Information Sciences 385–386 (2017) 438–456

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Integration of graphs from different data sources using crowdsourcing

Younghoon Kim[a], Woohwan Jung[b], Kyuseok Shim[b,*]

[a] Department of CS, Hanyang University, 55 Hanyangdaehak-ro, Sangnok-gu, Ansan, Republic of Korea
[b] Department of ECE, Seoul National University, Kwanak P.O. Box 34, Seoul 151-600, Republic of Korea

ARTICLE INFO

Article history:
Received 18 July 2016
Revised 29 December 2016
Accepted 2 January 2017
Available online 3 January 2017

Keywords:
Graph integration
Crowdsourcing
Entity resolution

ABSTRACT

Data integration is the process of identifying pairs of records from different databases that refer to the same entity in the real world. It has been extensively studied with regard to entity resolution, record linkage, duplicate detection or network alignment. With the increasing use of crowdsourcing platforms as a means of assessing queries manually at low cost, many studies have begun to consider ways to exploit crowdsourcing systems for efficient data integration.

In this paper, we present an efficient algorithm to integrate two graphs collected from different sources using crowdsourcing systems. Given two graphs, we repeatedly select a query node from a graph and request a human annotator to find its matching node from the other graph, which is considered to be the one indicating the same entity as the query node. The proposed method is to choose the query nodes that would increase the precision the most if it is labeled. By experiments with both the simulated answers and the labels collected by real crowdsourcing, we show that our algorithm finds more accurate graph matches with a smaller cost for crowdsourcing than the baseline algorithms.

1. Introduction

Graph integration is a type of entity resolution that merges at least two graphs obtained from different sources that concern the same real-world entities. Given a pair of different graphs, a general approach to graph integration is to select an object from a graph and determine its corresponding object appearing in the other graph that refers to the identical entity based on the attributes of nodes or the structural similarities in their neighborhood. Graph integration arises in many real world applications such as user linkage in different social media [25,47], XML schema integration [24,29] and protein structure matching [32].

Traditional studies on graph integration have investigated the automated integration of XML schemas [29] or network alignment [5,19]. The integration of XML schemas usually finds the identical elements between schemas but does not match individual instances or records between graphs. The goal of network alignment is to find the isomorphisms between different but similar graphs. Recently, network alignment has attracted much attention because of its many potential applications including the integration of different movie databases IMDB [3] and DBpedia [2], user linkage between different social media sites such as Facebook and Twitter, as well as the merging of the literature databases like DBLP and CiteULike. When

* Corresponding author.
E-mail address: shim@kdd.snu.ac.kr (K. Shim).

http://dx.doi.org/10.1016/j.ins.2017.01.006
0020-0255/© 2017 Elsevier Inc. All rights reserved.

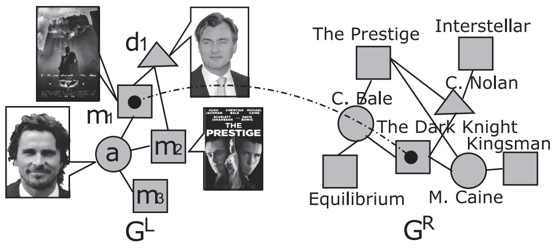


Fig. 1. Two movie DBs collected from different sources.

additional semi-identifiers such as entities' names are not available, (e.g., the integration of IMDB and a movie database compiled in a language other than English), identifying matching nodes between different graphs is more challenging because we must ask people to find matching pairs of nodes. Thus, crowdsourcing platforms such as Amazon's Mechanical Turk, which has emerged as a popular platform of collecting information from human annotators, are being exploited for entity resolution [4,35,36].

The main advantage of crowdsourcing is that it costs relatively little compared to the cost of hiring dedicated experts. In crowdsourcing systems, tasks are carried out by workers with little or no domain expertise at low price; thus, the smallest task units possible are usually assigned in order to acquire answers with a uniform quality. Because graph integration by finding matching nodes between two graphs can be easily split into multiple simple queries, crowdsourcing is a very suitable method for collecting labeled data to match graphs from difference sources.

In this paper, we develop an active learning algorithm to exploit crowdsourcing systems efficiently for integrating two graphs from different data sources. Active learning is used to repeatedly select the most informative unlabeled nodes in one graph, and ask annotators to find their matching nodes from the other graph in order to achieve high accuracy using the fewest number of questions possible. Furthermore, because annotators usually have little expertise in the given tasks, we must develop a robust graph matching algorithm to incorrectly labeled nodes.

Example 1.1. Suppose that we have the IMDB database, which is an online database of information related to movies including casts and directors, represented by $\mathbb{G}^R$ in Fig. 1. The left graph $\mathbb{G}^L$ in Fig. 1 represents another movie database that we want to integrate with $\mathbb{G}^R$, but we do not have any clues for matching each node of $\mathbb{G}^L$ to that of $\mathbb{G}^R$. Assume that we have already obtained a label indicating that node m_1 in $\mathbb{G}^L$ and *The Dark Knight* in $\mathbb{G}^R$ refer to the same movie. If an annotator is asked to find the matching node in $\mathbb{G}^R$ for the director d_1 in $\mathbb{G}^L$, the most effective way is to find any neighboring movie of d_1 that is already matched to a movie in $\mathbb{G}^R$, and search $\mathbb{G}^R$ for the matching director of the movie. Because m_1 in $\mathbb{G}^L$ corresponds to *The Dark Knight* in $\mathbb{G}^R$, the annotator would pair *Christopher Nolan* with d_1 among the neighboring directors of *The Dark Knight* in $\mathbb{G}^R$.

Considering this strategy for annotators to find the matching node for a given query node, will d_1 or a_1 be the best query node for the next question based on the given label? Since d_1 is the only director node of m_1, we can easily conjecture that d_1 is *Christopher Nolan*. Thus, a query about d_1 will provide little information for graph matching. If we inquire about a_1 instead of d_1 and receive the correct answer that a_1 is *Christian Bale*, we can additionally obtain the key clue that m_3 is matched with the movie *Equilibrium*. Thus, we must select the query node judiciously to minimize the number of questions to be asked for graph matching.

In previous works on entity resolution with crowdsourcing such as [35–37], the authors study de-duplication or record linkage algorithms mainly focusing on the problem of *reconciling the conflicting answers* by human annotators, and do not consider how to match the entities between graphs at minimum cost. Many network alignment algorithms [5,6,19] were developed to find the matching nodes between different graphs. However, the algorithms either *strongly rely on plenty of already-known matching node pairs* [5,6] or *do not use those pairs at all* [19]. Thus, it is not effective for them to be incorporated with active learning for crowdsourcing where we need to improve graph matching by gradually collecting already-known matching pairs.

Our contributions: To achieve the most accurate graph matching at the lowest cost, our proposed algorithm interactively improves the graph matching by asking about the most informative nodes first. We first propose a random walk model for computing the similarity between every pair of nodes each of which appears in different graphs. Based on the random walk model, we derive an EM algorithm to estimate the similarities using semi-supervised learning. We next suggest an active selection algorithm, which discovers the most informative node, to use as a query in crowdsourcing systems. Finally, we provide an efficient method for automated graph matching by using the similarities between nodes calculated based on our model.

To the best of our knowledge, the algorithm presented here is the first work for the problem of integrating graphs collected from different sources using crowdsourcing systems. Note that active learning based on crowdsourcing has been addressed as a seminal challenge in [14]. The contributions of this paper are summarized as follows:

- We introduce a new active learning problem to match two graphs collected from different sources.

	Without known matching pairs		With known matching pairs	
	Static	Active	Static	Active
Topology only	[19] and [39]	N/A	[5] and [6]	This work
Topology + node attributes	[18],[25],[47] and [48]	N/A	-	-

Fig. 2. Graph matching algorithms.

- We design an efficient strategy to exploit crowdsourcing platforms for matching graphs.
- We propose a random walk model to define the similarity between two nodes from different graphs and develop an inference algorithm to compute the similarities based on the idea of semi-supervised learning.
- We devise a method to select the most informative query that will most improve the quality of graph matching.
- Through extensive performance evaluations with both simulated crowdsourcing and actual crowdsourcing, we show that our algorithm achieves the most accurate graph matching with the lowest cost compared to the baseline algorithms.

The remainder of the paper is organized as follows: After discussing the related works in Section 2, we define the *active graph matching* problem in Section 3. We next provide an overview of our active graph matching algorithm in Section 4. Section 5 presents our random walk model and EM algorithm to compute matching probability distributions. In Section 6, we develop a method to select the best query and candidate nodes for crowdsourcing volunteers. We also describe how to match the nodes between two graphs with the matching pairs of nodes collected within the budget in Section 7. Finally, we present experimental results in Section 8 and conclude our study in Section 9.

2. Related work

We first examine traditional algorithms of entity resolutions for integrating different graphs and recent works on developing efficient crowdsourcing systems. Then, we examine the recent development of active learning algorithms.

