Unsupervised Entity Resolution on Multi-type Graphs

Linhong Zhu, Majid Ghasemi-Gol, Pedro Szekely, Aram Galstyan, Craig A. Knoblock

Information Sciences Institute, University of Southern California
{linhong, ghasemig, pszekely, galstyan, knoblock}@isi.edu

Abstract. Entity resolution is the task of identifying all mentions that represent the same real-world entity within a knowledge base or across multiple knowledge bases. We address the problem of performing entity resolution on RDF graphs containing multiple types of nodes, using the links between instances of different types to improve the accuracy. For example, in a graph of products and manufacturers the goal is to resolve all the products and all the manufacturers. We formulate this problem as a multi-type graph summarization problem, which involves clustering the nodes in each type that refer to the same entity into one super node and creating weighted links among super nodes that summarize the inter-cluster links in the original graph. Experiments show that the proposed approach outperforms several state-of-the-art generic entity resolution approaches, especially in data sets with missing values and one-to-many, many-to-many relations.

1 Introduction

The increasing number of entities created online raises the problem of integrating and relating entities from different sources. In this work, we focus on the entity resolution problem. It is a common challenge in various domains including digital libraries, E-commerce, natural language understanding, etc. For example, in digital libraries, a challenging problem is to automatically group references that refer to the same publication and disambiguate author names, venues, etc. In E-commerce, a difficult problem is to match products from one domain (e.g., Amazon) to another domain (e.g., eBay).

Consider the example in Figure 1, where we have five products from different sellers represented by RDF. The entity resolution task is to group vertices of the same product entity (e.g., 1 and 2) and vertices of the same manufacturer entity (e.g., Bose and Bose Electronic) together and relate product entities with manufacture entities.

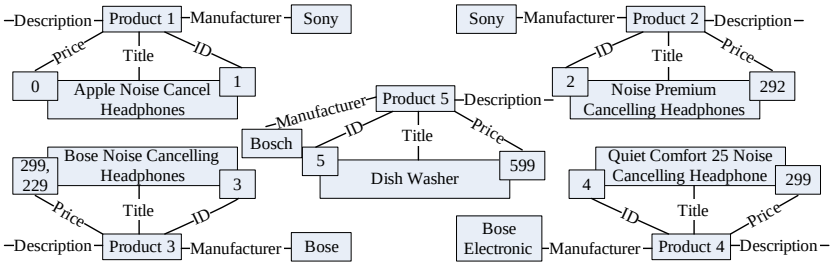
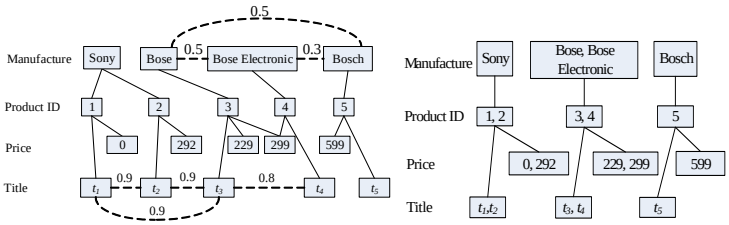


Fig. 1. An example of RDF graph for five products, where values of description field are omitted

There are several challenges in tacking entity resolution tasks. The first challenge is due to the poor quality of data, such as different spellings (cancel and cancelling), missing values (e.g., missing price for product 1) and ambiguity (e.g., the title of product 1 "Apple Noise Cancel Headphones" actually means that the headphones are suitable for apple products, but not manufactured by Apple). This makes traditional pair-wise distance measures approaches [9, 25, 30] less effective with noisy content and context (see related work). The second challenge is due to the one-to-many and many-to-many relation between entities. For instance, in the product entity resolution example, a product might be associated with many prices (normal or discount), and each manufacturer is associated with many products. The heterogeneous nature of relationships brings in an additional challenge when performing collective entity resolution [5, 10, 12]: to determine which kind of relationship is best suited for resolving a particular type of entity.



(a) An Input Graph (b) A Summary Graph

Fig. 2. An example of multi-type graph representation for Figure 1 and its corresponding summary graph. The description type of vertices is omitted

To address the aforementioned challenges, we model the observed RDF graph as a multi-type graph and formulate the collective entity resolution as a multi-type graph summarization problem. Particularly, the goal is to transform the original k-type graph into another k-type *summary graph* composed of *super nodes* and *super edges*. Each super node is a cluster of original vertices (of the same type) representing a latent entity, while super edges encode potentially valuable relations between those entities. As shown in Figure 2(a), we model the observed RDF graph as a multi-type graph, where vertices represent different types of objects, and edges represent either co-occurrence between two-types of vertices (solid edge), or similarity between the same-type vertices (dashed edge). The dashed similarity edge can be computed by any similarity measures such as graph proximity based similarity (e.g., number of common neighbors) or content-based similarity (e.g., string similarity). An example of a summary graph is shown in Figure 2(b), where each super node is a cluster of original vertices representing a hidden entity and each super edge relates one type entity (e.g., product 1, 2) to another type entity (e.g., Sony). The weights of the super edges also indicate which type of information is more useful when resolving certain types of entities. For instance, for product 1 and 2 disambiguation, manufacturer is more reliable than price, while for disambiguating product 3 and 4, price is a more reliable indicator than manufacturer.

In this work, we thus propose a unified, multi-type graph co-summarization based entity resolution framework (**CoSum**), which 1) jointly condenses a set of similar vertices in the observation into a super node in the summary graph so that each super node

(hidden entity) is coherent; 2) reveals how entities of different types are related with each other. Our main contributions are summarized as follows:

1. A novel formulation for the entity resolution problem, where we model the observed relations between different types of mentions as a multi-type graph and reduce the entity resolution to a graph summarization problem.
2. A multi-type graph co-summarization based generic entity resolution framework, which supports determining how many entities are discussed, entity disambiguation and entity relation discovery simultaneously.
3. A generic entity resolution framework that supports different user-supplied similarity measures as inputs. Those similarity measures can be of any general form and are not restricted to simple distance-based metrics.

