A Graph-Theoretic Fusion Framework for Unsupervised Entity Resolution

Dongxiang Zhang [#1], Long Guo [†2], Xiangnan He [*3] Jie Shao [#4], Sai Wu [$5], Heng Tao Shen [#6]

[#] Center for Future Media and School of Computer Science &Engineering, UESTC, China
[†] Key Lab of High Confidence Software Technologies (MOE), Peking University, China
[*] School of Computing, National University of Defense Technology, China
[$] College of Computer Science, Zhejiang University, China
{zhangdo,shaojie}@uestc.edu.cn guolong@pku.edu.cn xiangnanhe@gmail.com
wusai@zju.edu.cn shenhengtao@hotmail.com

***Abstract*—Entity resolution identifies all records in a database that refer to the same entity. The mainstream solutions rely on supervised learning or crowd assistance, both requiring labor overhead for data annotation. To avoid human intervention, we propose an unsupervised graph-theoretic fusion framework with two components, namely ITER and CliqueRank. Specifically, ITER constructs a weighted bipartite graph between terms and record-record pairs and iteratively propagates the node salience until convergence. Subsequently, CliqueRank constructs a record graph to estimate the likelihood of two records resident in the same clique. The derived likelihood from CliqueRank is fed back to ITER to rectify the edge weight until a joint optimum can be reached. Experimental evaluation was conducted among 14 competitors and results show that without any labeled data or crowd assistance, our unsupervised framework is comparable or even superior to state-of-the-art methods among three benchmark datasets.**

I. Introduction

Entity resolution, also known as duplicate record detection, is a fundamental problem in data integration and data cleaning. Given a database with ambiguous and error-prone records, it identifies clusters of records referring to the same real-world entity. Early research works [1], [2], [3] are dedicated to devise various string-based distance functions, either character-based [1] or token-based [2], to measure pair-wise record similarity. Two items are considered to be an identical entity if their textual similarity exceeds a predefined threshold. However, the effectiveness of these unsupervised approaches is limited. In addition, as emphasized by Bilenko et al. [3], there does not exist a single metric for all data sets.

To overcome the limitations of unsupervised approaches, machine learning based techniques [4], [5], [6], [7] became popular and improved the accuracy to certain extent. The entity resolution problem is viewed as a binary classification task. With hand-crafted features, standard classifiers such as SVM or naive Bayesian can be directly applied. One severe problem with the supervised learning techniques is the requirement for a good number of training instances. It entails considerable manual overhead to label the positive and negative examples. In addition, the number of non-matches far exceeds the number of matches. How to set an appropriate ratio between positive and negative samples becomes another challenge.

Corresponding Author: Heng Tao Shen

With the availability of affordable crowd workers in Amazon Mechanical Turk, recent research focus has shifted to utilize human intelligence to help verify uncertain pairs [8], [9], [10], [11], [12], [13]. They use machines to conduct an initial and coarse filtering based on a simple distance measure to remove pairs unlikely to match. For instance, Jaccard similarity was used in [10], [12] and a threshold of 0.3 was set to filter unpromising pairs. Then, crowd workers were used to verify the most promising pairs. To avoid enumerating all record-pairs to the crowd, which can be prohibitively expensive and time-consuming, they are focused on devising optimal strategies to select the record-pairs that are most worthy of verification. The objective was to minimize the number of questions asked while achieving high accuracy.

In this paper, we focus on unsupervised entity resolution that does not require preparing a labeled training dataset for supervised learning or paying crowd workers to verify the enumerated candidate pairs. We propose a new graph-theoretic fusion framework that provides an accurate estimation of the probability of two records referring to the same entity. We observed that, in a specific domain, terms usually exhibit different discrimination power, which should be taken into account in the textual similarity measure. For instance, terms about brand and model are more important than others in distinguishing E-commerce products, while telephone number and address are more discriminative for restaurants. An "intelligent" algorithm should be able to automatically differentiate the importance of terms from data, and consequently, provide high-quality entity resolution. It is worth noting that even though IDF (inverse document frequency) can also set higher weight to rare terms such as product models, the measure is not able to differentiate the truly discriminative terms and noisy terms with low frequency in the dataset. This observation motivated us to propose an enhanced content-based similarity measure that can accurately identify entities

by discriminating the importance of terms from data; and most importantly, our method works in a fully unsupervised way. We first construct a new type of bipartite graph to model the relationship between terms and record–record pairs, developing an *Iterative Term-Entity Ranking* (ITER) algorithm to jointly estimate the importance of terms and similarity of record-pairs. We will show in the experiments that our proposed method is more effective than existing textual similarity measures and random-walk based term weighting strategies.

Our next contribution is to derive the confidence of two records belonging to the same entity based on the topological property. In case of an ideal textual similarity measure, two records r_i and r_j referring to different entities should be located at different cliques, which are unreachable from each other. On the contrary, if r_i and r_j refer to the same entity, they should be in the same clique and reachable from each other. From the random-walk perspective, if we start a random walk from one record r_i, it will be very likely to reach the other record r_j within a fixed number of steps. We leverage this key insight to develop CliqueRank to identify the matching probability. This procedure can be viewed as a normalization step that transforms similarity scores to matching property with the range $[0, 1]$, from which it is easier to determine a universal threshold for different domains to judge whether two records are matching.

Finally, the enhanced textual similarity and pair matching probability can reinforce each other in our fusion framework. Considering that the matching probability of record-pairs is helpful to determine the discrimination power of terms — for example, the shared terms of non-matching pairs should be punished with a lower importance — we further feed the output of CliqueRank into the ITER algorithm. As such, our final solution iterates ITER and CliqueRank algorithms until a stable state is reached.

We conduct extensive experiments on three popular datasets from different domains. Without any labeled data or crowd-assistance, our unsupervised framework can achieve comparable or superior performance over the state-of-the-art solutions. All the datasets and implementations of our proposed algorithms are publicly accessible at Github[1].

II. Related Work

Due to space limit, we only review resolution methods that are distance-based, learning-based and crowd-based. A comprehensive survey on this topic is available in [14].

A. Distance-based Methods

The early efforts emphasized on finding or developing a string-based distance (or similarity) metric to detect duplicate records. For instance, edit distance was used in [1] and TF-IDF distance metric was proposed in [2]. Cohen et al. conducted an experimental study to compare several types of distance metrics for name-matching tasks [15]. Two items are considered to be identical entity if their textual similarity exceeds a predefined threshold. Even though simple and scalable, these approaches are not effective in practice. In addition, they are not adaptive: no single metric is suitable for all data sets [3] and it is non-trivial for the distance-based techniques to define an appropriate matching threshold.

B. Learning-based Methods

To overcome the limitations of unsupervied distance-based methods, supervised learning was used to learn the parameters of string-edit distance metrics [16], [17] or combine the results from different distance metrics [18], [19]. In [5], the entity resolution was modeled as a classification task and solved with SVM. One severe problem with the supervised learning techniques is the requirement for a large number of training instances. In addition, the number of non-matches far exceeds the number of matches, which also brings challenges in preparing the training dataset.

