-matching certainty, (n, n') is assigned to $\mathbf{M}^+$, and vice versa. By substituting Eq. (1) into formula (6), the previous decision rule can finally be stated via CCF as follows:

$$(n, n') \in \begin{cases} \mathbf{M}^+ \text{ if } \frac{F(x|\mathbf{M}^+)}{F(1-x|\mathbf{M}^-)} \geq \frac{q(\mathbf{M}^-)}{q(\mathbf{M}^+)} \\ \mathbf{M}^- \text{ else} \end{cases} \quad (7)$$

Where the ratio $\frac{F(x|\mathbf{M}^+)}{F(1-x|\mathbf{M}^-)}$ can be calculated by either interCCF or powerCCF, and the *prior ratio* $\frac{q(\mathbf{M}^-)}{q(\mathbf{M}^+)}$ is required as a parameter, which can be simply estimated from the fractions of the training examples in each of the classes. In experiments, we will show that our technique can achieve high quality of the result without fine-tuning of this parameter, even if the prior ratio is set with a large range of different values.

However, the cost of executing the decision rule in Eq. 7 is still expensive, since the ratio $\frac{F(x|\mathbf{M}^+)}{F(1-x|\mathbf{M}^-)}$ has to be repeatedly calculated for every unlabeled instance pair. A simpler manner to make a decision according to Eq. 7 is to figure out the decision boundary θ, i.e. the threshold, so that any instance pair that has similarity greater than θ must be assigned to $\mathbf{M}^+$. In the example of Fig. 2, it corresponds to finding the value of x shown by the vertical dotted line.

PROPOSITION 1. *The threshold θ is calculated as the solution of the equation as follows:*

$$\frac{F(x|\mathbf{M}^+)}{F(1-x|\mathbf{M}^-)} = \frac{q(\mathbf{M}^-)}{q(\mathbf{M}^+)} \quad (8)$$

PROOF. The ratio $\frac{F(x|\mathbf{M}^+)}{F(1-x|\mathbf{M}^-)}$ is a monotonically increasing function, since $F(x|\mathbf{M}^+)$ and $F(1-x|\mathbf{M}^-)$ are monotonically increasing and monotonically decreasing respectively. Therefore, as the solution of Eq.8, θ must be the decision boundary for Eq. 7, since for any similarity value $x \geq \theta$ there must be $\frac{F(x|\mathbf{M}^+)}{F(1-x|\mathbf{M}^-)} \geq \frac{q(\mathbf{M}^-)}{q(\mathbf{M}^+)}$, and vice versa.

□

Alg. 1 shows the algorithm to find the threshold. Since it is difficult to directly calculate the solution of Eq. (8), Alg. 1 searches an approximate solution $\bar{\theta}$ that satisfies $|\theta - \bar{\theta}| < \epsilon$, where θ is supposed to be the exact solution of Eq. (8) and ϵ is the *difference restriction* that restrict the difference between the exact and the approximate solutions. In reality, we can set a relative small value of the difference restriction to guarantee that instance matching result will not be affected by it. For example, when the difference restriction is set to 0.001, an approximate threshold $\bar{\theta} = \theta \pm 0.001$ may result in the same instance-matching result as the exact threshold θ.

We apply a recursive binary search algorithm to find the approximate threshold $\bar{\theta}$. The algorithm returns if the approximate threshold equals to the exact threshold, or the difference between them is less than the pre-defined difference restriction ϵ (line 2). The complexity of Alg. 1 is $O(\log \frac{1}{\epsilon} \log k)$ for interCCF and $O(\log \frac{1}{\epsilon})$ for powerCCF, where k is the number of training examples (see analysis in Appendix C).

```
Algorithm 1: Learn Threshold
  Input: lower bound $x_l$, upper bound $x_u$, difference $\epsilon$,
         estimated distribution $F(x|M^+)$ and
         $F(1-x|M^-)$
  Result: Estimated threshold $\bar{\theta}$
1 $\bar{\theta} := \frac{(x_l + x_u)}{2}$;
2 if $\frac{F(\bar{\theta}|M^+)}{F(1-\bar{\theta}|M^-)} = \frac{q(M^-)}{q(M^+)}$ or $x_u - x_l < \epsilon$ then
3     return $\bar{\theta}$ ;
4 else if $\frac{F(\bar{\theta}|M^+)}{F(1-\bar{\theta}|M^-)} > \frac{q(M^-)}{q(M^+)}$ then
5     $x_u := \bar{\theta}$;
6 else
7     $x_l := \bar{\theta}$;
8 return
  $LearnThreshold(x_l, x_u, \epsilon, F(x|M^+), F(1-x|M^-))$;
```

3.4 Evaluate mIR Candidate

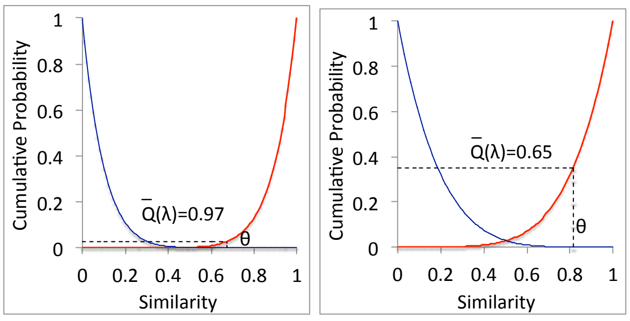
For a given rule function f, we can now generate a mIR candidate $\lambda(f, \theta)$ by combining f with the learned threshold θ. As discussed, mIR candidates can be directly evaluated by a pre-defined objective function, e.g. the F-measure. Then the best-evaluated mIR can be selected as the result. However, since it is required to enumerate all the training examples to calculate the matches, the cost of executing the objective function is expensive. In this section, we introduce the method for fast evaluation of the mIR candidate.