Network alignment: Entity resolution is the process of identifying distinct objects to merge databases or remove duplicates in a database. It is highly useful in maintaining the integrity of databases by providing consistent data. Network alignment is an entity resolution problem seeking isomorphisms using the neighborhood topology between different but similar graphs, and has been under extensive investigation. While the algorithms that align graphs based on known matching pairs of nodes possibly including incorrect matches are presented in [5,6], the algorithms in [19,39] align graphs without any known matching pair of nodes. Furthermore, to identify user linkages across different social media sites, graph matching is tackled using not only the neighborhood topology but also user contents such as text messages or images in [18,25,47,48]. The studies in [25,47,48], however, are irrelevant to our work because they require user contents to discover identical nodes in social graphs. Note that such information is not provided for our problem.

As shown in Fig. 2, the above works can be classified according to the usage of known matching node pairs, the clues used for computing graph matching (i.e., whether they use topology only or additionally consider node attributes such as user contents in social networks) and the ability of the interactive collection for matching pairs. As we can see in Fig. 2, the alignment algorithms in [5,6,18,19] do not collect matching pairs actively; however, they can be extended to actively improve the performance by repeatedly estimating the matching probabilities between a pair of nodes from different graphs and collecting new matching pairs from human annotators. Thus, in this paper, we empirically evaluate the algorithms in [18,19] and the most accurate among those in [5] and [6] to show the effectiveness of our proposed algorithms.

In [28,34], novel techniques to predict user preferences by exploiting multiple social networks together were developed. Furthermore, the techniques of learning user preferences are applied for predicting volunteerism tendency in [33]. These works infer user interests from multiple social networks but do not provide a way to integrate those graphs by explicitly matching nodes between different graphs. Furthermore, matching multimedia data collected from different sources has been studied from the viewpoint of merging multiple graphs generated from video [41] or learning metric distance between images [44,45]. To use multimodal features for image ranking, the algorithm proposed in [42] learns the distance metric by integrating with the relevance scores, while [46] proposes deep multimodal distance metric learning by using click features. These algorithms, however, are difficult to generalize for active graph matching problems.

Crowdsourcing: Because crowdsourcing has been the subject of much recent attention as a way to efficiently exploit human resources for computation, many general frameworks to develop interactive human–machine systems have been proposed. In [12], a system called CrowdDB was suggested to integrate an extended SQL database with crowdsourcing systems such as Amazon's Mechanical Turk [1]. Qurk [27] was developed for human users to build queries and monitor their progress in a crowdsourcing system. CrowdSearch [43] is a human-aided image searching system that improves automated image search results by human validation. There are many other studies on crowdsourcing, but the best solution to utilize a crowdsourcing system varies depending on applications, data types and query forms.

Active learning: Active learning is a kind of semi-supervised machine learning in which we interactively request new data to users for further training. An extensive survey on active learning for multimedia annotation and retrieval has been carried out in [40]. Naturally, crowdsourcing and active learning are starting to be investigated for entity resolution to overcome the limitation of automated data integration by interactively collecting labels from human annotators. In [35–38], a hybrid human–machine system, which first automatically finds the matching pairs of records from two different tables and then verifies them using humans, was proposed. The record linking algorithm developed in [10] dynamically generates the questions to be queried in a crowdsourcing system based on a probabilistic framework. These systems have mainly focused on reconciling inconsistencies between collected labels, and do not provide an active learning algorithm to match graphs with the lowest cost. Furthermore, the problem of inconsistent labels is inherently considered in our probabilistic model. Note that no existing work has yet explored how to integrate different graphs using crowdsourcing.

3. Preliminaries

In this section, we first present the definitions to be used to describe the graphs obtained from two different data sources, and then provide the problem formulation of graph integration using crowdsourcing systems.

Observed data: Let $\mathbb{G}^L = (V^L, E^L)$ be a graph consisting of a vertex set V^L and an edge set E^L obtained from the first data source. Similarly, the graph collected from the second data source is denoted by $\mathbb{G}^R = (V^R, E^R)$. For the convenience of reference to the graphs, in this paper we will refer to $\mathbb{G}^L$ and $\mathbb{G}^R$ as the *left* graph and the *right* graph, respectively. We assume that V^L and V^R have the same set of possible node types, which is denoted by T (e.g., T={title, actor, director} for movie graphs). The node sets of type $t \in T$ in V^L and V^R are represented by V^L_t and V^R_t, respectively. Note that V^L_t for a type t may be different from V^R_t as they are generated from different databases. We use $N^L_{t,s}(u)$ to denote the set of neighboring nodes in V^L_s connected to $u \in V^L_t$. Similarly, the set of neighboring nodes of $v \in V^R_t$ with type s is $N^R_{t,s}(v)$.

Setting for crowdsourcing: To acquire the matching pairs of nodes between two graphs by turning to annotators via crowdsourcing systems, we may utilize binary comparative matching or n-ary comparative matching. Binary comparative matching is a form of question that suggests a pair of nodes ($u \in V_L$, $v \in V_R$) to ask whether they refer to the same entity, while n-ary comparative matching suggests a query node $u \in V_L$ and a set of n candidate nodes $C \subseteq V_R$ to request the selection of node matching u from C.

Binary comparative matching has been popularly used as a form of query to find the matching objects [17]. When we find matching pairs of nodes between two large graphs, however, most of the binary comparative queries will probably be answered negatively. Because negative answers hardly provide any useful information for matching graphs, n-ary comparative matching is more suitable for graph matching.

In the remainder of our paper, we assume that the left graph $\mathbb{G}^L$ is the graph from which we select a query node to ask annotators and the right graph $\mathbb{G}^R$ is the other graph from which we choose the n candidate nodes to be suggested to annotators. Furthermore, a node in the left graph is usually asked once. Because annotators may give the wrong answer, however, we assume that a node in the left graph can be selected multiple times in our system.

Integration of graphs with different schemas: The schemas of the graphs collected from different data sources may differ even if the graphs represent the same real-world domain. Consider the case in which two types of nodes in a graph are represented as a single type in another graph. For example, the type ‘cast’ in one movie graph may be separated as ‘actor’ and ‘actress’ in another one. In this case, it is inevitable to match the types ‘actor’ and ‘actress’ with the type ‘cast’ if we cannot obtain additional information to split the instances of type ‘cast’ to match with the nodes of type ‘actor’ and ‘actress’. Because schema matching has been extensively studied for relational database [8] and XML [29], we assume that we integrate the graphs from two different data sources after performing the schema matching between both schemas and transforming the graphs to have identical schema.

Problem definition: Our goal is to compute the most precise matches between two graphs by requesting annotators within a limited budget in a crowdsourcing system such as Amazon Mechanical Turk. Assuming that we pay a fixed reward to annotators for each query, we must not only select the most informative query nodes to ask to annotators, but also match two graphs as accurately as possible with the available labels obtained from the crowd.

Let $M = \{(u, \bar{\mathbf{w}}_{t,u})\}$ be the data set of already-matched nodes labeled by annotators, where u is a node in the left graph that was inquired at least once and $\bar{\mathbf{w}}_{t,u} = \{\bar{w}_{t,u\to v} | v \in V^R_t\}$ is the *observed matching probability distributions* of the node u. When a node $u \in V^L_t$ has been asked q_u times and matched with a node $v \in V^R_t$ repeatedly $q_{u,v}$ times by annotators, we calculate $\bar{w}_{t,u\to v}$ as $\frac{q_{u,v}}{q_u}$. Note that a node of type t in V^L_t cannot be matched with a node of a different type from t. We define the problem of *active integration of graphs from different sources*, or *active graph matching*, as follows:

Definition 3.1. Suppose that we have two graphs $\mathbb{G}^L$ and $\mathbb{G}^R$ which come from different sources. Given a data set M of already-matched nodes labeled by annotators, we define the problems of *active graph matching* as follows.

- (Similarity computation problem) Compute the most precise matches between the nodes of V^L and V^R based on M,
- (Query selection problem) Select a query node from V^L and the candidate nodes from V^R to ask annotators, which would provide the most useful information for matching two graphs in the next computation of matching nodes.

Example 3.2. Consider a pair of graphs in Fig. 3 which shows the relationships between movies, actors/actresses and directors (i.e., T = {movie, actor, director}) represented by squares, circles and triangles, respectively. Suppose that the left graph $\mathbb{G}^L$ is the internet database IMDB[3] and the right graph $\mathbb{G}^R$ is another movie database in a non-English language.