C. Crowd-based Methods

Recent research interests on this topic have been shifted to crowd-assisted solutions in the database community. CrowdER [8] was the first work to systematically adopt a hybrid human-machine approach in which machines were used for a coarse filtering based on simple string-based distance metrics, and then human intelligence was leveraged for verification of the promising but uncertain record-pairs. ZenCrowd [20] is an alternative effort to combine both probabilistic reasoning and crowdsourcing techniques. Although CrowdER and ZenCrowd can achieve a relatively promising precision, they incur considerable amounts of crowdsourcing overhead. There were several works [9], [11], [10] attempting to reduce the number of verification tasks sent to the crowd but they compromised the accuracy. The state-of-the-art work [12] that achieves the highest accuracy is an adaptive approach which uses correlation clustering to reduce the financial costs and improve the accuracy. Following this work, Chai et al. proposed a partial-order approach [13] to significantly reduce the candidate pairs delivered for crowd-task while not sacrificing the quality. Fan et al. studied the problem of entity collection with the help of crowd workers [21].

III. Graph-Theorectic Baselines

PageRank [22] and SimRank [23], as two most representative graph-theoretic algorithms, have been successfully applied to many NLP and IR problems [24]. PageRank is effective in measuring the node authority by counting the number and quality of incoming edges. Intuitively, an important or authoritative hub is connected by a large number of high-quality neighbors. SimRank is useful in measuring the pair similarity derived directly from the

[1]https://github.com/uestc-db/Unsupervised-Entity-Resolution

topological structure. It is defined in a recursive manner, i.e., two nodes are similar if they are connected to similar nodes. In this section, we present three types of graph-theoretic baseline algorithms by appling PageRank or SimRank on different graph models.

A. SimRank on Bipartite Graph

Bipartite graph is a popular data model that captures the relationship between two domains of data and frequently used in user-item recommendation [23]. In the Bipartite SimRank proposed in [23], an item-purchasing graph is built to estimate the similarity between two users in a recommender system. Two persons A and B are similar if their purchased item sets $O(A)$ and $O(B)$ are similar. Analogously, as shown in Figure 1, we can construct a bipartite graph between records and terms and claim that two records are similar if they contain similar terms. Formally, the similarity $s_b(r_i, r_j)$ between two records r_i and r_j is defined as:

$$s_b(r_i, r_j) = \frac{C_1}{|O(r_i)||O(r_j)|} \sum_{t_i \in O(r_i)} \sum_{t_j \in O(r_j)} s_b(t_i, t_j) \quad (1)$$

where C_1 is a decay factor, $O(r_i)$ contains out-neighbors of r_i and $s_b(t_i, t_j)$ is the similarity score between two terms, which can be defined in the same recursive way:

$$s_b(t_i, t_j) = \frac{C_2}{|I(t_i)||I(t_j)|} \sum_{r_i \in I(t_i)} \sum_{r_j \in I(t_j)} s_b(r_i, r_j) \quad (2)$$

where C_2 is also a decay factor and $I(t_i)$ contains incoming-neighbors of t_i. Finally, we can derive the pair similarity scores by iterating these two equations until convergence is reached. Two records are said to refer to the same entity if their topological similarity $s_b(r_i, r_j)$ exceeds a pre-defined threshold.

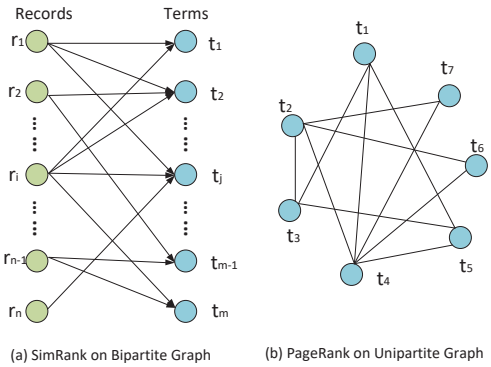


Figure 1. Graph models in graph-theoretic baselines.

B. PageRank on Unipartite Graph

An alternative direction is to derive a new textual similarity measure by replacing the TF-IDF term weighting strategy with graph-theoretic approaches [25], [26], [27], [28]. For example, TextRank [25] constructs an undirected term graph, as depicted in Figure 1, whose nodes are terms and edges correspond to the co-occurrence of two terms within a fixed-size sliding window in a document. Then, PageRank based text surfing model is applied to estimate the term importance in the domain. The same graph model can also be found in TW-IDF [29] for effective document retrieval. TW-IDF is essentially a variant of TF-IDF by replacing term frequency with term salience derived from PageRank.

Our second baseline algorithm adopts TW-IDF as a new textual similarity measure by treating each textual record as a paragraph in a document and constructing the associated term co-occurrence graph. The salience of term t_i is derived by

$$s(t_i) = (1 - \phi) + \phi \sum_{t_j \in N(t_i)} \frac{s(t_j)}{|N(t_i)|} \quad (3)$$

where ϕ is a damping factor which is generally set to 0.85 and $N(t_i)$ denotes the neighboring terms in the undirected term graph. Then, the TW-IDF based textual similarity $s'(r_i, r_j)$ between two records is defined as the weight sum of their common terms:

$$s_u(r_i, r_j) = \sum_{t \in r_i \wedge t \in r_j} s(t) \cdot \log \frac{n+1}{df(t)} \quad (4)$$

where n is the total number of records in the dataset and $df(t)$ is the number of records containing t. If two records do not share common terms, their similarity is defined as 0. By defining an appropriate threshold, we consider two records referring to the same entity if their textual similarity exceeds the threshold.

C. Hybrid Similarity Measure

We can see that Equation 1 actually measures the topological similarity between two records. In the meanwhile, Equation 4 captures the textual similarity. A straightforward fusion framework is to adopt linear combination to take into account these two types of similarity at the same time:

$$s_h(r_i, r_j) = \beta \cdot s_b(r_i, r_j) + (1 - \beta) \cdot s_u(r_i, r_j) \quad (5)$$

The parameter β can be tuned for different domains. If the textual similarity plays a more important role, we decrease the value of β. Otherwise, β is increased to prefer topological similarity.

IV. Unsupervised Fusion Framework

In this section, we propose a novel fusion framework to take into account content-based similarity and topological structure simultaneously. Our fusion framework exhibits three major differences compared to the naive linear combination presented in Section III-C.

1) Instead of calculating term salience in a crafted term graph, we are interested in incorporating term discrimination power when measuring textual similarity. Note

that these two ranking schemes can be conflictive in certain cases and details will be explained in Section V.

2) Instead of measuring topological similarity from a bipartite graph, we leverage the topological structure in a different way. In particular, we construct a weighted record graph with the objective of estimating the *probability* of two records referring to the same entity.
3) Unlike the rigid combination in Equation 5, our proposed fusion framework allows the content-based similarity and the pair matching probability to reinforce each other and improve the accuracy of entity resolution.

Figure 2 summarizes the workflow of our proposed method. We first construct a new bipartite graph to model the relationship between terms and *record–record pairs* and develop an Iterative Term-Entity Ranking (ITER) algorithm to jointly estimate the discrimination power of terms and textual similarity of record-pairs. With the pair similarity scores derived by the ITER algorithm, we construct another weighted record graph with the objective of leveraging its topological structure to estimate the probability of two records belonging to the same entity. Intuitively, a group of matching records should share high similarity scores and low similarity for non-matching records. In other words, if we start a random walk process from one record r_i and choose the next node according to the similarity score, the matching probability can be explained as how likely we can reach a matching record r_j within a fixed number of steps. This observation renders us to propose the CliqueRank algorithm to estimate the matching probability via graph-theoretic approaches.