From the view of probability, the calculated mIR candidate can be evaluated by the probability of identifying the true positives. The higher this probability is, the better the mIR candidate is evaluated.

Since all the instance pairs with similarities in the interval $[\theta, 1]$ are considered as matches, the probability of identifying the true positives is the joint probability of a instance pair (n, n') being a match and the similarity x of (n, n') is in the interval $[\theta, 1]$, as follows:

$$Pr(\theta \leq x \leq 1, \mathbf{M}^+) = F(1, \mathbf{M}^+) - F(\theta, \mathbf{M}^+) \quad (9)$$

Then if substitute Eq. 1 into Eq. 9, the probability to



(a) (b)

Figure 2: PowerCCF estimation for matching certainty $F(x|\mathbf{M}^+)$ (red line) and non-matching certainty $F(1-x|\mathbf{M}^-)$ (blue line), where x is similarity. Fig.(a) and (b) illustrate the certainty distributions that refer to $f_1 = \frac{Jaccard(\texttt{Title})+Cosine(\texttt{Authors})}{2}$ and $f_2 = QGram(\texttt{Authors})$ respectively.

identify true positives can be expressed as:

$$Pr(\theta \le x \le 1, \mathbf{M}^+) = q(\mathbf{M}^+)(1 - F(\theta|\mathbf{M}^+)) \tag{10}$$

Considering the prior $q(\mathbf{M}^+)$ is the same for all mIRs, a mIR is evaluated as follows:

PROPOSITION 2. *The quality of a given mIR $\lambda(f,\theta)$ is evaluated by $\bar{Q}(\lambda) = 1 - F(\theta|\mathbf{M}^+)$. The higher $\bar{Q}(\lambda)$, the higher quality of $\lambda(f,\theta)$.*

The time complexity of mIR evaluation is $O(\ln k)$ for the interCCF and is $O(1)$ for the powerCCF, where k is the number of training examples (see analysis in Appendix C).

EXAMPLE 6. *Taking rule function $f_1 = \frac{Jaccard(\texttt{Title})+Cosine(\texttt{Authors})}{2}$ for a example. Suppose we have already estimated $F_1(x|M^+) = x^{9.17}$ and $F_1(y|M^-) = y^{9.99}$ by powerCCF, where x is the value of similarity and y is the value of dissimilarity. Note $y = 1 - x$, we can rewrite $F_1(y|M^-)$ as $F_1(1-x|M^-) = (1-x)^{9.99}$. The curves of $F(x|M^+)$ and $F_1(1-x|M^-)$ are illustrated in Fig. 2(a). Then suppose the prior ratio $\frac{q(M^-)}{q(M^+)} = 2000$, we can calculate the threshold $\theta = 0.6744 \pm 0.0001$ by Alg. 1 with a difference restriction $\epsilon = 0.0001$. Finally we get a mIR $\lambda_1(f_1, 0.6744)$, whose quality is evaluated as $\bar{Q}(\lambda_1) = 1 - F(0.6744|M^+) = 1 - 0.6744^{9.17} = 0.9730$.*

Similarly, for the rule function $f_2 = QGram(\texttt{Authors})$, we have $F_2(x|M^+) = x^{9.91}$ and $F_2(1-x|M^-) = (1-x)^{5.76}$, which are plotted in Fig. 2(b). Then we can learn the corresponding mIR as $\lambda_2(f_2, 0.8129)$, whose quality is evaluated as $\bar{Q}(\lambda_2) = 1 - F_2(0.8129|M^+) = 1 - 0.8129^{9.91} = 0.8717$.

By comparing $\bar{Q}(\lambda_1)$ and $\bar{Q}(\lambda_2)$, λ_1 is selected as the mIR with better quality.

3.5 Learn Single mIR

In this section, we introduce the algorithm to learn only one mIR, which searches the best one from a significantly smaller set of mIR candidates compared with existing aIR-based approaches. More specifically, let l, m and k be the number of attributes, similarity functions, and training examples respectively, the total number of mIR candidates is m^l in our work, while the total number of aIR candidates is $(m \cdot k)^l$ in state-of-the-art aIR-based approaches [5, 6](see details in Appendix A and Appendix B). For example, assuming an instance-matching problem with 4 attributes, 20 similarity, and 200 training examples, there would be 20^4 mIR candidates in our method, but $4,000^4$ aIR candidates in current aIR-based approaches.

A brute-effort algorithm is to enumerate all candidates and then select the one that maximizes the evaluation score. However even for the relative small size of candidates, this approach is still expensive. As shown in algorithm 2 we propose a hill-climbing algorithm to search the best mIR candidate, which mainly consists of two steps: mIR initialization and mIR optimization.