We validate the proposed approach on real-world networks from both an E-commerce and a citation domain. The results show that the proposed approach outperforms other state-of-the-art approaches.

2 Related Work

Entity resolution has been extensively studied under different names such as record linkage [2, 7, 30], reference reconciliation [12], coreference resolution [23, 29]. In the following, we review a set of representative traditional entity resolution approaches and collective entity resolution approaches; while we refer to tutorials [13] and surveys [6, 8, 36] for more throughout reviews.

Distance-based entity resolution approaches focus on learning a pairwise distance metric between entities, and then either set a distance threshold or build a pairwise classifier to determine which entities are merged. The entire process can be unsupervised [9, 25, 30], or supervised [29], or a hybrid of these two [7, 15]. Limes [30] and Silk [15] are two representative entity resolution systems that focus on a pair of records at a time, and decide whether they are the same or not according to acceptance metrics and thresholds. Unfortunately, pairwise-based decision is very sensitive to noise and cannot capture the dependency between two pair-wise decisions.

To address the limitation of pairwise distance-based resolution decision, recently collective entity resolution has been extensively studied. This work can be categorized into three types. First, traditional collective entity resolution focuses on capturing the dependence among the same-type entities. For example, Pasula et al. [31] proposed the Relational Probabilistic Model for capturing the dependence among multiple coreference decisions. Conditional random fields (CRFs) [21] have been successfully applied to the entity resolution domain [26] and is one of the most popular approaches in generic entity resolution. On another hand, Singla and Domingos [34] proposed a well-founded, integrated solution to the entity-resolution problem based on Markov logic. Bhattacharya and Getoor [4] proposed a novel relational clustering algorithm that uses both attribute and relational information between the same-type entities for determining the underlying entities.

With heterogeneous data becoming more widespread, two additional types of collective entity resolution have emerged: 1) Collective resolution for entities with different types [5]. For instance, an extended LDA model was used in [5] to perform entity

resolution for authors and publications simultaneously; 2) Collective resolution for entities with the same type from different domains. For example, Dong et al. [12] models a pair of mentions or attributes from two different domains as a node and then applies a label propagation algorithm to perform collective entity resolution. Cudré-Mauroux et al. [10] adopt the factor-graph model to perform collective entity resolution for personal profiles. In this work, we propose a multi-type graph model for collective entity resolution, which supports the three aforementioned different types of collective entity resolutions in the same generic framework.

There is another direction of work that focused on methods to scale up entity resolution algorithms, such as using indexing [8] or blocking techniques [17, 18, 35] to facilitate pairwise similarity computation. A representative example is the Serf system [3], which developed strategies that minimize the number of invocations to these potentially expensive black-box entity resolution algorithms. Our framework is very generic, and any indexing/blocking technique can be seamlessly embedded into it.

Our work is related but less relevant to named-entity relation extraction, tagging [28, 32] and entity linking [16]. This work aims to extract named entities from a corpus and find the relation between entities deploying a fixed or universal schema, and implicitly do entity resolution along with extraction and tagging. However, our work focuses on resolving the same entities in a structured or semi-structured dataset, possibly extracted from different sources. In the outputted summary graph, our approach relates one type of entities to another type with weighted edges, but it does not support tagging the edge with a schema type such as "is-produced-by" between a product entity and a manufacturer entity.

3 Problem Definition

3.1 Notations

Let $G = (\cup_{t=1}^{k} V_t, \cup_{t=1}^{k} \cup_{t'=t+1}^{k} E_{tt'})$ be a k-type graph where each V_t denote a set of vertices of type t, and each $E_{tt'}$ denote the set of edges connecting two different types of vertices. Note that $E_{tt'}$ can be empty if none of the t-type vertices is connected to t'-type vertices. In addition, we also allow connections between vertices with the same type by introducing the similarity function $\texttt{sim}$. Each $\texttt{sim}_t(x, y) \geq 0$ evaluates the similarity between two t-type vertices x and y.

Given an input k-type graph G, a summary graph $\mathcal{S}(G) = (\cup_{t=1}^{k} S_t, \cup_{t=1}^{k} C_t, \cup_{t'>t}^{k} L_{tt'})$ is another k-type graph that consists of:

- k sets of super nodes $\{S_1, \cdots, S_k\}$, where each super node $s \in S_t$ (t =1 to k) denotes a cluster of t-type vertices in the original graph,
- $\binom{k}{2}$ sets of super links $L_{tt'}$ where each weighted edge $(s_t, s_{t'})$ denotes the expected probability that a t-type super node s_t is connected with a t'-type super node $s_{t'}$,
- k sets of "zoom-in" mappings $\{C_1, \cdots, C_k\}$ such that each C_t denotes probabilistic mapping between t-type vertices V_t and super nodes S_t.

Note that we use terms vertex and edge to refer to node and edge in original graph and terms super node and super link to refer to node and edge in summary graph. For simplicity, we use V to denote the set of vertices, E to denote the set of edges in original graph G, S to denote the set of super nodes and L to denote the set of super links in

Table 1. Notations and explanations.

Notations	Explanations.
n, m, p, q, k	number of vertices, edges, super nodes, super links, types
V,E,S,L	the set of vertices, edges, super nodes, super links
$E_{tt'}$	coreference links between t-type and t'-type vertices in the original graph
$L_{tt'}$	super links between t-type and t'-type super nodes
$C \in R^{n \times p}$	the mapping between vertices and super nodes
$\mathtt{sim}$	the similarity function between the same-type vertices
$C(x)$	the x^{th} row of a matrix C
$d(x), CN(x, y)$	the degree of vertex x, the common neighbors of vertex x and y
$J(x)$,$\mathcal{J}(x)$	Objective function, Lagrangian function of x
$\circ$	element-wise multiplicative operator

summary graph $\mathcal{S}(G)$. The total number of vertices and edges in G are denoted as n and m, and the total number of super nodes and super links in $\mathcal{S}(G)$ are denoted as p and q. We use symbols with subscript t to denote notations that are related to type t. A summary of all the notations and explanations are presented in Table 1.