Finally, the enhanced content-based similarity and pair matching probability can reinforce each other in our fusion framework. The shared terms of non-matching pairs should be punished with a lower importance as they appear in many pairs with low matching probability. The weight of discriminative terms that only occur in matching pairs would be promoted because these pairs are likely to have high similarity scores and their estimated matching probability would be close to 1. Our fusion framework iterates ITER and CliqueRank algorithms until stableness is reached for the update of term weight and record similarity.

It is also worth noting that we can directly use the matching probability, naturally ranged in $[0, 1]$, to determine whether two records belong to the same entity. Such a new criterion is more universal and straightforward to explain.

V. Learning Record Similarity

Instead of manually crafting a string-based distance measure, we propose an unsupervised computational method that automatically learns the similarity of records from data. We begin by defining an enhanced content-based similarity measure based on discrimination power, followed by constructing a bipartite graph that models the relationship between terms and record-pairs. After that, we elaborate the ITER algorithm that simultaneously estimates the importance of terms for entity resolution and the similarity of record-pairs. Finally, we prove the convergence of our ITER algorithm. The notations frequently used in this paper are summarized in Table I.

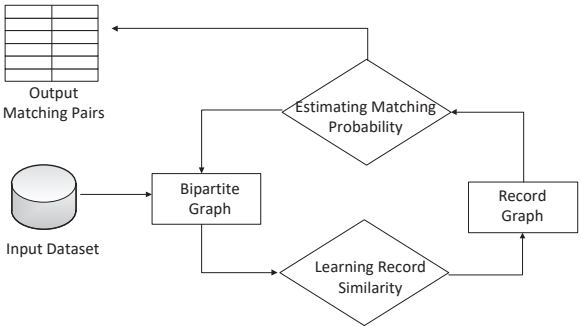


Figure 2. An overview of our framework for unsupervised entity resolution.

Table I
Notation Table

r_i	text record containing a set of terms
n	number of records in the dataset
m	number of unique terms in the dataset
$s(r_i, r_j)$	similarity score between r_i and r_j
$p(r_i, r_j)$	probability of r_i and r_j referring to the same entity
$p(r_i \to r_j)$	the probability of stepping to r_j in random walk
x_t	term weight or discrimination power of t
N_t	number of records containing term t
P_t	pairs of records sharing term t, i.e., $P_t = N_t \cdot (N_t - 1)/2$
η	matching probability threshold
α	exponent in the non-linear normalization of random walk
M	number of random walk based sampling in RSS
S	maximum number of steps in a random walk

A. Enhanced Content-based Similarity

The conventional TF-IDF measure assigns higher weight to frequent terms in a document (record) and punishes common or stop words using the factor of inverse document frequency [30]. Similarly, the TW-IDF measure also prefers frequent terms in the record corpus as they would co-occur with many other terms in the sliding window and serve as hubs in the crafted term graph. However, we observed that discriminative terms, that are really important when determining matching pairs, may not have high TF scores. For example, the term frequency of product models in the dataset such as “pslx350h” and “tu1500rd” may occur only once in a record, but they are highly discriminative in judging the identicalness of two products. Careful readers may argue that the factor of IDF in TF-IDF or TW-IDF can also help promote the weight of these product model terms. However, IDF is a general measure and cannot differentiate the truly discriminative terms and noisy terms with low frequency in the dataset.

In this paper, we propose an enhanced content-based similarity measure based on the discrimination power of terms. The intuition is that if a term only occurs in a group of matching records, then we consider the term as highly discriminative. Examples include product models for electronic devices and telephone numbers for restaurants. On the contrary, a common term without discrimination power is likely to be shared by many non-matching record pairs with low similarity score and matching probability. These two properties essentially differentiate our similarity measure from TF-IDF or TW-IDF. Let n be the number of records in a dataset and suppose term t is contained in a set of N_t records $\{r_1^t, r_2^t, \ldots, r_{N_t}^t\}$. We formally define our novel definition of the term weight x_t and record similarity score $s(r_i, r_j)$ in a recursive manner:

$$x_t = \sum_{i \neq j} \frac{p(r_i, r_j) s(r_i, r_j)}{P_t} \quad (6)$$

$$s(r_i, r_j) = \sum_{t \in r_i \wedge t \in r_j} norm(x_t) \quad (7)$$

where there involve $P_t = \frac{N_t \cdot (N_t - 1)}{2}$ pairs with different records and $p(r_i, r_j)$ indicates the probability of r_i and r_j ($i \neq j$) referring to the same entity. This probability, which will be introduced in Section VI, is initialized to 1 before it can be derived from CliqueRank. In Equation 6, a high x_t is assigned to term t (e.g., "pslx350h" and "tu1500rd") if all the record pairs sharing the term refer to the same entity. In other words, these pairs have high similarity score $s(r_i, r_j)$ and high matching confidence $p(r_i, r_j)$ at the same time. The denominator P_t is a normalization factor to punish frequently shared terms that are not discriminative (e.g., "dollar" and "device"). Recursively, the record similarity measure $s(r_i, r_j)$ in Equation 7 is defined in a straightforward fashion as the sum of normalized term weight. Based on the definition, $s(r_i, r_j)$ is set to 0 if r_i and r_j do not share any term. This is reasonable because we resolve entities from text-informative records. Each matching pair is likely to share a considerable number of discriminative terms.

B. Bipartite Graph Model

Intuitively, if two records share more discriminative terms, they are more likely to refer the same entity; and vice versa. To capture this, we use a bipartite graph to capture the relationship between terms and record pairs. Figure 3 shows an example, where there are two types of nodes — one is *term node* and the other is *pair node*, which denotes a pair of records.

Let t be a term node and (r_i, r_j) be a pair node. Nodes t and (r_i, r_j) are connected if and only if term t appears in both records of r_i and r_j. If a pair (r_i, r_j) does not share any term, we consider the pair of records not matching and exclude it from the bipartite graph. We associate a weight parameter for each node. For a term node t, its weight x_t indicates its power in discriminating entities; for a pair node (r_i, r_j), its weight $s(r_i, r_j)$ denotes the similarity score for the corresponding pair — the higher $s(r_i, r_j)$ is, the more likely pair (r_i, r_j) represents the same entity.

The two types of nodes are connected via weighted edges. The weight $p(r_i, r_j)$ is the estimated matching probability between r_i and r_j. It controls the amount of weight transferred from pair nodes to term nodes. If two records r_i and r_j are not matching, then we consider their shared terms are not contributing for this pair and their $s(r_i, r_j)$ will not be propagated back to adjust the term weight.

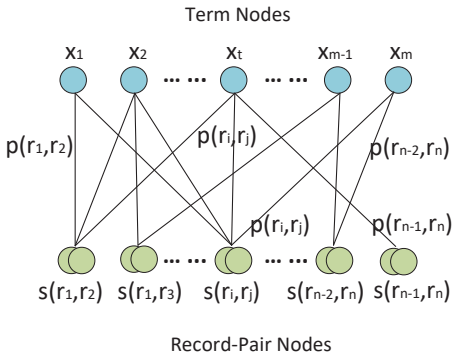


Figure 3. An example of bipartite graph for terms and record-pairs.

C. Iterative Term-Entity Ranking Algorithm

The objective of ITER algorithm is to estimate the weight of nodes, *i.e.*, x_t and $s(r_i, r_j)$ from the bipartite graph model tailored for entity resolution. As the sharing of discriminative terms is a strong indicator of the same entity, it is reasonable to infer the weight propagation from term nodes to pair nodes by summing up the node weight of term nodes. This is equivalent to our recursive definition in Equation 6. This equation captures our intuition that the more discriminative terms two records share, the more likely that they refer to the same entity.