Suppose that annotators have already labeled that the actor a_3 in $\mathbb{G}^L$ is identical to a'_4 in $\mathbb{G}^R$. Without any additional information to match nodes, for the director d_1, it is definitely a better conjecture that d_1 matches with d'_1 in $\mathbb{G}^R$ rather than d'_2 because all movies starring a_3 and a'_4 are directed by d_1 and d'_1, respectively, but a'_4 did not star in any movie

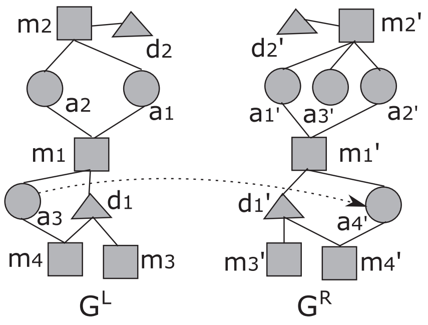


Fig. 3. Two graphs collected from different sources.

```
Function Active-Graph-Matching($\mathbb{G}^L$, $\mathbb{G}^R$, $M^0$, $budget$)
begin
1.  $M \leftarrow M^0$;
2.  while $budget > 0$ do
3.    compute the matching probabilities with $M$ using the EM algorithm (Section 5.3);
4.    $Q \leftarrow$ select $k_q$ query nodes from $\mathbb{G}^L$ based on the expected model change (Section 6);
5.    for each $u \in Q$ do
6.      $C \leftarrow$ select $k_c$ candidate nodes from $\mathbb{G}^R$ with the highest matching probabilities;
7.      $v \leftarrow$ get an answer from workers by asking to select a matching node of $u$ from $C$;
8.      $budget \leftarrow budget - 1$;
9.      if $v$ is not $null$ then $M \leftarrow M \cup \{(u, v)\}$;
10.   end for
11. end while
12. discover a matching node from $\mathbb{G}^R$ for each node in $\mathbb{G}^L$ (Section 7);
end
```

Fig. 4. *Active-Graph-Matching* algorithm.

directed by d'_2. For the movie m_1, it is difficult to decide which one between m'_1 and m'_4 is a better match for m_3 because a'_4 starred in both m'_1 and m'_4.

Let us assume that we ask an annotator to find the matching node of d_1. Then, even if the annotator answers that it corresponds to d'_1 in the right graph, no other important information for matching other nodes is provided. The reason is that we can guess that there is a high likelihood of d_1 matching d'_1 without any annotator's answer, while lacking the ability to distinguish among other matching nodes such as m_1 and m'_1.

If we ask annotators a query node a_2 instead and receive an answer that a_2 and a'_2 are identical, it becomes more certain that not only does a_2 match with a'_2 but m_1 also matches with m'_1 because the actor and director of m_1 are the same people as those of m'_1. Thus, selecting a_2 or m_1 as a query node is intuitively more informative than choosing d_1.

4. Overview of our crowdsourcing algorithm

We present our proposed algorithm *Active-Graph-Matching* which repeats the estimation of the matching probabilities based on already-matched node pairs, the selection of query nodes from $\mathbb{G}^L$ and the request to find their matching nodes from $\mathbb{G}^R$ to workers. Its pseudo code is shown in Fig. 4. We briefly summarize the process of our proposed algorithm as follows:

(1) Initially, we collect some matching pairs between the left and right graphs by randomly selecting query nodes from $\mathbb{G}^L$ and asking annotators to find their matching nodes from $\mathbb{G}^R$. Let M^0 be the initially obtained already-matched nodes and *budget* be the number of affordable tasks for crowdsourcing. Then, we call *Active-Graph-Matching* shown in Fig. 4.
(2) Based on the already-matched pairs of nodes M gathered so far, we first compute the matching probability distribution for each node in $\mathbb{G}^L$ over the nodes in $\mathbb{G}^R$ by the EM algorithm presented in Section 5.3. Then, we select the best k_q query nodes from $\mathbb{G}^L$, which are expected to most improve the accuracy of graph matching when answered. We choose the query nodes based on the expected model change which is described in Section 6.
(3) For each query node, we choose the best k_c candidate nodes from $\mathbb{G}^R$ to be suggested to workers in order of decreasing matching probabilities to the query node. Then, *budget* is decreased by 1. For each question, if workers discover a matching node among the candidates, they reply the answer and we append the matching pairs to M. If the query is not matched, the worker will answer with *null* indicating that there is no matching node in the candidates.
(4) We repeat the above steps (2) and (3) until we run out of the budget for crowdsourcing. Using the similarities calculated based on our model presented in Section 7, we match the remaining nodes in the left graph to those in the right graph.

Notation	Description
$\mathbb{G}^L=(V^L,E^L)$	The left graph from which we will select a query node
$\mathbb{G}^R=(V^R,E^R)$	The right graph from which annotators will find the node matching a given query node
V_t^L	The set of nodes with type t in the left graph, where $t \in T$ (Similarly, V_t^R is defined for the right graph)
$N_{t,s}^L(u)$	The set of nodes in V_s^L that are neighboring nodes of $u \in V_t^L$ ($N_{t,s}^R(u)$ is defined similarly)
$\bar{w}_{t,u\to v}$	The observed matching probability that $u \in V_t^L$ matches with $v \in V_t^R$
$\tilde{w}_{t,u\to v}$	The primitive matching probability that $u \in V_t^L$ matches with $v \in V_t^R$
$M=\{(u,\bar{\mathbf{w}}_{t,u})\}$	The set of labels consisting of the distributions $\bar{w}_{t,u\to v}$ for nodes $u \in V_t^L$ that have been queried at least once
$w_{t,u\to v}$	The matching probability, as a model parameter, that $v \in V_t^R$ is selected as the identical to $u \in V_t^L$
$\mathbf{w}_{t,u}$	The probability distribution of $w_{t,u\to v}$ over all v's in V_t^R
c_t	The confidence for type t ($c_t > 0$), which is a model parameter

Fig. 5. List of notations.

5. Matching nodes from different graphs

In this section, we propose a random walk model to compute the similarity between a pair of nodes from the left and right graphs. We next develop a semi-supervised learning algorithm to compute the similarity values based on the random walk model when we are given a set of already-matched nodes collected gradually by asking human annotators. The notations used in this paper are summarized in Fig. 5. In Section 5.1, we first define the *primitive matching probability* $\tilde{w}_{t,u\to v}$ based on our random walk model. We next expand the model to incorporate the already matched pairs for active learning and propose an algorithm to compute the *matching probability* $w_{t,\,u\,\to\,v}$ which will be used as the similarity between $u \in V_t^L$ and $v \in V_t^R$ in Section 5.2 and Section 5.3, respectively.

5.1. Random walk model

Similar to the other random walk models such as [15] and [21], our proposed random walk model estimates the similarity between two nodes by measuring how probable a random surfer starting from a node is expected to find the other node. However, our random walk model is designed to compute the proximity of two nodes from different graph where a set of already-matched pairs labeled by annotators exists.

We regard the probability, with which an annotator reaches to a node v in the right graph by randomly following the neighboring nodes starting from a node u in the left graph, as the similarity between a pair of nodes u and v. For nodes $u \in V_t^L$ and $v \in V_t^R$ of the same type t, let $w_{t,\,u\,\to\,v}$ denote the *matching probability* that annotators would select v among all nodes in V_t^R as identical to u. Note that $\sum_{v\in V_t^R} w_{t,u\to v} = 1$ holds for every node u in V_t^L. We will use $\mathbf{w}_{t,u}$ to denote the probability distribution $\{w_{t,u\to v}|v \in V_t^R\}$. Let c_t be the confidence of the type t which represents the probability that an annotator would choose a neighboring node of type t on the route to the matching node. Because the connectivity between a pair of nodes may differ depending on the node types (e.g., the connectivity between directors and movies is sparser than that between actors and movies), we take the probability c_t, which satisfies $\sum_{t\in T} c_t = 1$, into consideration in our random walk model. In this section, we first define the *primitive matching probability* $\tilde{w}_{t,u\to v}$ based on our random walk model. We will discuss how to compute the *matching probability* $w_{t,\,u\,\to\,v}$ for the semi-supervised learning in the following sections. Given a query node $u \in V_t^L$, an annotator reaches a node $v \in V_t^R$ with the probability $\tilde{w}_{t,u\to v}$ by the following recursive steps:

a. The annotator first chooses one of the types s with the probability c_s and then selects a neighboring node x from $N_{t,s}^L(u)$ with the uniform probability distribution (i.e., $1/|N_{t,s}^L(u)|$).
b. The annotator next visits a node $y \in V_s^R$ which is a neighboring node of v in the right graph with the probability $\tilde{w}_{s,x\to y}$.
c. Finally, the annotator reaches v if the node v is selected among y's neighboring nodes in $N_{s,t}^R(y)$ with the probability $1/|N_{s,t}^R(y)|$.

Example 5.1. Consider a pair of graphs in Fig. 6. Given a query node m_1 in $\mathbf{G}^L$, annotators will find m_1' in $\mathbf{G}^R$ by following the edges. Suppose that an annotator decides to visit one of m_1's neighbors of type *actor* with the probability c_{actor} and chooses a_1 among $N_{movie,actor}^L(m_1)$ with the probability $1/|N_{movie,actor}^L(m_1)|$. After the annotator discovers the actor a_1' in $\mathbf{G}^R$ is the matching node of a_1 with the probability $\tilde{w}_{actor,a_1\to a_1'}$, the annotator will finally reach m_1' with the probability $1/|N_{actor,movie}^R(a_1')|$. Thus, the probability to find m_1' starting from m_1 by following the route $m_1 \to a_1 \to a_1' \to m_1'$ is

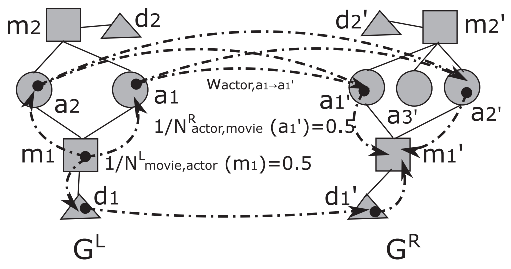


Fig. 6. A random walk from m_1 to m'_1.