Step 1-mIR initialization: The algorithm starts from the mIR with the rule function that is composed of all attributes and for each attribute, the best-performed similarity function. Given two rule functions $f_i = g_i(a)$ and $f_j = g_j(a)$, if $\bar{Q}(\lambda(f_i,\theta_i))$ is greater than $\bar{Q}(\lambda(f_j,\theta_j))$, then the similarity function g_i is said to perform better than g_j on attribute a. For each attribute a_i, Alg. 2 (line 1-13) sort all similarity functions, that are stored in G_i, in the descending order, the best one at first and the worst one at last. The array I is used as an index to record the current selection of similarity function for each attribute. For example, when $I[2] = 3$, we bind attribute a_2 with the third similarity function in G_2[5]. Then the function `ConstructRulefunction` can construct a rule function by combining the attributes with the selected similarity function according to I(Alg. 3). At beginning, all the values in I are initialized with 1, so that we can select the best-performed similarity function for each attribute to initialize mIR. Considering a mIR may *not* include all attributes, we add a *null* to the end of G_i to allow the attribute a_i being ignored when it is combined with similarity function *null*.

EXAMPLE 7. *Considering two similarity functions $g_1 = Jaccard$ and $g_2 = Cosine$, and two attributes $a_1 = \texttt{Title}$ and $a_2 = \texttt{Authors}$. After sorting similarity functions for each attribute, we have $G_1 = \{Jaccard, Cosine, null\}$ and $G_2 = \{Cosine, Jaccard, null\}$, which means Jaccard performs the best on `Title`, and Cosine performs the best on `Authors`. In the mIR initialization step, we have $I = \{1, 1\}$ such that the top similarity functions in G_1 and G_2 are combined with attributes a_1 and a_2 respectively. In this way, we get the initial rule function $f = \frac{Jaccard(\texttt{Title})+Cosine(\texttt{Authors})}{2}$.*

Moreover, attribute a_2 is ignored if we set $I = \{2, 3\}$, since a_2 is combined with null, the third similarity function in G_2. In this way, we get a rule function $f' = Cosine(\texttt{Title})$ by combining the second similarity function in G_1 with a_1.

step 2-mIR optimization: We adopt a hill-climbing algorithm to search the best mIR. Intuitively, starting from the initial mIR, we adjust the similarity function for each attribute for achieving a higher evaluation score $\bar{Q}$. We chose one attribute a_i in each iteration (iteration in line 15-27), and construct different rule functions by associating a_i with each of the similarity functions in G_i (iteration in line 18-27). We then evaluate the mIR candidate that is created according to each rule function (line 19-23), and finally record the one that achieves better evaluation score $\bar{Q}$ (line 24-27). The iteration (iteration in line 15-27) terminates if the evaluation score cannot be improved any more.

[5]Note we use 1 as the index of the first element in an arraly.

EXAMPLE 8. *Assuming in Step 1, we have $I = \{1,1\}$, $G_1 = \{Jaccard, Cosine, null\}$ and $G_2 = \{Cosine, Jaccard, null\}$ for attribute $a_1 =$ `Title` and $a_2 =$ `Authors` respectively. In the first iteration we choose the attribute a_1 and change the value of the first element in index I to $I = \{2,1\}$ and $I = \{3,1\}$ to construct two rule functions $f = \frac{Cosine(Title)+Cosine(Authors)}{2}$ and $f' = Cosine(Authors)$. We then create the mIR candidates according to f and f', and select the one that is better evaluated. In the next iteration, we will select the other attribute a_2, and repeat above process again. This process will continue until the evaluation score cannot be improved any more.*

The time complexity of Alg. 2 is $O(tlm \log \frac{1}{\epsilon} \log k)$ for interCCF, and $O(tlmk)$ for powerCCF, where t, l, m, and k are the number of iteration, attributes, similarity functions, and training examples respectively, and ϵ is the difference restriction for threshold learning (see analysis in Appendix C).

```
Algorithm 2: Learn single mIR
   Input: difference $\epsilon$, a set $A$ of attributes, a set $G$ of
          similarity functions
   Result: best mIR $\lambda_{best}$
 1 $\mathbf{G} = \{\emptyset, \emptyset, \cdots\}$;
 2 for $i := 1$ to $|A|$ do
 3     $G_i := \emptyset$;
 4     $G_i := G_i \cup G$;
 5     $Sort(G_i)$;
 6     $G_i := G_i \cup \{null\}$;
 7     $\mathbf{G}[i] := G_i$;
 8 $I := \{1, 1, \cdots\}$;
 9 $f := ConstructRuleFunction(A, \mathbf{G}, I)$;
10 Estimate $F(x|M^+)$ and $F(1-x|M^-)$;
11 $\theta := LearnThreshold(0, 1, \epsilon, F(x|M^+), F(1-x|M^-))$;
12 $Q_{best} := 1 - F(\theta|M^+)$;
13 $\lambda_{best} := \lambda(f, \theta)$;
14 $improved := true$;
15 while $improved = true$ do
16     $improved := false$;
17     for $i := 1$ to $|A|$ do
18         for $j := 1$ to $|G| + 1$ do
19             $I[i] := I[i] + 1$;
20             $f := ConstructRuleFunction(A, \mathbf{G}, I)$;
21             Estimate $F(x|M^+)$ and $F(1-x|M^-)$;
22             $\theta := LearnThreshold(0, 1, \epsilon, F(x|M^+)$
               $, F(1-x|M^-))$;
23             $\bar{Q} := 1 - F(\theta|M^+)$;
24             if $\bar{Q} > Q_{best}$ then
25                 $\lambda_{best} := \lambda(f, \theta)$;
26                 $Q_{best} := \bar{Q}$;
27                 $improved := true$;
28 return $\lambda_{best}$;
```