Likewise, the discrimination power of a term is reflected in its connected pair nodes — if all connected pairs refer to the same entity, then the term is a discriminative feature (an example is product model); on the contrary, if the connected pairs represent a diverse set of entities, then the term is less discriminative (an example is stop words). Mathematically, the propagation from pair nodes to term nodes is equivalent to our definition in Equation 6.

Without any external knowledge, $p(r_i, r_j)$ can be set uniformly for all pair nodes. In our solution, $p(r_i, r_j)$ has another functionality — serving as a bridge to connect our ITER and CliqueRank algorithm, which is detailed later in Section VI. Initially, ITER runs with a uniform $p(r_i, r_j)$ until convergence; then CliqueRank refines $p(r_i, r_j)$ and ITER runs with the new $p(r_i, r_j)$ as the edge weight in the bipartite graph. Equqations 6 and 7 naturally form

an iterative process for estimating the weight of nodes. Since Equation (7) is an additive rule that increases $s(r_i, r_j)$ without bounds, it is instructive to normalize the term weights x_t in each iteration for the sake of convergence. In our implementation, we set $x_t = 1/(1+\frac{1}{x_t})$ to normalize the weight between 0 and 1. Other normalization methods such as L_2 norm: $\sum_t x_t^2 = 1$ can also be applied. We summarize the iterative method in Algorithm 1.

Algorithm 1: ITER Algorithm

Input: Bipartite graph structure with edge weight $p(r_i, r_j)$;
Output: Node salience x_t and record pair similarity score $s(r_i, r_j)$;

1. Randomly initialize x_t in $(0, 1)$;
2. **while** x_t does not converge **do**
3. **for** each record pair (r_i, r_j) **do**
4. Set its weight $s(r_i, r_j) \leftarrow \sum_{t \in r_i \wedge t \in r_j} x_t$;
5. **for** each term t **do**
6. Set its weight $x_t \leftarrow \sum_{i \neq j} \frac{p(r_i, r_j) s(r_i, r_j)}{P_t}$;
7. Set $x_t = 1/(1 + \frac{1}{x_t})$
8. **return** x_t and $s(r_i, r_j)$

The key difference of our algorithm with the PageRank based approaches is in Equation (6), the denominator that punishes the score of x_t by its number of connected nodes. We point out that this key difference is crucial to estimate the discriminative power of a term. More specifically, if t is a common term and occurs in many entries (such as some domain-specific stop words), its weight will be diluted by the non-matching pairs. Since PageRank lacks the denominator for punishing frequent terms, it will assign a high score to common terms, thus failing in estimating the salience of a term for entity resolution. The other difference is that we introduce $p(r_i, r_j)$ as the edge weight to promote terms only occurring in matching pairs.

Complexity Analysis. The ITER algorithm is efficient as its running time grows linearly with the number of edges in the bipartite graph. In each iteration, all the edges are traversed twice to update x_t and $s(r_i, r_j)$ respectively. From our experimental results, we observed that the algorithm could quickly reach convergence with only a few iterations.

D. Proof of Convergence

Let matrix **S** denote the adjacency matrix of the bipartite graph, where each entry $S_{ab} = 1$ if and only if there is an edge between term t_a and pair p_b; otherwise, S_{ab} is 0. Here, we represent b as a unique index for each pair (r_i, r_j) and set $b = i*n+j$. Let vector **x** and **y** denote the weight vector for term nodes and pair nodes, respectively. Then, we can rewrite the iterative update rules in the matrix form:

$$\mathbf{y} = \mathbf{S}^T\mathbf{x}; \qquad \mathbf{x} = \mathbf{D}^{-1}\mathbf{S}\mathbf{C}\mathbf{y}, \tag{8}$$

where **D** is a diagonal matrix with each entry $D_{aa} = P_{t_a}$ $(1 \leq a \leq m)$, and **C** is also a diagonal matrix with each entry $C_{bb} = p(r_i, r_j)$ for $b = i * N + j$. In the next, we show the convergence of the **y** vector, as **x** can be directly derived from **y**.

Theorem 1: The **y** vector of iterations (defined in Equation (8)) converges to stationary solution $\mathbf{y}^*$.

Proof: First, we eliminate **x** in **y**'s update rule:

$$\mathbf{y}^{(k)} = (\mathbf{S}^T\mathbf{D}^{-1}\mathbf{S}\mathbf{C})\mathbf{y}^{(k-1)} = ... = (\mathbf{S}^T\mathbf{D}^{-1}\mathbf{S}\mathbf{C})^k\mathbf{y}^{(0)}, \tag{9}$$

where k denotes the number of iterations, and $\mathbf{y}^{(0)}$ denotes the initialization of **y**. By standard linear algebra [31], if **M** is a diagonalizable matrix, and **v** is a vector not orthogonal to the principle eigenvector of **M**, then the direction of $\mathbf{M}^k v$ converges to the principle eigenvector as k increases without bound. Since $\mathbf{S}^T\mathbf{D}^{-1}\mathbf{S}$ is a symmetric matrix and **C** is diagonal, $\mathbf{S}^T\mathbf{D}^{-1}\mathbf{S}\mathbf{C}$ is guaranteed to be a diagonalizable matrix. Thus, the iterative algorithm finally converges to the stationary solution:

$$\mathbf{y}^* = \mathbf{w}_1; \qquad \mathbf{x}^* = \mathbf{D}^{-1}\mathbf{S}\mathbf{C}\mathbf{w}_1, \tag{10}$$

where $\mathbf{w}_1$ denotes the principle eigenvector (after unitization) of matrix $\mathbf{S}^T\mathbf{D}^{-1}\mathbf{S}\mathbf{C}$. The proof is finished. ■

VI. Inferring Matching Probability

Given the similarity function of records, a typical solution is to set a threshold and treat the record pairs with similarity score larger than the threshold as the same entity. However, tuning such similarity threshold is challenging as it is subject to the concrete applications. To address this issue, we propose a new graph-theoretic method that estimates the matching probability (*i.e.*, $p(r_i, r_j)$ in ITER) based on the similarity function. Owing to the probability nature of $p(r_i, r_j)$, the matching threshold will be much easier to tune for practitioners — usually the value is close to 1 and the setting can be universal across domains. Another desired property is that the matching probability can be fed to the ITER algorithm, serving as the edge weight in the Bipartite graph to enhance the effectiveness of learned similarity.

In the section, we first present a graph-based problem formulation for entity resolution. By interpreting the classic problem from the perspective of random walk, we propose a heuristic random surfer-based sampling (RSS) method to estimate matching probability by utilizing the topological structure of record graph. However, RSS suffers from severe efficiency problem for dense graphs. We thus propose CliqueRank to resolve the efficiency issue and render our unsupervised framework applicable in practice.

A. Problem Formulation

Let G_r be a record graph, where each node represents a record and the edge weight denotes the similarity of two connected records[2]. Let G_r^{opt} be the "ground-truth" record

[2]Note that according to our ITER algorithm, two records are connected only if they share at least one term.

graph, in which two records are connected if and only if they refer to the same entity. It is clear that for G_r^{opt}, the records that refer to the same entity form a clique, which is disconnected with the remaining parts of G_r^{opt}. Our objective is to leverage the topological structure of G_r and transform it to G_r^{opt}.