$\frac{c_{actor}\tilde{w}_{actor,a_1\rightarrow a'_1}}{|N^L_{movie,actor}(m_1)||N^R_{actor,movie}(a'_1)|}$. The matching probability $\tilde{w}_{actor,m_1\rightarrow m'_1}$ becomes the sum of the probability of following every possible path from m_1 to m'_1.

Based on the random walk model, the primitive matching probability $\tilde{w}_{t,u\rightarrow v}$ is defined with the recursive formula below:

$$\tilde{w}_{t,u\rightarrow v} = \sum_{s\in T} \frac{c_s}{|N^L_{t,s}(u)|} \sum_{x\in N^L_{t,s}(u)} \sum_{y\in N^R_{t,s}(v)} \frac{\tilde{w}_{s,x\rightarrow y}}{|N^R_{s,t}(y)|} \tag{1}$$

We denote the probability distribution of $\tilde{w}_{t,u\rightarrow v}$ over every vertex $v \in V^R_t$ by $\tilde{\mathbf{w}}_{t,u}$. We next show that $\tilde{w}_{t,u\rightarrow v}$, which is computed by Eq. (1) over all vertices $v \in V^R_t$, can be regarded as a probability distribution without using any normalization term.

Lemma 5.2. *(Unitarity) For each $u \in V^L_t$, the sum of $\tilde{w}_{t,u\rightarrow v}$ over every v in V^R_t is 1. In other words,*

$$\sum_{v\in V^R_t} \tilde{w}_{t,u\rightarrow v} = 1.$$

Proof. We sum $\tilde{w}_{t,u\rightarrow v}$ for all v's in V^R_t as follows

$$\sum_{v\in V^R_t} \tilde{w}_{t,u\rightarrow v} = \sum_{s\in T} \frac{c_s}{|N^L_{t,s}(u)|} \sum_{x\in N^L_{t,s}(u)} \sum_{v\in V^R_t} \sum_{y\in N^R_{t,s}(v)} \frac{\tilde{w}_{s,x\rightarrow y}}{|N^R_{s,t}(y)|}.$$

Because summing $\tilde{w}_{s,x\rightarrow y}/|N^R_{s,t}(y)|$ over all pairs (v, y) with $v \in V^R_t$ and $y \in N^R_{t,s}(v)$ implies that we examine every edge between V^R_t and V^R_s one at a time, it is equivalent to summing $\tilde{w}_{s,x\rightarrow y}/|N^R_{s,t}(y)|$ over all pairs (v, y) with $y \in V^R_s$ and $v \in N^R_{s,t}(y)$. Thus, we obtain that $\sum_{v\in V^R_t} \tilde{w}_{t,u\rightarrow v}$ is equal to

$$\sum_{s\in T} \frac{c_s}{|N^L_{t,s}(u)|} \sum_{x\in N^L_{t,s}(u)} \sum_{y\in V^R_s} \tilde{w}_{s,x\rightarrow y} \sum_{v\in N^R_{s,t}(y)} \frac{1}{|N^R_{s,t}(y)|} = 1.$$

□

Note that Eq. (1) is a recursive formula for $\tilde{w}_{t,u\rightarrow v}$ based on the random walk model without considering any already-matched pairs of nodes collected so far. If $u \in V^L$ is not yet matched with any node in the right graph, we must follow neighboring nodes by a random walk resulting in the primitive matching probability $\tilde{w}_{t,u\rightarrow v}$ in Equation (1). If u has been already labeled by annotators and we have its observed matching probability distribution $\bar{\mathbf{w}}_{t,u}$ in M, however, $w_{t,\, u\, \rightarrow\, v}$ is required to be close to $\bar{w}_{t,u\rightarrow v}$ as well as $\tilde{w}_{t,u\rightarrow v}$.

In the following section, we develop a semi-supervised learning algorithm to utilize already-matched pairs when estimating the matching probability distribution of a node on the left graph.

5.2. Estimation of the matching probability

Semi-supervised learning algorithms usually define an objective function in terms of *the model cost* to be minimized if the model parameters best fit the given data for both labeled and unlabeled ones. Furthermore, it contains *a loss* which represents the error between the labeled data and the predicted labels based on the model so that the model parameters for the labeled data are in fact fixed according to their given labels [49].

Given a label set of already-matched nodes, which have been collected by querying the annotators so far, we design our model by using the following intuition:

1. The matching probability $w_{t,\, u\, \rightarrow\, v}$ should be similar to $\tilde{w}_{t,u\rightarrow v}$ computed by Eq. (1), regardless of whether u was already selected as a query node.

2. If u has been already selected as a query node, $w_{t,\,u\,\rightarrow\,v}$ should be as close to the observed matching probability $\bar{w}_{t,u\rightarrow v}$ as possible.

To formulate the model cost and loss terms by considering the above intuition in our objective function for computing the matching probability, we utilize *Bhattacharyya distance* [7], which is widely used to measure the similarity between two discrete or continuous probability distributions in many applications such as image processing and speech recognition [9].

Bhattacharyya distance: Let $\mathbf{p} = \{p_1, \ldots, p_n\}$ and $\mathbf{q} = \{q_1, \ldots, q_n\}$ be two discrete probability distributions with the same sample space S. The Bhattacharyya distance [7] between $\mathbf{p}$ and $\mathbf{q}$, denoted by $B(\mathbf{p}, \mathbf{q})$, is defined as

$$B(\mathbf{p}, \mathbf{q}) = -\log \sum_{i\in S} \sqrt{p_i q_i}. \tag{2}$$

For instance, suppose that we have three distributions $\mathbf{p} = \{0.2, 0.8\}$, $\mathbf{q}_1 = \{0.1, 0.9\}$ and $\mathbf{q}_2 = \{0.9, 0.1\}$. As we can expect, because $\mathbf{p}$ is more similar to $\mathbf{q}_1$ than $\mathbf{q}_2$, $B(\mathbf{p}, \mathbf{q}_1) = -\log(\sqrt{0.2\cdot 0.1} + \sqrt{0.8\cdot 0.9}) = 0.01$ is smaller than $B(\mathbf{p}, \mathbf{q}_2) = -\log(\sqrt{0.2\cdot 0.9} + \sqrt{0.8\cdot 0.1}) = 0.35$.

Objective function: Using Bhattacharyya distance, we define our objective function as follows:

$$L = -\sum_{\substack{t\in T,\\ u\in V_t^L}} B(\mathbf{w}_{t,u}, \tilde{\mathbf{w}}_{t,u}) - \mu \sum_{(u,\bar{\mathbf{w}}_{t,u})\in M} B(\mathbf{w}_{t,u}, \bar{\mathbf{w}}_{t,u})$$

where μ is a constant, s.t. $\mu > 0$, for regularization to control the trade-off between the two different terms. The first term is to compute the matching probability based on the proposed random walk model according to the first intuition. The second term is to minimize the difference between the matching probability computed by the random walk model and the observed matching probability, reflecting the second intuition.

Given a set of already-matched nodes M and a regularization constant μ, our optimization problem is to compute the matching probability $\mathbf{w}_{t,\,u}$ for every pair of $t \in T$ and $u \in V_t^L$ and the confidence c_t for every $t \in T$ that maximizes

$$\sum_{\substack{t\in T,\\ u\in V_t^L}} \log \sum_{v\in V_t^R} \sqrt{w_{t,u\rightarrow v} \sum_{s\in T} \frac{c_s}{|N_{t,s}^L(u)|} \sum_{\substack{x\in N_{t,s}^L(u),\\ y\in N_{t,s}^R(v)}} \frac{w_{s,x\rightarrow y}}{|N_{s,t}^R(y)|}} + \mu \sum_{(u,\bar{\mathbf{w}}_{t,u})\in M} \log \sum_{v\in V_t^R} \sqrt{w_{t,u\rightarrow v}\bar{w}_{t,u\rightarrow v}} \tag{3}$$

subject to

$$\sum_{t\in T} c_t = 1 \text{ and } \forall u \in V_t^L, \sum_{v\in V_t^R} w_{t,u\rightarrow v} = 1. \tag{4}$$

As with $\tilde{w}_{t,u\rightarrow v}$ defined in Eq. (1), note that we need not to normalize the first term of Eq. (3) due to Lemma 5.2.

5.3. Computation of similarity

In our inference algorithm, we compute the model parameters $\mathbf{w}_{t,u}$'s and c_t's that maximize the objective function in Eq. (3) through the Expectation-Maximization(EM) algorithm [11,26]. According to the EM algorithm, we define an arbitrary probability distribution α_{uv}^t, β_{uv}^t and $\delta_{uvxy}^{t,s}$ by Eqs. (5)–(7) in Fig. 8. By exploiting Jensen's inequality, we first obtain the lower bound $\mathbb{F}$ of our objective function as follows:

$$\mathbb{F} = \frac{1}{2}\Bigg[\sum_{t\in T}\sum_{u\in V_t^L}\sum_{v\in V_t^R} \alpha_{uv}^t \log w_{t,u\rightarrow v} + \sum_{t\in T}\sum_{u\in V_t^L}\sum_{v\in V_t^R}\sum_{s\in T}\sum_{x\in N_{t,s}^L(u)}\sum_{y\in N_{t,s}^R(v)} \alpha_{uv}^t \delta_{uvxy}^{t,s} \frac{c_s w_{s,x\rightarrow y}}{|N_{t,s}^L(u)||N_{s,t}^R(y)|} + \mu I_{(u,\bar{w}_{t,u})\in M} \sum_{t\in T}\sum_{u\in V_t^L}\sum_{v\in V_t^R} \beta_{uv}^t (\log w_{t,u\rightarrow v} + \log \bar{w}_{t,u\rightarrow v})\Bigg]$$

where $I_{condition}$ denotes the indicator function, equal to 1 if the condition holds and 0 otherwise.