3.6 Learn a Set of mIRs

Since one mIR may not cover all the positive examples, more mIR have to be used to maximize the objective function score. As discussed in previous research[5], the problem to learn a set of mIRs which maximize the objective function

```
Algorithm 3: Construct Rule Function
   Input: difference $\epsilon$, a set $A$ of attributes, a set $G$ of
          similarity functions, Index $I$
   Result: best mIR $\lambda_{best}$
 1 $N := 0$;
 2 for $i := 1$ to $|A|$ do
 3     $a_i := A[i]$;
 4     $G_i := \mathbf{G}[i]$;
 5     $g_i := G_i[I[i]]$;
 6     if $g_i \neq null$ then
 7         $N := N + 1$;
 8 return $f := \frac{\sum_{i:=0}^{|A|} g_i(a_i)}{N}$;
```

is NP-hard. To deal with this problem, we apply the similar greedy algorithm as previous solution[5]. The algorithm greedily select the best mIR each time, remove the positive and negative examples that are identified by the learned mIR, and then repeat this procedure on the remain examples until the objective function score cannot be improved anymore.

4. ALGORITHM FOR EXECUTING MIR

As discussed before, aIR can be efficiently executed mainly because it involves less similarity calculations. It does not have to test all the similarity function predicates, if any one of the predicates in the rule return false. Compared to aIR, since we calculate the similarity of two instances in a mIR as the average similarity of a set of attributes, the mIR may not be efficiently executed if all the corresponding similarities have to be calculated. However, we observe that some similarity calculations in mIR are unnecessary, since it is possible to make a decision only based on a part of similarities.

Let $\lambda : \frac{\sum_{i=1}^{d} g_i(a_i)}{d} \geq \theta$ be a mIR that involves d similarity functions, and assume we have already finished computing j $(j < d)$ similarity functions for a pair of instances (n, n'). The original mIR can be rewrite as follows:

$$\lambda : \sum_{i=1}^{j} g_i(a_i) + \sum_{i=j+1}^{d} g_i(a_i) \geq d\theta \tag{11}$$

where $sum_c = \sum_{i=1}^{j} g_i(a_i)$ is the sum of similarities that have already been computed, $sum_u = \sum_{i=j+1}^{d} g_i(a_i)$ is the sum of similarities that have not been calculated, and $d\theta$ is the threshold for the sum of all similarities. We observed that it is possible to make a decision only based on sum_c in two circumstances:

1. If $sum_c \geq d\theta$, (n, n') is classified to $\mathbf{M}^+$. Obviously, if we know the mIR has already been satisfied, it is not necessary to calculated sum_u any more (line 4-5 of Alg.4).

2. If $sum_c + d - j < d\theta$, (n, n') is classified to $\mathbf{M}^-$. Assuming each of the similarity function in sum_u can achieve the highest value of 1, the maximal value of sum_u is $d - j$. We can then estimate that the maximal sum of similarities for (n, n') is $sum_c + d - j$. Based on this estimation, we can directly classify (n, n') to $\mathbf{M}^-$ when $sum_c + d - j$ is less than the threshold $d\theta$ (line 6-7 of Alg.4).

Algorithm 4: Executing mIR

```
Input: mIR $\lambda = \frac{\sum_{i=1}^{d} g_i(a_i)}{d} \geq \theta$, instance pairs $(n, n')$
Result: whether $(n, n')$ satisfies the mIR
1 $sum := 0$;
2 for $j$:=1 to $d$ do
3     $sum+ = g_j(n[a_i], n'[a_j])$;
4     if $sum \geq d\theta$ then
5         return true;
6     if $sum + d - j < d\theta$ then
7         return false;
8 return $sum \geq d\theta$;
```

5. EXPERIMENTAL EVALUATION

To study the proposed solution, we employ a recent instance matching benchmark [8] that captures data from enterprise databases as well as synthetic data Restaurant. Compared against the state-of-the-art approaches *SiFi* [6] and *SVM* [9], our approach greatly improves efficiency by up to 87% reduction of time as well as result quality.

5.1 Dataset and Matching Task

We now briefly describe the datasets which cover the restaurant, bibliography and products domains. Tabel 3 provides an overview of the datasets.

Table 3: For each dataset pair: number of instances, all matches $\mathbf{M}^+$, and all non-matches $\mathbf{M}^-$.