B. Random Walk Based Interpretation

To tackle the graph reduction problem, our primary idea is to leverage the *clique* structure of G_r^{opt}, similar to the transitivity property used in the crowd-sourcing methods to resolve conflicting pairs. However, instead of judging the pair similarity in an interactive fashion with crowd workers, our algorithm automatically learns the matching probability without human intervention. Ideally, if records r_i and r_j refer to different entities, they are supposed to be located in different cliques, which are unreachable from each other. On the contrary, if r_i and r_j refer to the same entity, they should be in the same clique and reachable from each other; that is to say, *if we start a random walk from one record r_i, it will be very likely to visit the other record r_j within a fixed number of steps via a rectified random walk strategy.* We leverage this key insight to develop a random surfer-based sampling (RSS) method to identify matching records.

The algorithmic framework of RSS, sketched in Algorithm 2, is simple and straightforward. After constructing the record graph with the similarity function, we perform the sampling process for each record pair (r_i, r_j). In particular, we simulate M times of random walk, half of them starting from r_i and the other half starting from r_j, and estimate the probability of r_i reaching r_j or vice versa (Lines 3–7). Each random walk outputs a number 1 or 0, indicating whether the surfer has reached the target node within a given number of S steps. The matching probability $p(r_i, r_j)$ is then estimated as the percentage of walks that have successfully reached the target (line 8). If (r_i, r_j) is a matching pair, we expect the reaching probability to be close to 1.

We consider both directions of random walk for an edge (r_i, r_j) because the reaching probability from r_i and r_j could be greatly different. An extreme example is the corner case where a node only has one neighbor, such that, the surfer from the node will always reach the target since there are no other choices. The bi-directional random walk can depress such negative cases.

In the next, we elaborate how we design the random walk process, a key component of our RSS method.

Rectified Random Walk. Recall that our objective is to interpret matching probability $p(r_i, r_j)$ as the reaching probability from r_i to r_j within a moderate number of S steps. If two records refer to the same entity, the reaching probability is close to 1. Otherwise, the probability would be close to 0. Conventional random walk cannot satisfy the requirements because it uses linear transition probability among the outgoing edges. If S is set large, running time increases dramatically and there would be many non-matching pairs with reaching probability close to 1. If S is set small, we may obtain very low reaching probability of matching pairs, which are considered as false negatives. Consequently, S would become a very sensitive parameter to tune.

Algorithm 2: Random-Surfer Sampling (RSS)

1. Construct a record graph G_r based on $s(r_i, r_j)$;
2. **for** each edge $(r_i, r_j) \in G_r$ **do**
3. $c_1 \leftarrow 0$; $c_2 \leftarrow 0$
4. **for** $m \leftarrow 1; m \leq M/2; m++$ **do**
5. $c_1 \leftarrow c_1 + RandomWalk(r_i, r_j)$;
6. **for** $m \leftarrow 1; m \leq M/2; m++$ **do**
7. $c_2 \leftarrow c_2 + RandomWalk(r_j, r_i)$;
8. $p(r_i, r_j) \leftarrow (c_1 + c_2)/M$;
9. **return** $p(r_i, r_j)$;

Algorithm 3: RandomWalk($start$, $target$)

1. $cur \leftarrow start$;
2. **for** $s \leftarrow 1; s \leq S; s++$ **do**
3. pick a random value $b \in (0, 1)$;
4. $s'(cur, target) \leftarrow (1 + b) \cdot s(cur, target)$;
5. $next \leftarrow$ pick a node from neighbors of cur based on the new transition probability $p_b(r_i \rightarrow r_j)$;
6. **if** $next == target$ **then**
7. **return** 1;
8. **if** edge $(next, target) \notin G_r$ **then**
9. **return** 0;
10. $cur \leftarrow next$;
11. **return** 0;

To resolve the issue, we propose to champion the edges of high similarity scores. Instead of using a linear transformation of edge weights, we thus devise the transition probability as non-linear transformation of edge weights:

$$p(r_i \rightarrow r_j) = \frac{s(r_i, r_j)^\alpha}{\sum_{r_j \in O(r_i)} s(r_i, r_j)^\alpha}, \tag{11}$$

where $p(r_i \rightarrow r_j)$ denotes the probability of choosing r_j with the current step at r_i and $O(r_i)$ denotes the neighbors of r_i in G_r. α is a tunable parameter that controls the degree of promoting edges of high weights – the larger α is, the more likely to choose the high-weight edges. In practice, α is often set large enough to increase the probability gap between picking matching records as the next node and picking non-matching neighbors. In this way, the whole random walk procedure is expected to be restricted in the "groud-truth" clique until the target node is reached.

With the rectified transition probabilities, the reaching probability between two non-matching records is still very small, even after many steps of random walk. On the other hand, the similarity scores between matching pairs are generally close to each other and not affected by the normalized transition probability. After a moderate number of steps, a record is highly likely to reach its matching pair. In this way, we can use the reaching probability to estimate the matching probability.

In our implementation, besides the non-linear transition probability, we also make two other refinements. First, before choosing the neighbor to jump, we add a bonus weight to the edge between the current node and target node (lines 3-5, where b is a random number in the range (0,1)). More specifically, let r_i be the current node of random walk and r_t be the target node. The normalization factor becomes

$$norm(r_i, r_t) = (1+b)^\alpha s(r_i, r_t)^\alpha + \sum_{r_k \in O(r_i) \wedge r_k \neq r_t} s(r_i, r_k)^\alpha \tag{12}$$

We define the weight boosted transition probability $p_b(r_i \rightarrow r_j)$ with respect to a particular random walk process from r_i to r_t as

$$p_b(r_i \rightarrow r_j) = \begin{cases} \frac{(1+b)^\alpha s(r_i, r_j)^\alpha}{norm(r_i, r_t)} & \text{if } r_j = r_t \\ \frac{s(r_i, r_j)^\alpha}{norm(r_i, r_t)} & \text{if } r_j \neq r_t \end{cases}$$

Note that b is randomly generated for each $p_b(r_i \rightarrow r_j)$. This step is important to ensure the RSS algorithm can identify the entities that are associated with many records. For example, in the Paper benchmark dataset, the largest entity has 192 records, which correspond to a very large clique with 192 nodes in the record graph; if the surfer starts from a node of the clique, it may have to jump many steps to reach the target node, even though the target is in the same clique (as the non-linear transition probability for each outgoing edge is very small, around 1/191 when the edge weights in the same clique are close to each other). This results in a dilemma: if we want to detect the clique by random walk, S has to be a very large number, which will significantly slow down the simulation process. By boosting the weight of edge between the current node and target node in each step, we can resolve the big-clique issue and speed up the simulation without degrading the resolution accuracy. Note that the reaching probability between non-matching pairs may also be boosted. Since the bonus factor is selected randomly between $(0, 1)$ for each step of random walk and these pairs normally have small weight, the success rate of reaching a non-matching neighbor within S steps is still quite small.

Second, we additionally apply an early stop strategy: if the surfer has reached a node that is not a neighbor of the target, we consider it a failure and return 0 (line 8–9). This is because the surfer has traversed outside of the clique that the target node locates in, meaning that the start node and target node are less likely to refer to the same entity.

Complexity Analysis. Let n be the number of records and the number of edges in G_r is $O(n^2)$ in the worst case. Since we conduct the sampling process for all the edges for M times and the random walk needs to select the next destination from all the neighbors for at most S steps, the complexity of RSS is $O(M \cdot S \cdot n^3)$. Obviously, the cubic complexity renders RSS not applicable in practice, especially in large-scale dense G_r. In the following, we will present a more efficient matrix-based algorithm named CliqueRank to replace RSS.