To maximize $\mathbb{F}$ subject to the constraints in Eq. (4), we set the derivative of the Lagrangian function of $\mathbb{F}$ with respect to each of $\mathbf{w}_{t,\,u}$ and c_t to zero resulting in Eq. (8) and (9). Then, based on the formulas in Fig. 8, the EM algorithm repeatedly updates α_{uv}^t, β_{uv}^t and $\delta_{uvxy}^{t,s}$ in each E-step as well as $\mathbf{w}_{t,\,u}$ and c_t in each M-step until the objective function $\mathbb{F}$ converges.

Example 5.3. Given two graphs and already-matched labels as shown in Fig. 3, the matching probabilities computed by our EM algorithm are presented in Fig. 7. As we expected in Example 3.2, d_1 and d_2 match d_1' and d_2' with a high probability. Furthermore, we cannot determine the matching node of m_3 with the current matching labels at this moment.

Reducing the time and space complexities: Computing the matching probability for every possible pair of nodes from V^L and V^R requires $O(|V^L| \cdot |V^R|)$ space and $O(n_{steps} \cdot |V^L| \cdot |V^R| \cdot \hat{N}^L \cdot \hat{N}^R)$ time, where n_{steps} is the number of EM steps executed until the objective function $\mathbb{F}$ converges, and $\hat{N}^L$ and $\hat{N}^R$ are the maximum sizes of neighbors among V^L and V^R, respectively.

u \ v	m1'	m2'	m3'	m4'
m1	0.28	0	0.06	0.66
m2	0	1	0	0
m3	0.33	0	0.33	0.33
m4	0.25	0	0.17	0.58

(a) movie

u \ v	a1'	a2'	a3'	a4'
a1	0.22	0.22	0.17	0.39
a2	0.22	0.22	0.17	0.39
a3	0	0	0	1

(b) actor

u \ v	d1'	d2'
d1	1	0
d2	0	1

(c) director

Fig. 7. Matching probabilities.

E-step:

$$\alpha^t_{uv} \propto \sqrt{w_{t,u\to v} \sum_{s\in T} \frac{c_s}{|N^L_{t,s}(u)|} \sum_{x\in N^L_{t,s}(u)} \sum_{y\in N^R_{t,s}(v)} \frac{w_{s,x\to y}}{|N^R_{s,t}(y)|}} \tag{5}$$

$$\beta^t_{uv} \propto \sqrt{w_{t,u\to v}\bar{w}_{t,u\to v}} \tag{6}$$

$$\delta^{t,s}_{uvxy} \propto \frac{c_s w_{s,x\to y}}{|N^L_{t,s}(u)||N^R_{s,t}(y)|} \tag{7}$$

M-step:

$$w_{t,u\to v} \propto \alpha^t_{uv} + \sum_{s\in T} \sum_{x\in N^L_{t,s}(u)} \sum_{y\in N^R_{t,s}(v)} \alpha^s_{xy}\delta^{s,t}_{xyuv} + \mu I_{(u,\bar{\mathbf{w}}_{t,u})\in M}\beta^t_{uv} \tag{8}$$

$$c_t \propto \sum_{s\in T} \sum_{u\in V^L_s} \sum_{v\in V^R_s} \sum_{x\in N^L_{s,t}(u)} \sum_{y\in N^R_{s,t}(v)} \alpha^s_{uv}\delta^{s,t}_{uvxy} \tag{9}$$

Fig. 8. E-step and M-step.

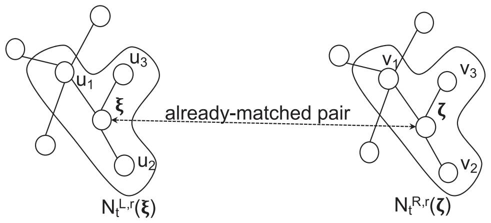


Fig. 9. Approximate computation of matching probabilities with r=1.

To address the scalability problem, we compute the matching probabilities between the nodes located within a fixed radius from an already-matched node pair only, similarly to the approximate algorithm for SimRank in [15]. Our method not only reduces the cost of computing the matching probabilities over V^L and V^R by neglecting the computation of very small matching probabilities but also approximates effectively the matching probabilities, as our experimental results will show later. Our approximate method works as follows. Let $N_t^{L,r}(u)$ be the node set in the left graph such that every node in $N_t^{L,r}(u)$ can reach $u \in V_t^L$ within the distance r. Similarly, we define $N_t^{R,r}(v)$ for the right graph. For a node u of type t, we compute $w_{t,\,u\to v}$ only for the nodes $v \in V_t^R$ where there exists an already-matched node pair ($\xi \in V_s^L, \zeta \in V_s^R$) for every pair of $u \in N_s^{L,r}(\xi)$ and $v \in N_s^{R,r}(\zeta)$. For the rest of the nodes $v \in V_t^R$, we regard every $w_{t,\,u\to v}$ as 0 and do not compute it. For example, consider a pair of homogeneous graphs with an already-matched node pair (ξ, ζ) in Fig. 9. We compute $w_{t,\,u\to v}$ only for $u \in N_s^{L,r}(\xi) = \{u_1, u_2, u_3, \xi\}$ and $v \in N_s^{R,r}(\zeta) = \{v_1, v_2, v_3, \eta\}$ where r is set to 1.

Because we compute the matching probability for each pair of nodes in both graphs that have appeared in M at each step of inquiring to workers, we compute the matching probability distribution only for $O(\hat{N}^L \cdot |M|)$ nodes in $\mathbb{G}^L$ over $O(\hat{N}^R \cdot |M|)$ nodes in $\mathbb{G}^R$. Thus, the EM algorithm costs $O(|M|^2 \cdot \hat{N}^L \cdot \hat{N}^R)$ memory and $O(n_{steps} \cdot |M|^2 \cdot (\hat{N}^L)^2 \cdot (\hat{N}^R)^2)$ time in each step.

6. Selection of query and candidate nodes

In this section, we develop a method to select the best query node from the left graph, as well as the candidate nodes from the right graph that are the most probable matches with the selected query node.

Selection of query nodes by expected model change: A popular strategy for selecting the questions to be asked in active learning is *uncertainty sampling* which chooses the most uncertain query, i.e., the one for which it is the most difficult to guess its label to be answered by annotators [23]. Based on our model in Section 5, uncertainty sampling simply selects the node in the left graph whose matching probability distribution has maximum entropy. That is, we choose a query node u^* such that $u^* = \arg\max_{u \in V_t^L} \sum_{v \in V_t^R} -w_{t,u\to v} \log w_{t,u\to v}$.

Another efficient approach for selecting queries utilizes the *expected model change*. Its goal is to choose the instances that are expected to change of the model over every possible resulting label the most [23]. This strategy has been shown to be very efficient in many empirical studies [30,31], but computing the model change whenever a query is selected can be quite expensive.

We next define the expected model change based on our random walk model described previously in Section 5. To reduce the computational cost, we develop an approximate method to calculate the expected model changes.

Expected model change: Let $w_{s,\, x \to y|t,\, u \to v}$ be the *conditional matching probability* with which the node x in V_s^L is matched with y in V_s^R if an annotator answered that $u \in V_t^L$ is identical to $v \in V_t^R$. We use $\mathbf{w}_{s,\, x|t,\, u \to v}$ to denote the conditional probability distribution of $w_{s,\, x \to y|t,\, u \to v}$ for every $y \in V_s^R$. Let $w_{s,\, x \to y}$ be the current matching probability as defined in Section 5.

We compute the expected model change based on the Bhattacharyya distance, which has been used in our model to compute the distance between the probability distributions. Assuming that we obtained a matching pair $(u \in V_t^L, v \in V_t^R)$, let $B_{sum}(t, u, v)$ be the sum of Bhattacharyya distance between $\mathbf{w}_{s,\, x}$ and $\mathbf{w}_{s,\, x|t,\, u \to v}$ for all $s \in T$ and $x \in V_s^L$. The expected model change for selecting a query node u from V_t^L is the expectation of $B_{sum}(t, u, v)$ for every node $v \in V_t^R$ which is calculated as

$$\sum_{v \in V_t^R} w_{t,u\to v} \left[\sum_{s \in T} \sum_{x \in V_s^L} -\log \sum_{y \in V_s^R} \sqrt{w_{s,x\to y} w_{s,x\to y|t,u\to v}} \right] \tag{10}$$

Among every node $u \in V_t^L$, we choose the query node u whose expected model change computed with Eq. (10) is the largest.