3.2 Problem Formulation

As explained in Section 1, our goal is to reduce the entity resolution problem to a graph summarization problem, where the nodes representing different mentions of the same hidden entity are summarized, or condensed, into the same super node. There are numerous ways to summarize a graph depending on specific objectives. We now provide some intuition about what constitutes a good summarization in the context of the entity resolution task. In particular, we postulate that the super nodes in our summary graph need to be coherent, in the sense that the nodes comprising a given super node should be similar to each other. The rationale behind this assumption is that different mentions of the same hidden entity needs to share some similarities, otherwise the problem becomes infeasible. Furthermore, we differentiate between *inherent similarity*, as described by the content of those nodes themselves (e.g., string similarity between their labels), and *structural similarity*, which reflects similar connectivity patterns in the multi-type graph.

To accommodate for the first type of similarity, we define the following optimization problem:

$$\arg\min_{\mathcal{S}(G)} \sum_{t} \sum_{x,y \in V_t} \mathtt{sim}_t(x, y) \|C_t(x) - C_t(y)\|_F^2 \tag{1}$$

This objective function ensures that any summary graph in which two highly similar vertices (x, y) are not mapped to the same super node, incurs a penalty. The intuition behind this term is illustrated in the example in Figure 2. If the titles t_1 and t_2 are very similar, then it is very likely that t_1 and t_2 will be condensed into the same super node.

To accommodate for structural similarity, we note that if two t-type vertices are connected to the same t'-type vertex (or a set of t'-type vertices representing the same entity), it is likely that those two vertices are referring to the same entity as well. For instance, as shown in Figure 2, since both record 1 and record 2 are connected to the manufacturer "Sony" (and their connected titles/descriptions are very similar), it is likely that the records 1 and 2 are about the same product. Based on this intuition, we define

the following optimization criterion [1]:

$$\arg\min_{\mathcal{S}(G)} \sum_{t} \sum_{t'>t} \|G_{tt'} - C_t L_{tt'} {C_{t'}}^T\|_F^2 \tag{2}$$

Next, we combine Eq. (1) with Eq. (2) and formulate the following optimization problem:

Problem 1. Given an input k-type graph G and the similarity function $\mathtt{sim}_t$ for each vertex type t, find a summary graph $\mathcal{S}(G)$ for G that minimizes the following objective:

$$J(\mathcal{S}(G)) = \sum_{t} \sum_{x,y \in V_t} \mathtt{sim}_t(x,y) \|C_t(x) - C_t(y)\|_F^2 + \sum_{t} \sum_{t'>t} \|G_{tt'} - C_t L_{tt'} {C_{t'}}^T\|_F^2 \tag{3}$$

It is worthwhile to note that while both terms in Eq. 3 tend to produce more coherent super nodes, there are also certain important differences. Namely, the first term becomes irrelevant if two nodes are very dissimilar ($\mathtt{sim}_t(x,y) \approx 0$), whereas the second term will tend to assign structurally dissimilar nodes to different super nodes. Furthermore, the second term favors a larger number of super nodes, whereas the first term tends to condense similar nodes as much as possible. These differences introduce some non-trivial tradeoffs in the optimization process, which allow us to arrive at good summary graphs.

4 Methodology

4.1 Solution Overview

In this section, we introduce our solution to Problem 1. The overview of our solution is as follows (as well as outlined in Algorithm 1): Start with a random summary graph (Line 1), we first search for an improved summary graph with fewer super nodes, by crossing out one or many super nodes (Section 4.3). The second step is to fix the number of super nodes $[p_1, \cdots, p_k]$, and compute the vertex-to-clustering mapping C and super links L (Lines 4–10). These two procedures are performed alternately, until they reach a locally optimal summary graph (Lines 2–11).

4.2 Graph Summarization with given Super Nodes

We first study the summarization algorithms with a simplified condition, in which we assume that the number of super nodes in the summary graph ($[p_1, \cdots, p_t]$ is given. With this assumption, we show that the vertex to super nodes mapping C and the connections among super nodes L can be computed with a standard multiplicative update rule [22]. The intuition of the multiplicative rule is that whenever the solution is smaller than the local optimum, it multiplies with a larger value; otherwise, it multiplies with a smaller value.

Lemma 1. *With a non-negative initialization of $C_t \in R^{n_t \times p_t}$, C_t can be iteratively improved via the following update rule:*

$$C_t = C_t \circ \sqrt{\frac{\sum_{t'>t} G_{tt'} C_{t'} {L_{tt'}}^T + sim_t C_t}{\sum_{t'>t} C_t E_{tt'} {C_{t'}}^T C_{t'} {E_{tt'}}^T + D_t C_t}} \tag{4}$$

[1] Note that since the input graphs we focused are undirected, we save the half computation by assuming that types of vertices are ordered and restricting edges from a precedent type t to t'.

Algorithm 1 The graph summarization framework for k-partite graphs

Input: A k-type Graph G
Output: A k-type summary graph $\mathcal{S}(G)$
01: Initialize a random k-type summary graph, with number of super nodes $[n_1, \cdots, n_k]$
02: **repeat**
  /* vertex allocation optimization (Section 4.3)*/
03: $\mathcal{S}(G)$=Search(G, $\mathcal{S}(G)$) (see Alg. 2)
  /* fix the number of super nodes, and optimize super nodes assignment (Section 4.2)*/
04: **do**
05: **for** each t-type vertices
06: update C_t with Eq. (4)
07: **for** each non-empty edge set between t- and t'-type vertices
08: update $L_{tt'}$ with Eq. (5)
09: **while** C and L converge
10: construct the new summary graph $\mathcal{S}(G)$
11: **until** $J(\mathcal{S}(G))$ converges
12: **return** $\mathcal{S}(G)$

where D_t is the diagonal weighted degree matrix of the similarity matrix $\mathtt{sim}_t$, and $\circ$ (/) is the element-wise multiplicative (division) operator.