Task	Datasets		Mapping Candidates	
	Dataset1	Dataset2	$\mathbf{M}^+$	$\mathbf{M}^-$
Rest	864		112	5,270
AB	1,081	1,092	1,097	7,040
AD	2,616	2,294	2,224	3,140

Restaurant(Rest). The restaurant dataset is available at OAEI 2010[6]. We manually removed the attribute `telephone` from the dataset because the dataset with it is too easy for instance matching task to be useful for comparing algorithms.

ACM-DBLP (AD). These datasets in the benchmark [8] include well-structured bibliographic data from DBLP and the ACM digital library. This one is manually created and thus, is of higher quality among all datasets. As a result, the matching task represented by this is of low difficulty.

Abt-Buy (AB). This matching task in the benchmark [8] is performed between instances of the product dataset from `http://abt.com` and `http://buy.com`. This dataset contains the most noises in the attribute values. Therefore, the matching task for this is the most difficult.

We adopt a preprocessing step, named blocking [4], to select candidate instance pairs that are most likely to be the same, as shown in Tbl 3. In the experiments, we classify these candidates to either $\mathbf{M}^+$ of $\mathbf{M}^-$, and compare the efficiency and effectiveness among different approaches.

5.2 Experimental Setting

System. In the experiments, we select two baseline methods, `SiFi` [6] and `SVM` (using LIBSVM libarary [9]). **Sifi** is a recent aIR learning approach that learns similarity functions and thresholds from positive and negative examples. It requires attributes to be manually set for each rule. Also, it requires the similarity functions and thresholds to be manually set for some attributes so that they can be learned for other attributes. `SVM` requires features as similarities of all attributes calculated by all similarity functions. We compare these approaches against our solutions using interCCF and powerCCF, called **interMIR** and **powerMIR** respectively.

Similarity. We selected 20 similarity functions in the experiments provided by an open-source Java package `SimMetrics`[7]. In case there is null value when comparing two values, we suppose the similarity is 0.

All experiments were run on a computer with one 2.4GHz Intel Core 2 Duo CPU, using 4GB of main memory, running Linux with kernel version 2.6.18.

5.3 Efficiency

We first compare against baseline approaches in term of efficiency. Table 4 shows the training and testing time for all four approaches, which use all the mapping candidates as training and testing data. We can see that `powerMIR` and `interMIR` run the fast in the training process. For example, on dataset `AB`, `powerMIR` requires only 59 seconds for training, which is only 13% of the time of `SiFi` and half of the time of `SVM`. `interMIR` is slightly outperform `powerMIR` because it is more efficient in CCF estimation (see time complexity analysis in Appendix C). The cost of `SiFi` is the most expensive in training because of the large number of aIR candidates. Even though `SiFi` proposed to eliminate candidates with redundant similarity functions and thresholds, the process of removing redundancy itself is still expensive in reality.

In the testing process, `SVM` is clearly the worst comparing to other methods. For example, `powerMIR` only spends 5 seconds on the task of `AD`, while `SVM` costs 78 seconds. As a general model, `SVM` spends most the time on similarity calculations, since it require similarities of all attributes that are calculated by all similarity functions as features. Especially, because we avoid unnecessary similarity calculations in mIR execution, our approaches can achieve same testing time as the aIR approach `SiFi`.

Table 4: Performance of training and testing in seconds.

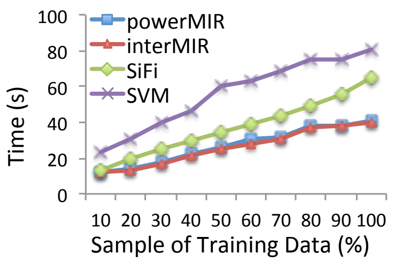
	Rest		AD		AB	
	train	test	train	test	train	test
powerMIR	37	3	55	4	59	8
interMIR	**36**	3	**50**	4	**57**	8
SiFi	54	4	342	4	518	9
SVM	59	60	85	78	109	103

We further vary the proportion of training data and plot the total running time of different methods in Fig. 3. The running time of our techniques increase the slowest when the size of the training data increases, mainly resulting from the number of mIR candidates is irrelevant to the number of training examples. However, since the number of candidates in `SiFi` increases fast with the increase of training examples, the time of `SiFi` increases the fastest.

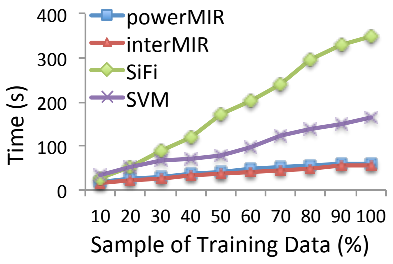
5.4 Effectiveness

[6]`http://oaei.ontologymatching.org/2010`

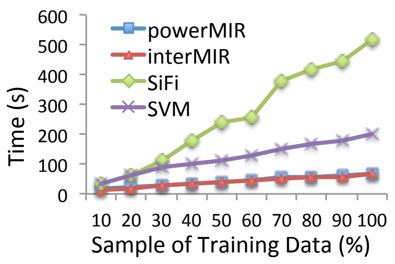
[7]`http://www.dcs.shef.ac.uk/~sam/simmetrics.html`