Approximate expected model change: To select a node with the maximum expected model change, we need to perform the EM algorithm on every possible match between $u \in V_t^L$ and $v \in V_t^R$. This, however, is too expensive. Thus, instead of using the EM algorithm, when u is matched with v, we approximately compute $w_{s,\, x \to y|t,\, u \to v}$ based on Eq. (1) by considering only the nodes whose matching probabilities would significantly change. Our assumptions are as follows:

- *For the node u:* For every $v' \in V_t^R$, let $\bar{w}^+_{t,u\to v'}$ denote the observed matching probability updated by considering a new matching node pair (u, v). We assume that the conditional matching probability $w_{t,u\to v'|t,u\to v}$ is simply identical to $\bar{w}^+_{t,u\to v'}$.
- *For the node $x \in N^L_{t,s}(u)$:* We assume that the matching probability distribution $\mathbf{w}_{s,\, x|t,\, u \to v}$ is updated for every $x \in N^L_{t,s}(u)$ only. For such a node $x \in N^L_{t,s}(u)$, we also assume that $w_{s,\, x \to y|t,\, u \to v}$ is increased only for every $y \in N^R_{t,s}(v)$. Thus, $w_{s,\, x \to y|t,\, u \to v}$ is approximately computed based on Eq. (1) as follows:

$$w_{s,x\to y|t,u\to v} = w_{s,x\to y} + \frac{c_t}{|N^L_{s,t}(x)|} \sum_{\zeta \in N^R_{s,t}(y)} \frac{\Delta w_{t,u\to\zeta}}{|N^R_{t,s}(\zeta)|} \sim w_{s,x\to y} + \frac{c_t}{|N^L_{s,t}(x)|} I_{(y \in N^R_{t,s}(v))} \frac{\bar{w}_{t,u\to v} - w_{t,u\to v}}{|N^R_{t,s}(v)|}$$

where $w_{s,\, x \to y|t,\, u \to v}$ is normalized such that $\sum_{y \in V_s^R} w_{s,x\to y|t,u\to v} = 1$.

According to the above assumptions, we update the matching probability $w_{s,\, x \to y}$ only when either $x = u$ or $x \in N^L_{t,s}(u)$. Then, we can reduce the computation of the expected model changes in Eq. (10) by summing the Bhattachayya distances only when $x \in N^L_{t,s}(u)$ for every $s \in T$ or $x = u$ instead of summing for every pair $s \in T$ and $x \in V_s^L$.

Example 6.1. Consider the graphs in Fig. 3. Based on the matching probabilities in Fig. 7, the expected model changes of all nodes are listed in Fig. 10(a). The expected model changes for d_1 and d_2, which are zeros, imply that asking those nodes will not provide any informative clue to match other nodes. Fig. 10(b) shows the matching probabilities computed by our EM algorithm after adding a matching label (d_1, d'_1) to the set of already-matched node pairs M. The result is exactly the same as the matching probabilities shown in Fig. 7. Thus, it confirms that a query with d_1 does not provide any useful information at all.

Selection of candidate nodes using matching probability: We next select the candidate nodes to be suggested to annotators based on the matching probabilities calculated with our EM algorithm presented in Section 5 as follows: for each query node $u \in V_t^L$, we choose the candidate nodes from V_t^R by simply selecting the top-k_c nodes with the k_c largest matching probabilities of $w_{t,\, u \to v}$.

7. Node matching with similarity

We may utilize the Hungarian algorithm [20] for the optimal assignment but it takes too much time. Thus, we exploit a greedy algorithm [13] to match nodes with the matching probabilities (i.e., similarity). In [13], this greedy algorithm is shown to be faster than the Hungarian algorithm and to find a nearly optimal assignment. We greedily match the pairs of nodes between V_t^L and V_t^R for each type t: (1) we first enumerate all pairs of nodes $(u \in V_t^L, v \in V_t^R)$ in the decreasing order of $w_{t,\, u \to v}$, (2) for each pair (u, v), we simply match u to v if neither u nor v has been assigned to another node so far.

(a)

	EMC
m1	0.42
m2	0
m3	0.55
m4	0.48
a1	0.67
a2	0.67
a3	0
d1	0
d2	0

(b) With a1 = a1'

u \ v	m1'	m2'	m3'	m4'
m1	0.49	0	0.07	0.44
m2	0	1	0	0
m3	0.33	0	0.33	0.33
m4	0.24	0	0.16	0.6

u \ v	a1'	a2'	a3'	a4'
a1	0.98	0.01	0	0.01
a2	0.26	0.26	0.17	0.31
a3	0	0	0	1

u \ v	d1'	d2'
d1	1	0
d2	0	1

(c) With d1=d1'

u \ v	m1'	m2'	m3'	m4'
m1	0.28	0	0.06	0.66
m2	0	1	0	0
m3	0.33	0	0.33	0.33
m4	0.25	0	0.17	0.58

u \ v	a1'	a2'	a3'	a4'
a1	0.22	0.22	0.17	0.39
a2	0.22	0.22	0.17	0.39
a3	0	0	0	1

u \ v	d1'	d2'
d1	1	0
d2	0	1

Fig. 10. Expected model changes and matching probabilities of the next stage.

Example 7.1. Let us reconsider a pair of graphs in Fig. 3. Suppose that we had requested an annotator to find the matching node of a_1 whose expected model change is the largest, as illustrated in Example 6.1, and the annotator correctly discovered a'_1. Fig. 10(b) presents the matching probabilities computed by adding the matching label (a_1, a'_1) to M. We first match d_1, d_2, m_2 and a_3 with d'_1, d'_2, m'_2 and a'_3, respectively because their matching probabilties are the largest among all pairs, having value of 1. Because a_1 has the second largest value of $w_{t,a_1\to a'_1} = 0.98$, we match a_1 with a'_1. Next, m_4 is matched with m'_4 and, m_1 is matched with m'_1. Finally, the remaining nodes m_3 and a_2 are matched with m'_3 and a'_4, respectively.

8. Experiments

We empirically evaluated the performance of our proposed algorithms. All experiments reported were performed on a computer with Intel i5 3.30 GHz CPU and 32 GB of main memory. All algorithms were implemented with version 1.7 of Javac Compiler.

8.1. Implemented algorithms

For our experiments, we implemented six algorithms to compute the similarity (or matching probability) between nodes and four methods to select query nodes.

Similarity computation: The implemented algorithms to compute the similarity (or matching probability) between nodes.

- *EM*: This is our algorithm to calculate the matching probability by using the EM algorithm presented in Section 5.3.
- *SR*: It represents an extended SimRank algorithm [15] modified for computing the similarity between two nodes from a pair of different graphs.
  We compute SimRank $sim^{SR}_t(u, v)$ for every pair of $u \in V^L_t$ and $v \in V^R_t$ in a merged graph of $\mathbb{G}^L$ and $\mathbb{G}^R$. In other words, $sim^{SR}_t(u, v)$ is defined recursively as
  $$sim^{SR}_t(u, v) = \frac{c}{|T|} \sum_{s\in T} \sum_{x\in N^L_{t,s}(u)} \sum_{y\in N^R_{t,s}(v)} \frac{sim^{SR}_s(x, y)}{|N^L_{t,s}(u)||N^R_{t,s}(v)|}$$
  Note that $sim^{SR}_s(x, y) = 1$ if $x \in V^L_s$ and $y \in V^R_s$ were already matched together. The damping factor c is set to 0.8.
- *RW*: We extended the graph matching algorithm in [16] to heterogeneous graphs. The matching probability $sim^{RW}_t(u, v)$ for a pair of nodes (u, v) of type t between two graphs is estimated by using a random walk model, where $sim^{RW}_t(u, v)$ is defined recursively as
  $$sim^{RW}_t(u, v) = \sum_{s\in T} \sum_{(x,y)\in N^L_{t,s}(u)\times N^R_{t,s}(v)} sim^{RW}_s(x, y) \frac{\bar{w}_{t,u\to v}}{\sum_{t'\in T} \sum_{(u',v')\in N^L_{s,t'}(x)\times N^L_{s,t'}(y)} \bar{w}_{t',u'\to v'}}$$
  where $\bar{w}_{t,u\to v}$ is the observed matching probabilities defined in Section 3.
- *UA*: This indicates the graph alignment algorithm, called *UniAlign*, introduced in [19]. It constructs the feature matrices $L \in \mathbb{R}^{|V_L|\times d}$ and $R \in \mathbb{R}^{|V_R|\times d}$ for the left and right graphs, respectively. The i-th row of each matrix consists of d dimensional feature vector of the i-th node. As an example, the feature vector can contain the i-th node's degree and the mean degree of its neighbors. Then, it computes the permutation matrix P to minimize the mismatches between the two matrices L and R. To utilize the already-matched pairs obtained from crowdsourcing, we extended the algorithm by adding the number of annotations and the mean number of annotations of the neighboring nodes to the feature vectors.
- *AL*: This is the anchor link prediction algorithm developed in [18] that extracts the topological features as well as node attributes for each user and discovers the identical users between different social networks by using a support vector

	EMC	MAXENT	MINVAR	RAND
EM	EMC+EM	MAXENT+EM	-	RAND+EM
SR	-	-	MINVAR+SR	RAND+SR
RW	-	-	MINVAR+RW	RAND+RW
UA	-	-	MINVAR+UA	RAND+UA
AL	-	-	MINVAR+AL	RAND+AL
MP	-	-	MINVAR+MP	RAND+MP

Fig. 11. Implemented algorithms.

machine. We utilized only the topological features, such as similarities of neighbors, among those suggested in [18] because the other features (i.e., messages, the posting times and locations) are neither used in our model nor available in the datasets.