Proof (sketch): The update rule can be derived following the similar proof procedure proposed by Ding et al. [11] and Zhu et al. [37]. For each C_t, we introduce the Lagrangian multiplier Λ for non-negative constraint (i.e., $C_t \geq 0$) in Eq. (3), which leads to the following Lagrangian function $\mathcal{J}(C_t)$:

$$\mathcal{J}(C_t) = \sum_{t'>t} \|G_{tt'} - C_t L_{tt'} {C_{t'}}^T\|_F^2 + \sum_{x,y \in V_t} \mathtt{sim}_t(x,y) \|C_t(x) - C_t(y)\|_F^2) + \mathtt{tr}(\Lambda_{C_t} C_t^T)$$

The next step is to optimize the above terms w.r.t. C_t. We set the deviation of $\mathcal{J}(C_t)$ to zero ($\nabla_{C_t}\mathcal{J}(C_t)$=0), and obtain:

$$\Lambda_{C_t} = -2(\sum_{t'>t} G_{tt'} C_{t'} {L_{tt'}}^T + \mathtt{sim}_t C_t) + 2(\sum_{t'>t} C_t E_{tt'} {C_{t'}}^T C_{t'} {E_{tt'}}^T + D_t C_t)$$

Using the KKT condition $\Lambda_{C_t} \circ C_t$=0 [20], we obtain:

$$[-2(\sum_{t'>t} G_{tt'} C_{t'} {L_{tt'}}^T + \mathtt{sim}_t C_t) + 2(\sum_{t'>t} C_t E_{tt'} {C_{t'}}^T C_{t'} {E_{tt'}}^T + D_t C_t)] \circ C_t = 0$$

Since C_t is non-negative, we show that when the solution converges, the above equation is identical to the fixed point condition of following term:

$$[-2(\sum_{t'>t} G_{tt'} C_{t'} {L_{tt'}}^T + \mathtt{sim}_t C_t) + 2(\sum_{t'>t} C_t E_{tt'} {C_{t'}}^T C_{t'} {E_{tt'}}^T + D_t C_t)] \circ C_t^2 = 0$$

That is, either an entry of C_t or the corresponding entry of the left factor is zero. We thus have:

$$C_t = C_t \circ \sqrt{\frac{\sum_{t'>t} G_{tt'} C_{t'} {L_{tt'}}^T + \mathtt{sim}_t C_t}{\sum_{t'>t} C_t E_{tt'} {C_{t'}}^T C_{t'} {E_{tt'}}^T + D_t C_t}}$$

This completes the proof.

Note that in Eq. (4), when we compute the vertex to super nodes mapping C_t for t-type vertices, we utilize their connections to all the other t'-type vertices (i.e., $G_{tt'}$) and the vertex to super nodes mapping for all the other t'-type vertices (i.e., C'_t).

Similarly, the connections among super nodes $L_{tt'} \in R^{p_t \times p_{t'}}$ can be computed via the following Lemma:

Lemma 2. *The solution of $L_{tt'}$ can be approximated via the following multiplicative update rule:*

$$L_{tt'} = L_{tt'} \circ \sqrt{\frac{{C_t}^T G_{tt'} C_{t'}}{C_t^T C_t L_{tt'} {C_{t'}}^T C_{t'}}} \tag{5}$$

Proof (sketch): The proof is omitted since it is similar to that of Lemma 1.

To develop some intuition about the above solution, let us again consider the example in Figure 2. Assume that the product IDs 3 and 4 share many discriminative words in their respective descriptions. After the first iteration of the algorithm, this evidence will be captured by Lemma 1 and those nodes will be grouped together in $C_{Product}$ mapping. After this step, using Lemma 2, the links between the new super-node and other-type nodes will be updated. The updated links show that "*Bose*" and "*Bose Electronic*" nodes in the manufacturer type have a common neighbor in the product type (share the same product). This evidence, along with the similarity link between those two nodes, will be captured by Lemma 1, so that those two nodes will be clustered together.

4.3 Searching for the Optimal Number of Super Nodes

We have discussed the proposed algorithm that computes the "best-effort" summary graph and mapping between the original graph and the summary graph with the assumption that the number of super nodes in the summary graph is known in advance. However, a remaining challenge is to determine the actual number of entities (super nodes). A possible approach is to enumerate all the combinations of numbers of super nodes for each type of vertices and then pick the "best" one with an exhaustive search. Unfortunately, such trial-and-error procedures can be inefficient in practice. In the following, we propose a greedy local search algorithm that can automatically determine the number of super nodes for each type of vertices. The intuition of our approach is to utilize a backward search procedure: starting with an initialization of a summary graph, where each type of vertices is assigned to a maximum number of clusters, it repeatedly removes one or many super nodes from a summary graph with the lowest information. The details of the above procedure are presented in Algorithm 2, where $\texttt{Info}(s)$ denotes the information of a super node s.