(a) Rest

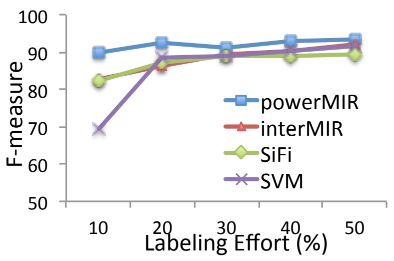


(b) AD

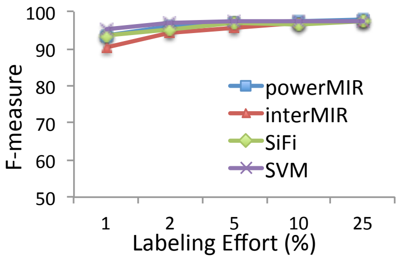


(c) AB

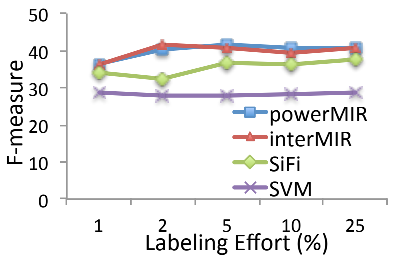
Figure 3: Comparison for the total running time with different proportion of training data.



(a) Rest



(b) AD



(c) AB

Figure 4: Evaluation result for different labeling effort

Table 5 shows the results of effectiveness based on the average of F-measure in 10 runs. For each run, we randomly select 30% of the mapping candidates as training data and use the remained as testing data. Our techniques achieve the best result on all the tasks. Note because we remove the attribute `Telephone` from `Rest`, this task becomes more difficult resulting in the worse result of `SiFi` compared to the result reported in the original paper. We also compare our solutions with `SiFi(auto)`, in which `SiFi` automatically learn the rules without manual tuning of parameters, such as the pre-defined attributes and similarity functions. We can see that its results becomes much worse than our approaches as well as `SiFi` with fine-tuning of parameters.

Table 5: Effectiveness of instance matching in terms of F-measure.

	Rest	AD	AB
powerMIR	**93.16**	**97.47**	**41.69**
interMIR	90.33	97.21	40.87
SiFi	88.92	96.20	37.52
SiFi(auto)	70.55	95.32	33.49
SVM	90.28	97.30	28.78

We further compare the F-measure against the baselines provided with different labeling effort, as shown in Fig. 4. The proportion of labeling effort varies between 10% and 50% in `Rest`, and between 1% and 25% in `AD` and `AB`. For `AD`, all the approaches can soon achieve stable results for small size (2%) of training data. And in `Rest` and `AB`, `powerMIR` is managed to maintain high quality of results for all different labeling efforts, while `SVM` performs the worst given small size of training data. We also observe that, `interMIR` is worse than `powerMIR` when it is provided with less training data. As discussed before, it is because `interCCF` leverages linear interpolation, which may deviate form the true values if the real CCF does not follow a linear function. However, when there are more training data available, this estimation becomes more accurate leading to `interMIR` achieving almost the same result as `powerMIR`. Therefore, we can directly apply `interMIR` when we are not clear about the data distribution, but have large amount of training data available.

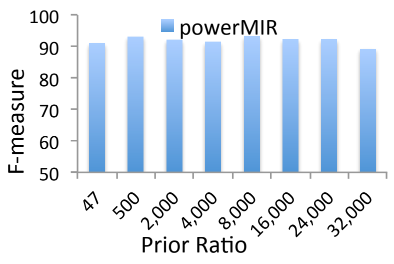
5.5 Parameter Sensitiveness

We now analyze the parameters. Throughout the paper, we have two parameters: the prior ratio $\frac{q(\mathbf{M}^-)}{q(\mathbf{M}^+)}$ for threshold learning according to Eq. (8); and ϵ as the difference restriction between the approximate and the real threshold in Alg. 1. We show that our approach can achieve stable result without fine-tuning of these parameters.

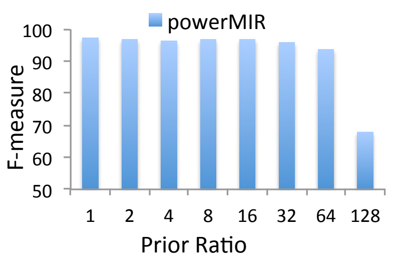
Figure 5 shows the result for different values of prior ratio. The results are stable when the prior ratio is set between 47 and 24,000 for `Rest` (the real prior ratio is 47), between 1 and 64 for `AD` (the real prior ratio is 1.4), and between 6 and 160 for `AB` (the real prior ratio is 6.4). Basically, a higher prior ratio determines a higher threshold. Therefore, a mIR with higher threshold may result in higher precision but less identified matches. However, since we keep adding the best evaluated mIR into the rules until the objective score cannot be improved, a higher prior ratio may result in a larger set of higher-precision mIRs. In this way, a matching instance pair that is fail to be identified by one mIR can be identified by another instead. Therefore, we can achieve stable results when prior ratio is set with values that are higher than the real one.

Figure 6 shows the results for different values of ϵ. Intuitively, the smaller ϵ, the higher accuracy. However, in reality there are no differences between the results leading from the real threshold and the approximated threshold with ϵ less than 0.01 for all the matching tasks.