C. CliqueRank Algorithm

The RSS algorithm is essentially designed to estimate the probability of r_i reaching r_j within S steps. With sufficient number of steps, the probability is expected to be close to 1 if r_i and r_j refer to the same entity. To revisit RSS from the perspective of matrix operations, we first introduce an $n \times n$ probability transition matrix M_t, whose entry is the probability normalized non-linearly in Equation 11:

$$M_t[i, j] = p(r_i \rightarrow r_j) \tag{13}$$

If r_i and r_j are disconnected in G_r, we set $M_t[i, j] = 0$. We also set $M_t[i, i] = 0$ such that the random walk always chooses a neighboring node to proceed.

Let $M_t^S[i, j]$ be an $n \times n$ matrix whose entry is the probability matrix of r_i reaching r_j with exactly S steps. With the transition matrix M_t, we have

$$M_t^S = \underbrace{M_t \times M_t \times \ldots \times M_t}_{S-1 \text{ times of operation } \times} \tag{14}$$

When $S = 1$, the transition matrix M_t has captured the probability of r_i reaching r_j with one step and $M_t^S = M_t$. When $S > 1$, we leverage the preserved probability matrix M_t^{S-1} to calculate M_t^S. In particular, we split the random walk from r_i to r_j with S steps into two components. First, we pick a neighbor of r_i in G_r, say r_k, with probability $p(r_i \rightarrow r_k)$ preserved in M_t. For each possible r_k, its probability of reaching r_j with $S - 1$ steps has been calculated in M_t^{S-1}. By multiplying these two matrices, we yield M_t^S which sums up all the probabilities of paths with S steps.

It is noticeable that we customize the standard random walk with two improvements in our RSS algorithm. One is the weight boosting strategy to resolve the issue of low reaching probability when walking within a very large clique. The other customization is to return 0 if the random walk visits a node not directly connected to the target. To address these two refinements, we define a new weight boosted matrix M_b whose entry $M_b[i, j]$ is set to $p_b(r_i \rightarrow r_j) = \frac{(1+b)^\alpha s(r_i, r_j)^\alpha}{norm(r_i, r_j)}$. Let M_n be the adjacency matrix in which $M_n[i, j] = 1$ if r_i and r_j are connected in G_r. Otherwise, $M_n[i, j] = 0$. With the matrices M_b, M_t and M_n, we revise the definition of M_t^S to

$$M_t^S = \begin{cases} M_b & \text{if } S = 1 \\ M_t \times (M_t^{S-1} \odot M_n) & \text{if } S > 1 \end{cases}$$

The $\odot$ operation refers to pair-wise multiplication between two matrices and $\times$ is the normal matrix multiplication. When $S = 1$, the matrix is initialized to M_b. In the subsequent steps, $M_t^{S-1} \odot M_n$ sets the reaching probability of non-adjacent pairs to 0 to prevent the random

walk deviating from the groundtruth clique. The effect is equivalent to lines $8-9$ in RandomWalk algorithm. Then, matrix multiplication sums up the reaching probability with exactly S steps by reusing the results in M_t^{S-1}. Finally, the matching probability $p(r_i, r_j)$ can be derived from M_t^S. It is approximated by the probability of r_i reaching r_j within S steps. To take into account random walk from both directions, we take an average of the probabilities of $M_t^k[i,j]$ and $M_t^k[j,i]$ for all $1 \leq k \leq S$:

$$p(r_i, r_j) = \sum_{k=1}^{S} \frac{M_t^k[i,j] + M_t^k[j,i]}{2} \tag{15}$$

Complexity Analysis. Since the calculation of M_t^S involves $S-1$ times of matrix multiplication between two $n \times n$ matrices, the complexity is cubic, i.e. $O(S{\cdot}n^3)$. Compared to RSS, CliqueRank has two advantages. First, it can reuse the intermediate results of matrix multiplication from steps $S-1$ to S. Second, it can leverage third-party library optimized for fast and parallel matrix operation. As we will show in the experiments, compared to RSS, CliqueRank can boost the efficiency by up to 60x, making the running time of our unsupervised framework acceptable in practice.

VII. Experiments

We conduct comprehensive experiments to evaluate the accuracy and efficiency of our proposed graph-theoretic approaches by comparing with 14 competitors on three benchmark datasets. In addition, we provide break-down analysis to examine the effectiveness of ITER algorithm and the reinforcement effect between ITER and RSS.

A. Benchmark Datasets

We use three benchmark datasets that have been widely used by existing solutions to entity resolution to evaluate the effectiveness of our unsupervised resolution framework.

Restaurant[3] is a single-source dataset with 858 non-identical restaurant records. Each record contains the information of restaurant name and address. We can generate $\frac{858*857}{2} = 367,653$ candidate pairs from the dataset, among which 106 pairs refer to the same entity.

Product[4] is a dataset from two sources, one with 1081 records coming from the *abt* website and the other with 1092 records from the *buy* website. Among the $1081 \times 1092 = 1,180,452$ candidate pairs, $1,092$ of them referring to the same entity. Each product record contains its name and descriptive information.

Paper[5] is a single-source dataset with 1865 non-identical publication records. Each record has a unique cluster id and its textual information consists of authors, title, publication venue and year. Records within the same cluster refer to the same entity. There are 96 clusters with at least 3 records and the largest one contains 192 records. Hence, the dataset generates much more matching pairs.

For data pre-processing, we first tokenize the textual contents and then remove the terms that are very frequent. This step is a common practice widely adopted in natural language processing. These terms may dilute the effect of discriminative terms and eliminating them can help improve the accuracy to certain extent. After the tokenization step, each record consists of a group of raw tokens with abbreviations and misspellings, from which our goal is to identify pairs of records referring to the same entity.

B. Comparison Methods

For performance evaluation, we compare our proposed ITER+CliqueRank with 14 competitors, including string-distance based, machine-learning based, crowd-sourcing based and the proposed graph-theoretic baselines. Detailed algorithms are listed in Table II. All the experiments were carried out on a server with Xeon CPU 2.6GHz (32 cores) and 32GB memory, running Centos 6.5. We implemented string-based and graph-theoretic approaches with C++. The results of machine-learning based and crowd-sourcing based methods are obtained from their published papers. In the implementation of CliqueRank, Eigen library[6] was used to boost matrix multiplication.

C. Parameter Setting

For string-based similarity methods (Jaccard and TF-IDF), an appropriate threshold is required to determine matching pairs. Our strategy is to quantize the domain $[0, \max\{s(r_i, r_j)\}]$ into 1000 discrete values and automatically select the threshold with the highest F1-measure by computer programming, which is an upper bound of manually tuned parameters.

In the graph-theorectic baseline methods, we set C_1 and C_2 to be 0.8 for SimRank on bipartite graph, as suggested in [23]. For PageRank on term graph, damping factor ϕ is set to 0.85. β in the hybrid combination of SimRank and PageRank is set to 0.5. Similar to the string-based methods, we select an optimal similarity threshold by examining all the possibilities.