Note that our algorithm differs from the traditional bottom-up merging or top-down split algorithm. Bottom-up merging iteratively picks two clusters such that merging of these two cluster leads to improved performance. Therefore, at each iteration, it requires to search over all cluster pairs, which is computationally very expensive (p^2 in a naïve implementation and $p \log p$ with a heap implementation). In contrast, in our search algorithm, we only have to decide whether a super node will be removed (lines 3–6), which results in a time-complexity that is linear in the number of super nodes p. For the top-down split algorithm, although the computational cost of searching for the best cluster

Algorithm 2 `Search`$(G, \mathcal{S}(G), p)$

Input: A k-type Graph G, a summary graph $\mathcal{S}(G)$
Output: A refined summary graph $\mathcal{S}_{new}(G)$
01: **for** each t-type super nodes and vertices
02: $\quad\theta = \min_{s \in S_t} \sum_{v \in V_t} C_t(v, s)$
03: $\quad$**for** each $s \in S_t$
04: $\quad\quad$`Info`$(s) = \sum_{v \in V_t} C_t(v, s)$
05: $\quad\quad$**if** `Info`$(s) == \theta$ and $(J(\mathcal{S}(G)) - J(\mathcal{S} \setminus \{s\}(G)) > 0$
06: $\quad\quad\quad$delete s from $\mathcal{S}(G)$
07: **return** $\mathcal{S}(G)$

to be split is linear, the algorithm requires sophisticated heuristics to perform a split, which entails reassigning each vertex from one cluster to one of two smaller clusters. In our algorithm, on the other hand, the vertices within a removed super node can be merged into the remaining super nodes through the procedure presented in Section 4.2.

4.4 Complexity Analysis

Table 2. The time complexity for each basic operator with both dense and sparse matrices representation. Here $(nz)_t$ is number of non-zero entries in the matrix $\mathtt{sim}_t$.

	Dense	Sparse
C_t	$O(n_t^2 p_t + n_t \sum_{t'>t}(n_t p_t + p_t p_t))$	$O((nz)_t p_t + \sum_{t'>t}(m_{tt'} + n_t q_{tt'}))$
$L_{tt'}$	$O(n_t p_t^2 + p_t(n_t n_{t'} + p_{t'} n_{t'} + p_t p_{t'}))$	$O(n_t p_t^2 + p_t(m_{tt'} + q_{tt'} + p_t n_{t'}))$

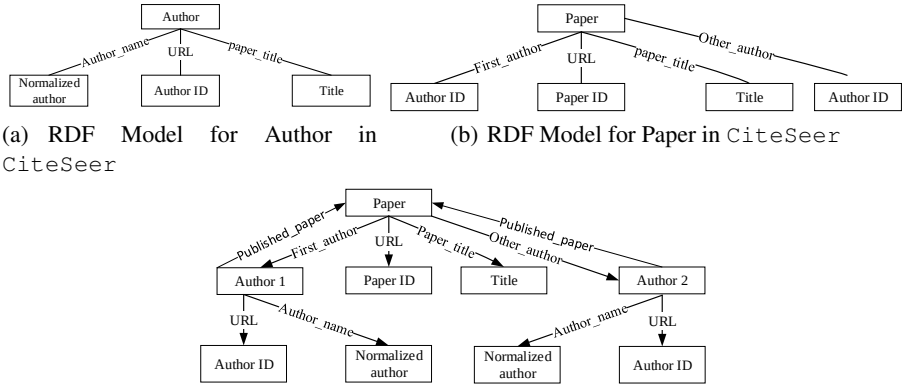
In this section, we analyze the time complexity of our proposed graph summarization algorithm. The time complexity for each basic operation is summarized in Table 2. In addition, for the Algorithm 2, the computational cost is dominated by the computation of the objective function in Line 7. Fortunately, instead of computing the objective function, we are only required to compute the change in the objective function after removing a super node. The differences (i.e., $J(\mathcal{S}(G)) - J(\mathcal{S} \setminus \{s\}(G))$) can be computed in time that is linear in the number of nodes. Therefore, the time complexity of `Search`$(G, \mathcal{S}(G), p)$ is $O(\sum_t p_t n_t)$.

With the above analysis, the overall time complexity of Algorithm 1 is $O(r_o r_i[\sum_t \sum_{t'>t} n_t n_{t'}(p_t + p_{t'}) + \sum_t (n_t)^2 p_t])$ for dense matrices and $O(r_o r_i[\sum_t \sum_{t'>t} m_{tt'} + q_{tt'}(n_t + n_{t'})])$ for sparse matrices, where r_i/r_o is number of iterations within inner loops (Line 4–10)/outer loops (Line 2–11). Both r_i and r_o are small in practice, which are around 20–200.

5 Experiments

5.1 Dataset and Comparable Methods

Data. We use two datasets from different domains: `Product` [19], and `Citeseer` [4]. `Product` consists of product entities from two online retailers *Amazon.com* and *Google Products*. Each record has attributes ID, title, description, manufacturer and price. The RDF schema of `Product` data is shown in Figure 1. Note that we only use a flat schema to model the `Product` data because we cannot retrieve many-to-many relations (e.g.,



(a) RDF Model for Author in `CiteSeer`

(b) RDF Model for Paper in `CiteSeer`

(c) RDF Model for Author and Paper in `CiteSeer`

Fig. 3. RDF Model Schema of `Citeseer` Data

many-to-many relations between products and manufacturers) due to the fact that only the product field has a unique identifier. Based on the schema, we create an input multi-type graph that consists of two types of vertices: product and word. Each product is connected to a word that appears in the title, manufacturer, and description. In addition, we also provide product to product similarity and word to word similarity. The available ground truth presents product equivalences but not manufacturer equivalences.

In the `Citeseer` data set, each publication has a title and multiple authors. We modeled the `Citeseer` data in two ways, with a multi-object schema preserving author-to-paper, paper-to-author, and author-to-author relations (see Figure 3(c)), and with a flat schema only preserving author-to-paper and paper-to-author relations (see Figure 3(a) and Figure 3(b)). Compared to the flat schema, the former model is more informative in terms that it supports accessing the list of co-authors in a paper for each author. However, some entity resolution systems need the data to be in CSV/XML format, and the latter flat model is more suitable for such systems. Based on the flat schema, we create a multi-type graph that consists of four types of vertices: normalized name, author ID, paper ID, and word. Each author ID is connected to its normalized name and its related paper ID; while each paper ID is connected to words from its title and authors. In addition, we also provide the author to author similarity and paper to paper similarity. The ground truths are whether two paper IDs refer to the same publication and whether two author IDs refer to the same author entity.