- *MP*: This denotes the network alignment algorithm *NetAlignMP* in [6] which matches two graphs based on the belief propagation. For our experiments, we set the *damping type* to 1. Furthermore, for an input candidate matching pair required by *NetAlignMP*, we used a node pair ($u \in V^L$, $v \in V^R$) such that u and v share an already-matched pair among the neighboring nodes within two hops.

Query selection: We implemented the following query selection algorithms.

- *EMC*: This is our algorithm to select a query node using the expected model change proposed in Section 6.
- *MAXENT*: This is an uncertainty sampling algorithm that selects a node from V_t^L whose entropy of the matching probability distribution is the largest (i.e., we select a node $u^* = \arg\max_{u \in V_t^L} \sum_{v \in V_t^L} -w_{t,u \to v} \log w_{t,u \to v}$).
- *MINVAR*: This denotes a method to find the node in V_t^L whose variance of the similarities compared to the nodes in V_t^R is the smallest.
- *RAND*: This is a simple algorithm that randomly chooses a query node from $\mathbb{G}^L$.

In our experiments, we evaluate the performance with 13 combinations of the above algorithms listed in Fig. 11.

8.2. Data sets and quality evaluation

In our experiments, we evaluate the performance of the active graph matching algorithms by simulating the answers that the annotators in a crowdsourcing platform would give based on both real-life and synthetic graph data sets. We also assess the efficiency of our algorithm by collecting the labels from human annotators in our own crowdsourcing platform using two real-life graphs downloaded from difference sources.

Graphs for synthetic data sets: We use three graphs to produce three synthetic data sets. For each of the following graphs, we generated two copies of the original graph and randomly deleted 5% of the edges in each copy for a more realistic situation of two graphs that match each other with some discrepancies.

- *IMDB*[1]: It includes 16,296 nodes consisting of 5997 movie titles, 4902 actors/actresses and 5397 directors. The average number of neighboring actor nodes for movies (i.e., $|N_{movie,\ actor}|$) is 2.40. The averages of $|N_{actor,\ movie}|$, $|N_{movie,\ director}|$ and $|N_{director,\ movie}|$ are 1.22, 1.39 and 1.25, respectively. There is no edge between directors and actors.
- *ARXIV*[2]: This is a collaboration network of scientific literatures from ARXIV, collected and used for experiments in [22]. Each of 5242 nodes in this network represents an author. An edge exists between two nodes if the two authors have published a paper together. In this data set, one author collaborated with an average of 5.53 authors (i.e., $|N_{author,author}| = 5.53$).
- *DBPEDIA*[3]: The data set contains 581 movies, 2005 actors and 436 directors collected from DBpedia. The averages of $|N_{movie,\ actor}|$ and $|N_{actor,\ movie}|$ are 7.32 and 2.14, respectively. The average degree of movie to director and director to movie are 2.12 and 1.50, respectively.

Graphs for real-life data set: We collected two movie graphs from different sources to evaluate the performance of the graph integration algorithms with more dissimilar graphs.

- *MOVL*: We reused DBPEDIA for the left graph $\mathbb{G}^L$ from which we select query nodes to be suggested to annotators.

[1] We downloaded it from http://www.imdb.com/.
[2] The dataset is publicly available at http://snap.stanford.edu/data/ca-GrQc.html.
[3] It is accessible at https://dbpedia.org/sparql.

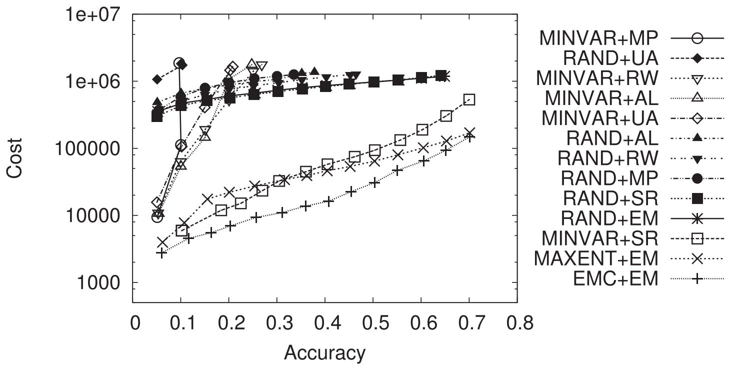


Fig. 12. Experiments with all implemented algorithms.

- *MOVR*: It includes 20,868 nodes consisting of 581 movie titles, 19,811 actors/actresses and 476 directors newly collected from IMDB. The average number of neighboring actor nodes for movies (i.e., $|N_{movie,\ actor}|$) is 47.04. The averages of $|N_{actor,\ movie}|$, $|N_{movie,\ director}|$ and $|N_{director,\ movie}|$ are 1.38, 1.15 and 1.40, respectively. Note that this graph contains approximately ten times as many actor/actress nodes as MOVL. We utilized MOVR as the right graph $\mathbb{G}^R$ from which we will select the candidate nodes for each query node.

In addition, we randomly selected a small number of initially already-known matches. For the synthetically generated graph data sets, we gave only a single already-known match and for the real-life data sets using IMDB and DBpedia, we set 10 already-known matches for the first query selection.

Quality measures: At each round of the iteration, we evaluate the quality of graph matching in terms of two measures: the *accuracy* of matching and the *accumulated cost* for manual labeling. The *accuracy* of matching is the ratio of the number of correctly matched nodes in the left graph over the total number of nodes in the left graph. Assuming that annotators check the candidates sequentially one by one to find the match for the query node, the *accumulated cost* is computed by summing the number of candidates that annotators must examine until they find the matching node to answer the question. The cost of querying against the achieved accuracy has been widely used for evaluating the efficiency of active learning and is similar to the *latency* which was generally used to measure the performance of crowdsourcing systems [36].

8.3. Graph matching of synthetic data

We first evaluate the efficiency of active graph matching algorithms by simulating what the annotators would answer without using actual crowdsourcing systems as existing commercial crowdsourcing systems such as Amazon Mechanical Turk [1] do not allow users to utilize their own user-defined rules to select a query or candidates. Note that, in the next section, we also provide the result of actual crowdsourcing experiments performed on our own crowdsourcing platform built for integrating two graphs from different sources.

Setting to simulate crowdsourcing: For two graphs obtained from different sources, we generated two copies from the IMDB graph, the ARXIV graph, DBPEDIA graph and the synthetic graphs, and randomly deleted 5% of the edges in each copy for the sake of realism. We used those graphs as the left and right graphs, respectively. Due to the method of producing two graphs for our experiments, we can use the correct matching pair of every node in the left graph to evaluate the accuracy of matching.

To evaluate the performance of active graph matching algorithms, we first select 100 query nodes (i.e., $k_q = 100$) as well as 100 candidate nodes for each query node (i.e., $k_c = 100$), and obtain the correct answers among the candidates for every query node by crowdsourcing. To consider the possibility that annotators return incorrect labels, we occasionally match a query node by randomly selecting an incorrect node from among(the candidates. We next update the similarities based on the obtained matching pairs and perform the automated graph matching at the moment. We call such steps a *round* and repeat multiple rounds until we obtain sufficient accuracy of matching.

Default parameters: We conducted our experiments by varying the regularization constant μ in our model and the probability ϵ that annotators make an error in matching nodes. The default values of these parameters were: $\mu = 10$ and $\epsilon = 0.03$. The default value of μ is chosen empirically according to the experiment and will be further discussed later. For all performance evaluations, we repeat up to 200 rounds until the accuracy reaches 0.7.