For our proposed methods, the ITER algorithm does not involve any parameter that requires tuning. The CliqueRank algorithm has three parameters: α for the non-linear transformation in the RSS algorithm, S as the maximum number of steps for the random walk and finally the probability threshold η. In the following, we elaborate the criterion for parameter setting. α is normally set large enough to generate a dominating gap for the transition probability to matching and non-matching neighbors. S also needs to be set to a large value in order to identify big cliques in

[3] http://www.cs.utexas.edu/users/ml/riddle/data/restaurant.tar.gz
[4] http://dbs.uni-leipzig.de/Abt-Buy.zip
[5] http://www.cs.umass.edu/~mccallum/data/cora-refs.tar.gz
[6] http://eigen.tuxfamily.org/index.php?title=Main_Page

the Paper dataset. η is normally set to a value close to 1 as it refers to the matching probability threshold and we require a high confidence. In our implementation, we universally set $\alpha = 20$, $S = 20$ and $\eta = 0.98$ to achieve promising accuracies and run the iterations of ITER followed by CliqueRank for 5 times (the reinforcement effect will be examined in Section VII-F). This is an attractive feature because we can actually use the same parameter settings for all the three datasets from different domains. It greatly alleviates the pain of parameter tuning and exhibits the generality of our proposed framework.

D. Performance of Entity Resolution

Table II
RESULTS OF F1-SCORES IN THREE DATASETS.

	Restaurant	Product	Paper
String-distance based approaches			
Jaccard	0.836	0.332	0.792
TF-IDF	0.871	0.658	0.821
Machine-learning based approaches			
Gaussian Mixture Model [5]	0.704	-	-
HGM+Bootstrap [5]	0.844	-	-
MLE [5]	0.904	-	-
SVM [6]	0.922	-	0.824
Crowd-sourcing based approaches			
CrowdER [8]	0.934	0.800	0.824
TransM [10]	0.930	0.792	0.740
GCER [9]	0.930	0.760	0.785
ACD [12]	0.934	0.805	0.820
Power+ [13]	0.934	-	0.820
Graph-theoretic baselines			
SimRank	0.645	0.376	0.730
PageRank	0.905	0.564	0.316
Hybrid	0.946	0.593	0.748
Proposed graph-theoretic fusion framework			
ITER+CliqueRank	0.927	0.764	**0.890**

The F1-scores of the 14 competitors in benchmark datasets are presented in Table II. It is notable that the results of machine-learning based and crowd-sourcing based approaches are directly collected from the related publications. We did not implement these algorithms and it is possible to generate missing cells in the table if the results were not reported in the original papers. The string-similarity based methods, including Jaccard and TF-IDF, are efficient and easy to implement. However, their resolution accuracies are not high enough to be applied in practice. The TF-IDF achieves much higher F1-score in Product dataset because the idf assigns higher weight to more distinguishing product feature terms. The supervised machine learning approaches noticeably improve the accuracy, with limited effect. In addition, they require considerable amounts of labeled data for supervised training.

The crowd-assisted solutions proposed in recent years as state-of-the-art, including CrowdER [8], TransM [10], GCER [9], ACD [12] and Power+ [13], have been shown to achieve superior accuracy to the previous unsupervised or supervised approaches. These methods iteratively select the candidate pairs that are worth being sent to the crowd for verification until certain level of accuracy is achieved. However, these methods have to spend a remarkable amount of cash as well as waiting time on crowd-workers, which are proportional to the number of questions to be asked.

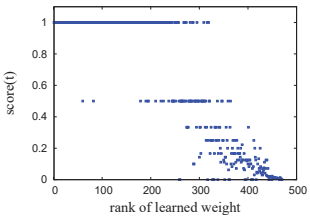
The results of graph-theoretic baselines are not promising as SimRank only leverages structural connection between records and terms; while PageRank considers term-based similarity and ignores the "clique" property. The hybrid baseline achieves superior performance because it takes both structural similarity and content-based similarity into account.

Our unsupervised approach has three advantages compared to crowd-sourcing methods. First, our method does not require human intervention and helps save financial budget, especially in large-scale datasets. Second, the crowd-sourcing methods are slow because they need to wait for the response from human workers; whereas, our method relies only on machine computing resources, which is always available and more efficient. Third, our unsupervised approach can achieve comparable or even superior accuracy over crowd-sourcing methods. As shown in Table II, the F1-score in the Paper dataset is boosted from 0.824 to 0.890. The results verify the effectiveness of our proposed graph-theoretic framework to automatically identify distinctive terms and determine matching record-pairs.

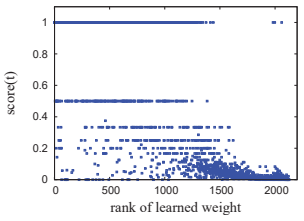
Running time. We report the time cost of ITER+CliqueRank for 5 iterations in Table III and have three observations. Firstly, the complexity of ITER algorithm is linear to edge number. It runs fast as our dataset only contains at most thousands of records. Its running time can be omitted when comparing to RSS and CliqueRank, which is the bottleneck of our unsupervised framework. Secondly, the time cost of CliqueRank in each dataset is less than half an hour, showing that our unsupervised method is efficient in practice and has the potential to resolve datasets with larger scale. It's also worth noting that the its running time is sensitive to the number of nodes in G_r rather than the number of edges. Thirdly, CliqueRank can significantly boost the performance in dense G_r. For example, it achieves 60 times speedup in the Paper dataset. It uses iterative matrix multiplication to calculate the reachability probability within S steps. The intermediate results of reaching probability for $S-1$ steps, i.e., M_t^{S-1}, can be reused to calculate M_t^S. In contrast, RSS conducts random-walk based sampling for each pair independently. In addition, the underlying Eigen library is optimized for matrix operation with multi-threads. Since our server is multi-core with 32 cores, it can better exploit the hardware resources for parallel computing.

E. Effectiveness of ITER Algorithm

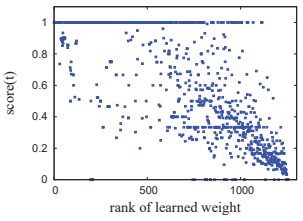
We also study the efficacy of the learned term weight of the ITER method. Since our primary objective is to detect the most discriminative terms in an application domain and assign them with high weights, we use the



(a) Restaurant



(b) Product



(c) Paper

Figure 4. Visualization of the effectiveness of the learned term weights.

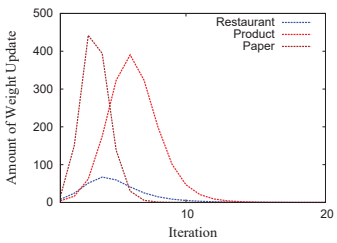


Figure 5. Convergence of ITER.

Table III
EFFICIENCY OF ITER+CLIQUERANK.

	Restaurant	Product	Paper
Number of nodes in G_r	858	2173	1865
Number of edges in G_r	5,320	151,939	980,780
Total running time	1.1min	21.6min	24.2min
Running time for ITER	3sec	20sec	58sec
Speedup compared to RSS	1.3x	1.5x	60x

Table IV
SPEARMAN'S RANK CORRELATION COEFFICIENT.

	Restaurant	Product	Paper
PageRank	0.30	0.02	0.08
ITER	0.96	0.76	0.80

following criterion to measure the effectiveness of a term t: $score(t) = \frac{\sum_{t\in r_i \wedge t \in r_j} I(r_i,r_j)}{P_t}$,where P_t is the number of record-pairs connected with term t in the bipartite graph and $I(r_i, r_j) = 1$ if its associated pair (r_i, r_j) refer to the same entity. Otherwise, we set $I(r_i, r_j) = 0$. Intuitively, if all the records in the inverted list of term t refer to the same entity, then t is said to be highly discriminative and assigned with score 1. In contrast, if t is a common term and occurs in many different entities, then t should be assigned with a lower score.