The statistics of two multi-type graphs are summarized in Table 3.

Table 3. The statistics of graphs.

Data	#types	# records	# nodes	# edges	# entities	Full input mapping
`Citeseer`	4	2892	8591	17521	author:1165, paper:899	8.4 Million
`Product`	2	4589	12397	41165	product:1104	4.4 Million

Comparable approaches. We compare our approach (**CoSum**) with representative state-of-the-art unsupervised entity resolution systems **Limes** [30], **Silk** [15], and **Serf** [3]. For `Product` data, we also report the best performance achieved by all the entity resolution approaches and unsupervised entity resolution approaches reported in the original paper that provide the data [19]. For `Citeseer` data, we report the best per-

Table 4. Configurations of different systems on `Product` Data.

	Name (N)	Price (P)	Description (D)	Manufacturer (M)	Acceptance Metric
Limes	Trigrams	Normalized difference	Cosine	-	N>0.6 AND P>0.5 AND D>0.5
Silk			Trigrams	Jaro	20N+10M+5P+D [2]
Serf			Jaccard+4-grams		N>0.6 AND P>0.5 AND D>0.5

formance achieved by the collective entity resolution method [4]. Moreover, Limes and Silk support reading data from a RDF store, which takes advantage of the graph representation and therefore more complicated data models. Thus, for Limes/Silk, we use **Limes-F/Silk-F** to denote running Limes/Silk using flat models (e.g., Figure 3(a) and Figure 3(b)), and **Limes-MO/Silk-MO** using multi-object models (e.g., Figure 3(c)).

Note that various graph summarization techniques have been proposed in terms of other purposes such as compressing minimum description length [33, 27, 24]. We also compare our approach to one representative minimum-description-length-based graph summarization approach **GSum** [33] in terms of entity resolution task.

Evaluation metrics. We evaluate the entity resolution quality using the usual measures: precision, recall, and F-measure. We also report the running time comparison of different approaches, although the comparison is unfair since they are implemented in different languages C++ (GSum), Java (Serf, Limes and Silk), and Matlab (CoSum). All the experiments are conducted on a single machine, with a 4-core 2.7GHZ CPU and 16 GB memory.

5.2 Configuration

Limes and Silk require a configuration file, describing the input/output format, as well as the acceptance metric and thresholds which determine whether a pair of records are the same or not. In Serf, the user is required to implement a decision-maker function that receives a pair of records and returns a true/false decision. For all the three systems, we need to determine which attribute/field to choose, their best similarity metrics, and how important their roles are in the acceptance decision. In our experiments, we tried our best to choose the best fitted metric functions based on what each system offers and the characteristic of data. Tables 4 and 5 illustrate a summary of the acceptance metrics for different systems on `Product` and `Citeseer` domains respectively. The details are described as follows.

We first select the set of attributes according to different systems. For instance, in Limes, the user first introduces all the attributes he wants to use for record comparison, and Limes requires all the specified attributes to be available in both records in order to compare them. As a result, we had to ignore the manufacturer name in `Product` domain, since more than 90% of the records in Google product dataset do not have the manufacturer name. The configuration in Silk is very similar to Limes, except that Silk allows the user to specify which attributes are not required for record-pair comparison and can be ignored if their value is missing. Therefore, we still use the manufacturer attribute in Silk. With the selected attributes and the details reported in the original work [4, 19] that provide these two benchmark datasets, we have tried combinations of

[2] In Silk, when choosing the weighted average score aggregation, the user just introduces rejection thresholds and weights for each attribute.

Table 5. Configurations of different systems on `Citeseer` data.

	Papers			Author		Acceptance Metric	
	Title (T)	First-author (F)	Authors (A)	Name (N)	Co-authors (C)	Papers	Authors
Limes-F	Trigrams	Jaro-Winkler	Jaccard	Jaro-Winkler	-	0.5T+0.4F+0.1A >0.75	N>0.85
Limes-MO					Jaccard		0.8N+0.2C>0.85
Silk-F			soft Jaccard		-	20T+20F+A	N>0.85
Silk-MO					soft Jaccard		6N+C
Serf					-	0.5T+0.4F+0.1A >0.75	N>0.85

various string and set similarity measures that are available in the systems (including Levenshtein, Jaro, N-grams, and Jaccard) as metrics. Finally, we perform multi-level grid search for optimal weights of attributes and the threshold. For instance, we search for the best acceptance threshold for Limes by a top-level grid search between [0, 1] with step size 0.2, following the bottom-level grid search with step size 0.01. Therefore, our manual configuration performs better than the active learning method within Silk because the learning method is based on a genetic algorithm (ActiveGenLink [14]).

For graph summarization approaches, the configuration is much easier. We do not need any acceptance metric since the decision is automatically given by the summary graph. In addition, if no domain knowledge is available, we could simply compute the similarity between the same-type vertices using graph proximity measures. In the experiments, CoSum-B denotes that the similarity between t-type vertices are computed using the weighted common neighbor approach proposed by [1]. That is, for each $x, y \in V_t$,

$$\texttt{sim}_t(x, y) = \sum_{z \in CN(x,y)} \frac{1}{\log d(z)} \tag{6}$$

where $CN(x, y)$ is the set of common neighbors shared by vertices x and y in the given k-type graph, and $d(z)$ denotes the weighted degree of vertex z. CoSum-P denotes that we use the string similarity metrics between the same-type vertices configured in Tables 4 and 5.