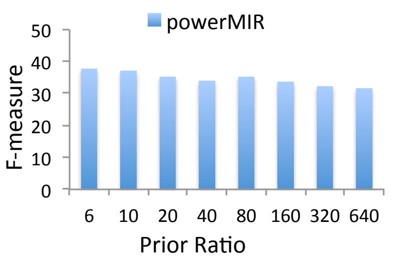
6. RELATED WORK



(a) Rest

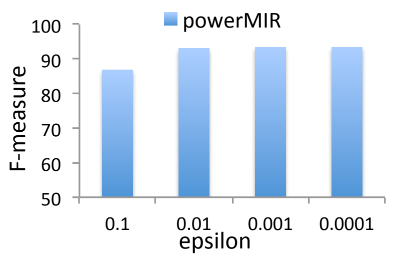


(b) AD

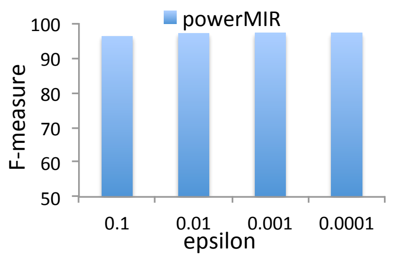


(c) AB

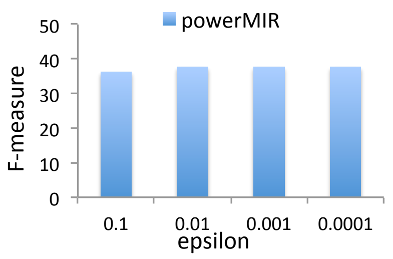
Figure 5: Influence of prior ratio



(a) Rest



(b) AD



(c) AB

Figure 6: Influence of ϵ

Instance matching (also know as entity resolution, entity co-reference, or record linkage), is about finding instances that refer to the same object. A high quality of instance matching result require the fine tunning of parameters selection, which are usually achieved with the help of machine learning techniques. Here, we provide a broader overview of machine-learning-based solutions for the instance matching problem.

Instance matching can be formulated as a standard classification task, where instance pairs are classified to either matches or non-matches. There are supervised learning techniques, which use decision trees [10, 11], Bayes decision rule [12] or SVM [13] to train classifier from the provided training examples. Among these techniques, SVM is known as the most effective approach.

There are also semi-supervised techniques based on probabilistic graphical models. It has been shown that many existing instance matching approaches can be specified in terms of Markov Logic formulas and reformulated as a Markov Logic learning problem [14]. However, this involves learning both the structure (formulas) and their weights. The current solution [14] still requires the logic for matching to be manually specified as Markov Logic formulas.

In contrast to those general machine-learning techniques, rule-based approaches are designed for specific instance matching problems. The instance-matching rules have advantages that they are explainable, and can be efficiently executed resulting from less similarity calculations. Existing rule-based approaches usually involve four subtasks to determine (1) (combinations of) attributes [15, 16, 17, 18, 5, 6], and for each of them, (2) the value representation function (e.g. selecting only the most important words in the values of `Title` and using only the first 3 tokens of each word) [11, 15, 16, 17, 18, 5], (3) the similarity functions [5, 6] and (4) the similarity thresholds [10, 11, 5, 6]. Recently Chaudhuri et al. [5] propose an algorithm for these four tasks based on a given set of positive and negative examples. Jiannan et al. [6] further improve the learning efficiency by eliminating rule candidates that are composed of redundant similarity functions and redundant thresholds. In this paper, we focus on task (1) (2) and (4).

7. CONCLUSION

For the problem of instance matching, we proposed an (almost) parameter-free and efficient approach to learn instance-matching rules by estimating matching and non-matching certainties. Comparing to state-of-the-art instance matching approaches, our solution greatly improves the efficiency (up to 87% reduction of time). At the same time, it is also effective, that when compared against SVM, that is currently known as the most effective approach for instance matching, we gain comparable or even better quality result in term of F-measure. Moreover, the approach can also achieve stable result when the parameters are set with a large range of different values. As future work, we aim to learn the weights of attributes, and the value representation functions for instance-matching rules.

8. REFERENCES

[1] S. Chaudhuri, V. Ganti, and R. Kaushik, "A primitive operator for similarity joins in data cleaning," in *ICDE*, 2006, p. 5.

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

[3] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," in *SIGMOD Conference*, 1995, pp. 127–138.

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

[5] S. Chaudhuri, B.-C. Chen, V. Ganti, and R. Kaushik, "Example-driven design of efficient record matching queries," in *VLDB*, 2007, pp. 327–338.

[6] J. Wang, G. Li, J. X. Yu, and J. Feng, “Entity matching: How similar is similar,” *PVLDB*, vol. 4, no. 10, pp. 622–633, 2011.

[7] M. Bilenko, R. J. Mooney, W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg, “Adaptive name matching in information integration,” *IEEE Intelligent Systems*, vol. 18, no. 5, pp. 16–23, 2003.

[8] H. Köpcke, A. Thor, and E. Rahm, “Evaluation of entity resolution approaches on real-world match problems,” *PVLDB*, vol. 3, no. 1, pp. 484–493, 2010.

[9] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[10] S. Tejada, C. A. Knoblock, and S. Minton, “Learning object identification rules for information integration,” *Inf. Syst.*, vol. 26, no. 8, pp. 607–633, 2001.