In the optimal scenario, the ranking order of terms sorted by the learned weight x_t should follow the same order as $score(t)$. To examine this, we plot in Figure 4 the terms in decreasing order of their weight in the x-axis. In other words, the term t with the i-th highest x_r is plotted at the position of $x = i$. The y-axis value for each term t is set to $score(t)$. We can see that the highly discriminative terms with $score(t) = 1$ are clustered in the front part of x-axis. It means our ITER algorithm also considers them as much more important terms and assigns them with higher weight. The common terms that are not discriminative (with small $score(t)$) are clustered in the bottom-right corner, which is also in accordance with our unsupervised entity-term ranking strategy.

Statistically, we use Spearman's rank correlation coefficient to access the quality of ITER's ranked list. It is defined as $r_s = 1-6d_a^2/(n(n^2-1))$, where d_a is the rank difference of term t when sorted by $score(t)$, and n is the number of terms. The results in Table IV show that PageRank cannot capture the discriminative terms and its coefficient value is very small. In contrast, our ITER algorithm demonstrates very high rank correlation between x_t and $score(t)$.

To empirically show that the node weight will converge, we plot the sum of weight value update of x_t and $p(r_i, r_j)$ in Figure 5 for the first 20 iterations. We can see that in the initial stage the weights change dramatically, exhibiting a sharp-peek pattern for all the three datasets. This is because the parameters are initialized randomly. With a few more iterations, they can quickly reach convergence.

F. Effect of Reinforcement

Table V shows the effect of reinforcement with multiple iterations between ITER and CliqueRank. In the first iteration, the weight $p(r_i, r_j)$ in ITER algorithm is initialized to 1 for all pairs. Then, we apply CliqueRank on the derived record similarity graph G_r and obtain F1-scores listed in the first row. Thereafter, we feed the matching probability $p(r_i, r_j)$ to the bipartite graph and run ITER in the second iteration. We can see that there is a noticeable improvement of F1-scores for CliqueRank in the three datasets. With more iterations, the accuracy exhibits an increasing trend. Note that there may be slight fluctuation as in Restaurant. The phenomenon is similar to the decreasing trend of loss values in supervised training.

Table V
EFFECT OF REINFORCEMENT.

	Restaurant		Product		Paper	
Iteration	F1-score	Time	F1-score	Time	F1-score	Time
1	0.916	13	0.543	253	0.844	207
2	0.935	25	0.712	514	0.888	515
3	0.931	39	0.747	768	0.889	819
4	0.931	52	0.754	1027	0.890	1135
5	0.927	64	0.764	1296	0.890	1453

VIII. CONCLUSION

In this paper, we proposed a graph-theoretic fusion framework for unsupervised entity resolution. In particular, we devised two novel algorithms ITER and CliqueRank, one for term-based similarity and the other for topological confidence, that can reinforce each other. Experimental results on three benchmark datasets verified the effectiveness of our framework. Without any labeled data or human

assistance, its accuracy can be comparable or even superior to the state-of-the-art crowd-assisted approaches.

IX. Acknowledgement

This work is supported in part by the National Natural Science Foundation of China under grants No. 61602087, 61672133, 61702016, 61632007, 61661146001, the Fundamental Research Funds for the Central Universities under grants No. ZYGX2016J080, ZYGX2014Z007, the China Postdoctoral Science Foundation under Grant No. 2017M610019 and the 111 Project No. B17008.

References

[1] A. E. Monge and C. Elkan, "The field matching problem: Algorithms and applications," in *KDD*, 1996, pp. 267–270.

[2] W. W. Cohen, "Data integration using similarity joins and a word-based information representation language," *ACM Trans. Inf. Syst.*, vol. 18, no. 3, pp. 288–321, 2000.

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

[4] S. Sarawagi and A. Bhamidipaty, "Interactive deduplication using active learning," in *KDD*, 2002, pp. 269–278.

[5] P. D. Ravikumar and W. W. Cohen, "A hierarchical graphical model for record linkage," in *UAI*, 2004, pp. 454–461.

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

[7] A. Arasu, M. Götz, and R. Kaushik, "On active learning of record matching packages," in *SIGMOD*, 2010, pp. 783–794.

[8] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "Crowder: Crowdsourcing entity resolution," *PVLDB*, vol. 5, no. 11, pp. 1483–1494, 2012.

[9] S. E. Whang, P. Lofgren, and H. Garcia-Molina, "Question selection for crowd entity resolution," *PVLDB*, vol. 6, no. 6, pp. 349–360, 2013.

[10] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *SIGMOD*, 2013, pp. 229–240.

[11] N. Vesdapunt, K. Bellare, and N. N. Dalvi, "Crowdsourcing algorithms for entity resolution," *PVLDB*, vol. 7, no. 12, pp. 1071–1082, 2014.

[12] S. Wang, X. Xiao, and C. Lee, "Crowd-based deduplication: An adaptive approach," in *SIGMOD*, 2015, pp. 1263–1277.

[13] C. Chai, G. Li, J. Li, D. Deng, and J. Feng, "Cost-effective crowdsourced entity resolution: A partial-order approach," in *SIGMOD*, 2016, pp. 969–984.

[14] 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.

[15] W. W. Cohen, P. D. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *IJCAI*, 2003, pp. 73–78.

[16] E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 5, pp. 522–532, 1998.

[17] M. Bilenko and R. J. Mooney, "Learning to combine trained distance metrics for duplicate detection in databases," Tech. Rep., 2002.

[18] 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.

[19] W. W. Cohen and J. Richman, "Learning to match and cluster large high-dimensional data sets for data integration," in *KDD*, 2002, pp. 475–480.

[20] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking," in *WWW*, 2012, pp. 469–478.

[21] J. Fan, Z. Wei, D. Zhang, J. Yang, and X. Du, "Distribution-aware crowdsourced entity collection," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–1, 2017.

[22] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," ser. WWW7, 1998, pp. 107–117.

[23] G. Jeh and J. Widom, "Simrank: A measure of structural-context similarity," in *KDD*, ser. KDD '02, 2002, pp. 538–543.

[24] R. F. Mihalcea and D. R. Radev, *Graph-based Natural Language Processing and Information Retrieval*, 1st ed. New York, NY, USA: Cambridge University Press, 2011.

[25] R. Blanco and C. Lioma, "Graph-based term weighting for information retrieval," *Inf. Retr.*, vol. 15, no. 1, pp. 54–92, 2012.

[26] X. He, M. Gao, M. Kan, Y. Liu, and K. Sugiyama, "Predicting the popularity of web 2.0 items based on user comments," in *SIGIR*, 2014, pp. 233–242.

[27] G. Hu, J. Shao, Z. Ni, and D. Zhang, "A graph based method for constructing popular routes with check-ins," *World Wide Web*, Nov 2017.

[28] X. He, M. Gao, M. Kan, and D. Wang, "Birank: Towards ranking on bipartite graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 57–71, 2017.

[29] F. Rousseau and M. Vazirgiannis, "Graph-of-word and TW-IDF: new approach to ad hoc IR," in *CIKM*, 2013, pp. 59–68.

[30] D. Zhang, L. Nie, H. Luan, K. Tan, T. Chua, and H. T. Shen, "Compact indexing and judicious searching for billion-scale microblog retrieval," *ACM Trans. Inf. Syst.*, vol. 35, no. 3, pp. 27:1–27:24, 2017.

[31] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.