5.3 Quality Comparisons

In this section, we evaluate the performance of proposed approach in terms of precision, recall and F-measure for entity resolution tasks.

Question 1 ***F-measure****: How does CoSum perform compared to the state-of-the-art entity resolution systems?*

> **Result 1** *CoSum outperforms several existing state-of-the-art generic entity resolution systems (including Limes, Silk, Serf) in terms of F-measure. In addition, the quality is comparable to the best performance reported in the literature on both* `Citeseer` *and* `Product`*.*

Question 2 ***Algorithm****: How does the proposed CoSum perform compared to other graph summarization algorithms?*

Result 2 *The poor quality achieved by GSum shows that minimum-description-based graph summarization algorithms may not work well for the entity-resolution task. On the contrary, the significant improvement achieved by CoSum compared to Gsum verified the advantage of the proposed graph summarization algorithm.*

Table 6. Quality comparisons of different approaches.

	Precision			Recall			F-measure		
	Author	Paper	Product	Author	Paper	Product	Author	Paper	Product
Limes-F	0.958	0.827	0.446	0.864	0.761	0.160	0.909	0.792	0.236
Silk-F	0.846	0.877	0.459	0.986	0.756	0.348	0.910	0.812	0.395
Gsum	0.727	0.668	0.01	0.569	0.624	0.587	0.638	0.645	0.02
CoSum-B	0.993	0.871	0.58	0.940	0.611	0.477	*0.966*	*0.718*	*0.524*
Limes-MO	0.912	0.827	0.446	0.944	0.761	0.160	0.928	0.792	0.236
Silk-MO	0.932	**0.877**	0.459	0.958	0.756	0.348	0.945	0.812	0.395
Serf	0.985	0.837	0.436	0.687	0.808	0.186	0.809	0.822	0.261
CoSum-P	**0.999**	0.771	**0.639**	**0.997**	**0.997**	**0.695**	***0.998***	***0.87***	***0.666***
Best in Literature	NA	NA	0.615 [19]	NA	NA	0.63 [19]	0.995 [4]	NA	0.622 [19]

Question 3 ***Modeling****: What's the effect of modeling on state-of-the-art entity resolution systems?*

Result 3 *As shown in Table 6, both Limes and Silk are very sensitive to modeling. For publication entity resolution performance, Limes-MO and Silk-MO perform much better than Limes-F and Silk-F by using the multi-object modeling that captures the co-authorship information.*

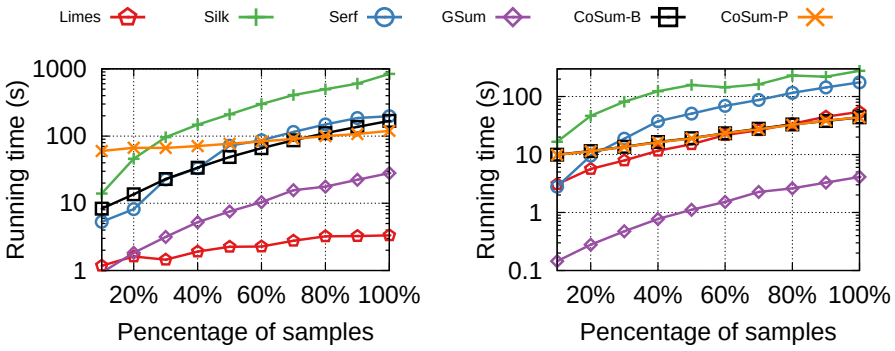
Question 4 ***Similarity****: What's the effect of similarity measures to the proposed CoSum approach?*

Result 4 *Compared to CoSum-B, CoSum-P achieves much better performance in disambiguating publication and product entities. This is because for publication, title information is a dominant feature and thus calculating paper to paper similarity using additional title trigram similarity improves the performance. Similarly, in* `Product` *data, using additional word-to-word similarity calculated with Jaro-Winkler captures correlation between noisy texts (e.g., Apple and Apple®)*

5.4 Efficiency Comparisons

In this section, we evaluate the scalability of the proposed approach. Unfortunately, examining the total running time only is unfair since the compared approaches are implemented in different languages: C++ (GSum), Java (Limes, Silk and Serf) and Matlab (CoSum). Therefore, we report how running time varies with the size of data to evaluate the scalability.

Question 5 ***Scalability with Sample Size****: How does the proposed CoSum scale compared to other approaches?*



(a) `Product` (b) `Citeseer`

Fig. 4. Running time comparisons for different approaches.

Result 5 *Among all the approaches, Limes is the most efficient and highly scalable with sample size. In terms of total running time, GSum is the most efficient because it is implemented in C++. However, we observed that the run-time for GSum, Serf and Silk scales super-linearly with the sample size, while for CoSum and Limes it scales almost linearly.*

Question 6 ***CoSum-B Versus CoSum-P****: How does the proposed CoSum scale compared to other approaches?*

Result 6 *Compared to CoSum-B, CoSum-P requires more I/O time on* `Product` *data because the word to word string similarity is much denser than weighted common neighbor similarity. However, the word to word string similarity is very informative and thus it helps convergence. Therefore, on product data, when the sample size is small, the I/O time is dominant for CoSum-P and thus CoSum-P is slower than CoSum-B. When the sample size becomes larger, the CPU time is dominant and thus CoSum-P is more efficient than CoSum-B (faster convergence). On* `Citeseer` *data, both paper-to-paper string similarity and paper-to-paper weighted common neighbor similarity are sparse. Therefore, the running time of CoSum-B and CoSum-P are very close.*

6 Conclusion

In this work, we proposed a multi-graph co-summarization-based method that simultaneously identifies entities and their connections. This framework is very generic, and does not require any domain-specific knowledge such as RDF modeling or tuning the pairwise similarity threshold. We applied the proposed approach to real multi-type graphs from different domains and obtained good results in terms of F-measure for entity-resolution tasks. The proposed method has some limitations. First, the quality of entity-resolution solution depends on the quality of the user-supplied same-type vertex similarity. We plan to extend the current method by adaptively refining the same-type vertex similarity with a small number of training samples. Second, if the same-type vertex similarity matrices and the observed graphs are very dense, the proposed algorithm is not scalable. In the future, we will improve the efficiency bottleneck by embedding the blocking techniques with the graph summarization algorithm. Finally, we plan to apply the graph summarization algorithm to the entity linking tasks, to evaluate the quality of super links in summary graphs.