[11] ——, “Learning domain-independent string transformation weights for high accuracy object identification,” in *KDD*, 2002, pp. 350–359.

[12] I. Fellegi and A. Sunter, “A Theory for Record Linkage,” *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969. [Online]. Available: http://dx.doi.org/10.2307/2286061

[13] M. Bilenko and R. J. Mooney, “Adaptive duplicate detection using learnable string similarity measures,” in *KDD*, 2003, pp. 39–48.

[14] P. Singla and P. Domingos, “Entity resolution with markov logic,” in *ICDM*, 2006, pp. 572–582.

[15] M. Michelson and C. A. Knoblock, “Learning blocking schemes for record linkage,” in *AAAI*, 2006.

[16] M. Bilenko, B. Kamath, and R. J. Mooney, “Adaptive blocking: Learning to scale up record linkage,” in *ICDM*, 2006, pp. 87–96.

[17] W. Hu, J. Chen, and Y. Qu, “A self-training approach for resolving object coreference on the semantic web,” in *WWW*, 2011, pp. 87–96.

[18] Y. Ma and T. Tran, “Typimatch: type-specific unsupervised learning of keys and key values for heterogeneous web data integration,” in *WSDM*, 2013, pp. 325–334.

APPENDIX

A. NUMBER OF AIR CANDIDATES

For aIR-based approaches, the threshold for an attribute is learned from the similarities of *all* training examples that are calculated by *all* similarity functions. Assuming there are m similarity functions, and k training examples, there would be $m \cdot k$ different threshold candidates, which form $m \cdot k$ similarity function predicates for each attribute.

aIR candidates can be generated as conjunctions of similarity function predicates of all attributes. For examples, assume there are two attributes a_1 and a_2, and for each attribute there are $m \cdot k$ similarity function predicates. We can generate aIR candidates by first selecting one of the similarity function predicate of a_1, and then creating a conjunctions with each of the similarity function predicates of a_2. In this way, we can generate $(m \cdot k)^2$ different aIR candidates. In general, assume there are l attributes, the total number of aIR candidates is $(m \cdot k)^l$.

B. NUMBER OF MIR CANDIDATES

For our technique, the number of mIR candidates equals to the number of rule functions, because for each rule function we construct only one mIR candidate.

A rule function involves two different elements: (1) a set of similarity functions, and (2) a set of attributes. For examples, if we consider a set $\{g_1, g_2\}$ of two similarity functions, and a set $\{a_1, a_2, a_3\}$ of three attributes, then we get a total of 2^3 rule functions by combing each attribute with every similarity functions, such as $f_1 = \frac{g_1(a_1)+g_1(a_2)+g_1(a_3)}{3}$. In general, assume there are l attributes and m similarity functions, the number of rule functions is l^m, which is the same with the number of mIR candidates.

C. TIME COMPLEXITY ANALYSIS

Assume there are l attributes, m similarity functions, and k training examples, we analyze the time complexity of our approach as follows:

CCF estimation: Because interCCF only require the observed similarities being sorted, the time complexity of interCCF is equal to the time complexity of sorting, which is $O(\log k)$. And because powerCCF require enumerating all train examples to estimate the parameters, its time complexity is $O(K)$.

CCF value calculation: When calculate the value of CCF for a unobserved similarity, interCCF require a binary search to find the nearest observed data for interpolation. Therefore, the time complexity of CCF value calculation for interCCf equals to the time complexity of binary search, as $O(\log k)$. For powerCCF, because CCF is directly calculated according to Eq. 5, the time complexity is $O(1)$.

Learning threshold: Given a difference restriction ϵ, Alg. 1 actually search the approximate threshold from maximal $\frac{1}{\epsilon}$ different values. The number of comparisons required by Alg. 1 is $\log \frac{1}{\epsilon}$. Noting the ratio $\frac{F(\bar{\theta}|M^+)}{F(1-\bar{\theta}|M^-)}$ is calculated in each comparison, we also take the complexity of CCF value calculation into account. Therefore, the overall complexity of learning threshold is $O(\log \frac{1}{\epsilon} \log k)$ for interCCF and $O(\log \frac{1}{\epsilon})$ for powerCCF.

mIR Evaluation: Since the evaluation score of a mIR is calculated as $\bar{Q} = 1 - F(\theta|\mathbf{M}^+)$, the time complexity of mIR evaluation equals to the time complexity of CCF value calculation.

Learn single mIR: For each iteration of Alg. 2, we evaluate different mIR candidates that are created by keep changing similarity functions for every attribute. Then there are total lm mIR candidates to be evaluated in each iteration. Such iteration will continue t times until the terminate condition is satisfied. So the overall number of mIR candidates that are examined is tlm. Because for each mIR we execute CCF estimation, threshold learning, and mIR evaluation sequentially, the time complexity for this process is determined by the highest time complexity of each step, which is $O(\log \frac{1}{\epsilon} \log k)$ for interCCF and $O(k)$ for powerCCF. In sum, the overall time complexity of learning single mIR is $O(tlm \log \frac{1}{\epsilon} \log k)$ for interCCF and $O(tlmk)$ for powerCCF.