Acknowledgments. This research is supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract number FA8750-14-C-0240. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL, or the U.S. Government.

References

1. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social networks 25(3), 211–230 (2003)
2. Arasu, A., Götz, M., Kaushik, R.: On active learning of record matching packages. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 783–794 (2010)
3. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widom, J.: Swoosh: A generic approach to entity resolution. The VLDB Journal 18(1), 255–276 (2009)
4. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Transactions on Knowledge Discovery from Data 1(1) (2007)
5. Bhattacharya, I., Getoor, L.: A latent dirichlet model for unsupervised entity resolution. In: SIAM International Conference on Data Mining (2007)
6. Brizan, D.G., Tansel, A.U.: A. survey of entity resolution and record linkage methodologies. Communications of the IIMA 6(3), 5 (2015)
7. Christen, P.: Automatic record linkage using seeded nearest neighbour and support vector machine classification. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 151–159 (2008)
8. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Transactions on Knowledge and Data Engineering 24(9), 1537–1555 (2012)
9. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 475–480 (2002)
10. Cudré-Mauroux, P., Haghani, P., Jost, M., Aberer, K., De Meer, H.: idMesh: Graph-based disambiguation of linked data. In: Proceedings of the 18th International Conference on World Wide Web. pp. 591–600 (2009)
11. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 126–135 (2006)
12. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 85–96 (2005)
13. Getoor, L., Machanavajjhala, A.: Entity resolution: theory, practice & open challenges. Proceedings of the VLDB Endowment 5(12), 2018–2019 (2012)
14. Isele, R., Bizer, C.: Active learning of expressive linkage rules using genetic programming. Web Semantics: Science, Services and Agents on the World Wide Web 23, 2–15 (2013)
15. Isele, R., Jentzsch, A., Bizer, C.: Silk server-adding missing links while consuming linked data. In: Proceedings of the First International Conference on Consuming Linked Data-Volume 665. pp. 85–96 (2010)
16. Ji, H., Nothman, J., Hachey, B.: Overview of TAC-KBP2014 entity discovery and linking tasks. In: Text Analysis Conference (2014)
17. Kejriwal, M., Miranker, D.P.: An unsupervised instance matcher for schema-free RDF data. Web Semantics: Science, Services and Agents on the World Wide Web 35, 102–123 (dec 2015)

18. Kolb, L., Thor, A., Rahm, E.: Load balancing for map-reduce based entity resolution. In: Proceedings of the IEEE International Conference on Data Engineering. pp. 618–629 (2012)
19. Köpcke, H., Thor, A., Rahm, E.: Evaluation of entity resolution approaches on real-world match problems. Proceedings of the VLDB Endowment 3(1-2), 484–493 (2010)
20. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability. pp. 481–492 (1950)
21. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. pp. 282–289 (2001)
22. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Proceedings of the Annual Conference on Neural Information Processing Systems. pp. 556–562 (2000)
23. Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., Jurafsky, D.: Deterministic coreference resolution based on entity-centric, precision-ranked rules. Computational Linguistics 39(4), 885–916 (2013)
24. LeFevre, K., Terzi, E.: Grass: Graph structure summarization. In: SIAM International Conference on Data Mining. pp. 454–465 (2010)
25. McCallum, A., Nigam, K., Ungar, L.H.: Efficient clustering of high-dimensional data sets with application to reference matching. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 169–178 (2000)
26. McCallum, A., Wellner, B.: Conditional models of identity uncertainty with application to proper noun coreference. In: Proceedings of the Annual Conference on Neural Information Processing Systems (2005)
27. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 419–432 (2008)
28. Neelakantan, A., Roth, B., McCallum, A.: Compositional Vector Space Models for Knowledge Base Completion (apr 2015), http://arxiv.org/abs/1504.06662
29. Ng, V., Cardie, C.: Improving machine learning approaches to coreference resolution. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. pp. 104–111 (2002)
30. Ngomo, A.C.N., Auer, S.: Limes: A time-efficient approach for large-scale link discovery on the web of data. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. pp. 2312–2317. AAAI Press (2011)
31. Pasula, H., Marthi, B., Milch, B., Russell, S.J., Shpitser, I.: Identity uncertainty and citation matching. In: Proceedings of the Annual Conference on Neural Information Processing Systems (2002)
32. Riedel, S., Yao, L., McCallum, A., Marlin, B.M.: Relation extraction with matrix factorization and universal schemas. In: HLT-NAACL (2013)
33. Riondato, M., Garcia-Soriano, D., Bonchi, F.: Graph summarization with quality guarantees. In: Proceedings of the IEEE International Conference on Data Mining. pp. 947–952 (2014)
34. Singla, P., Domingos, P.: Entity resolution with markov logic. In: Proceedings of the IEEE International Conference on Data Mining. pp. 572–582 (2006)
35. Whang, S.E., Menestrina, D., Koutrika, G., Theobald, M., Garcia-Molina, H.: Entity resolution with iterative blocking. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 219–232 (2009)
36. Winkler, W.E.: Matching and record linkage. Wiley Interdisciplinary Reviews: Computational Statistics 6(5), 313–325 (2014)
37. Zhu, L., Galstyan, A., Cheng, J., Lerman, K.: Tripartite graph clustering for dynamic sentiment analysis on social media. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 1531–1542 (2014)