Performance evaluation with all implemented algorithms: We first plotted the accumulated costs for all implemented algorithms at each round using the IMDB data set, as shown in Fig. 12. Each plot represents the quality measured at each round where the *x*-axis is the accuracy and the *y*-axis is the accumulated cost in log scale. The graph shows that our proposed *EMC+EM* is the best performer and *MAXENT+EM* is the second best. The accumulated cost of every extended graph alignment algorithms with *RW, AL, UA* and *MP* exceeds one million before their accuracy reaches 0.5. This is because all those traditional algorithms require the provision of many matched pairs in advance for matching node predictions. Moreover, *UA*

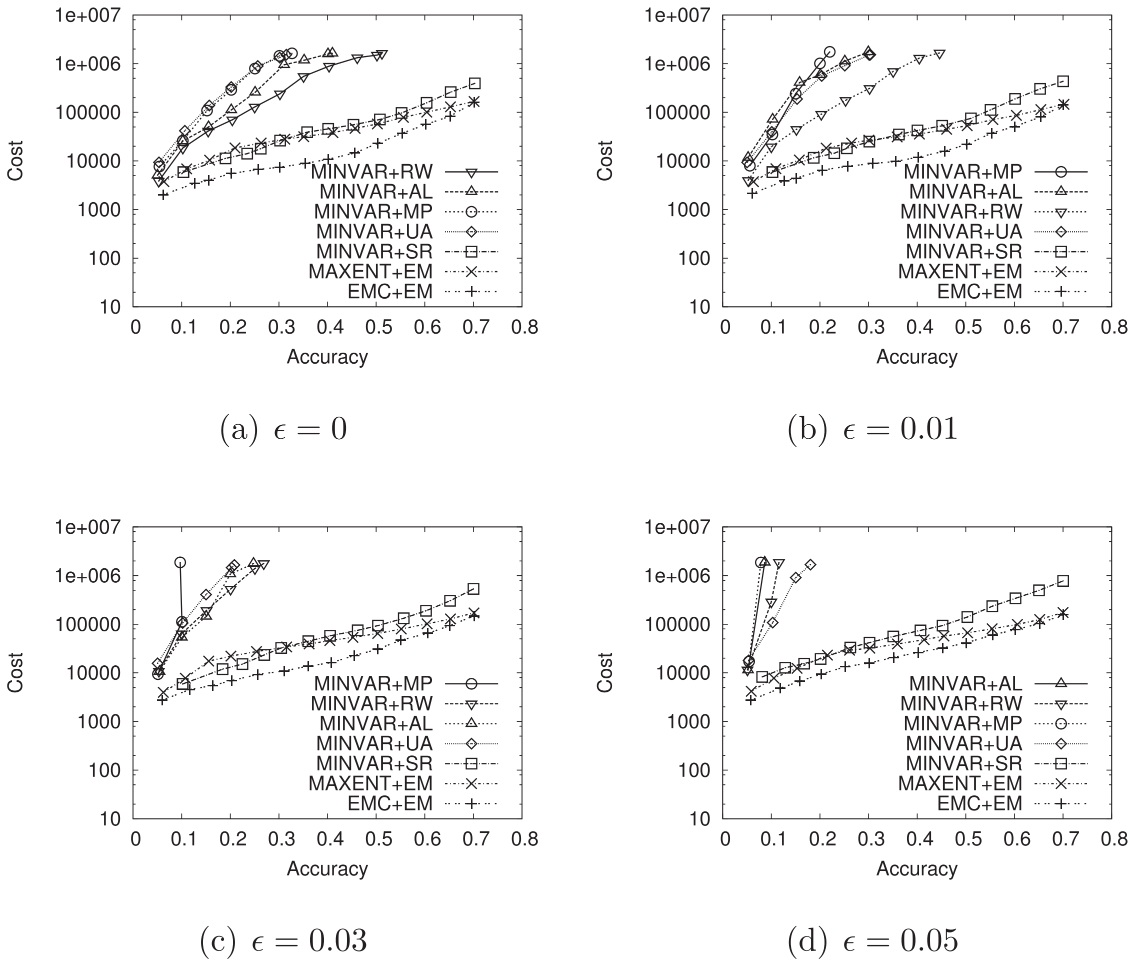


Fig. 13. Accumulated cost in each round with IMDB data set.

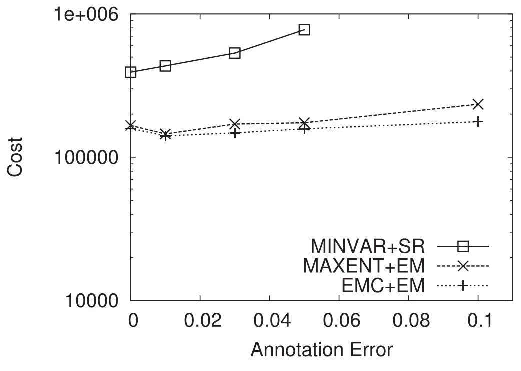
align graphs without the consideration of using already known matching pairs. It confirms that our proposed active learning algorithms improve the accuracy as we collect the matching pairs of nodes by repeatedly asking questions. Because all the algorithms combined with the random selection of query nodes (i.e., *RAND*) perform worse than those with the other query selection strategies, we will not show the performance result of the algorithms using *RAND* in the rest of the paper.

Varying ϵ: In Fig. 13(a)–(d), we presented the accumulated cost at each round for the seven implemented algorithms using the IMDB data set. Each figure shows the performance with increasing ϵ from 0 to 0.05. All graphs in Fig. 13 show that our proposed algorithm *EMC+EM* outperforms the other algorithms with every value of ϵ. When $\epsilon = 0.05$, *EMC+EM* saves 4.92 times as much cost compared to *MINVAR+SR* for matching graphs with an accuracy of 0.7. As ϵ increases, we can observe the performances of the algorithms using *RW, AL, UA* and *MP* decline faster than those using *EM*. This shows that *EM* can correctly estimate the matching probabilities although there are some incorrect answers.

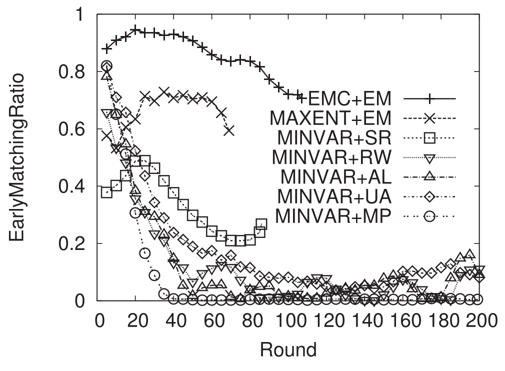
In Fig. 14(a), we plotted the accumulated cost of obtaining accuracy of 0.7 with increasing ϵ from 0 to 0.1. Because the algorithms using *RW, AL, UA* and *MP* never achieve the accuracy of 0.7, they are not shown in the graph. We found that the performances of *EMC+EM* and *MAXENT+EM* degrade little with increasing ϵ while the costs of the other algorithms increase significantly. The result once again confirms that our algorithm properly handles the possibility that annotators may give incorrect answers.

Ratio of the early-matched nodes: We presented the ratio of the early-matched nodes and all query nodes (i.e., 100 query nodes) in Fig. 14(b) where the early-matched node is a query node that receives an answer within the top-10 candidates. The graph shows that the ratio of the early-matched nodes for our algorithm *EMC+EM* is always larger than 0.7 in every round and it confirms that *EMC* desirably selects the candidate nodes containing the matching nodes from the right graph.

Varying μ: We traced the accumulated costs of *EMC+EM* with varying μ but did not present the graph in this paper due to space constraints. With a small μ, our inference algorithm tends to focus on computing the matching probabilities close to those estimated by the random model (i.e., Eq. (1) in Section 5) ignoring the observed matching probabilities obtained from the labels. Thus, it requires more questions to ask annotators. Empirically, when μ is larger than 10, the different per-

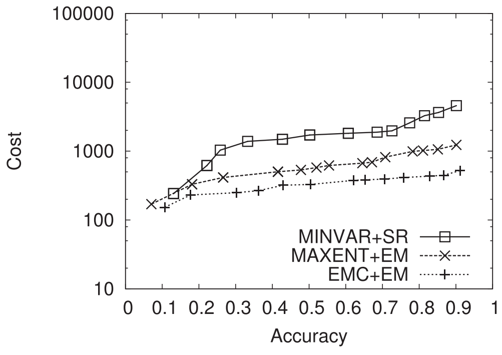


(a) Cost with varying ϵ

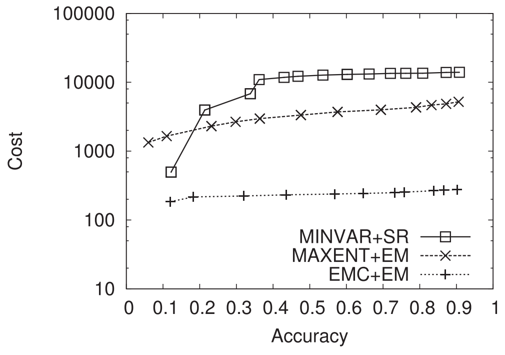


(b) Ratio of early matched pairs

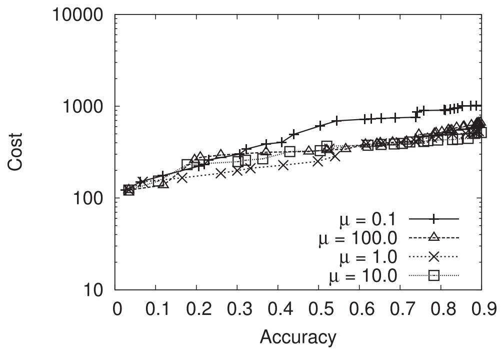
Fig. 14. Experiments with IMDB data set.



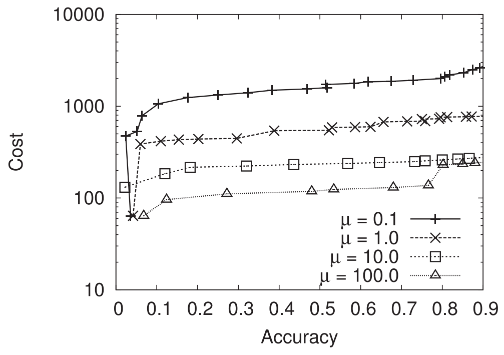
(a) Accumulated cost ($p = 0.01$)



(b) Accumulated cost ($p = 0.03$)



(c) Varying μ ($p = 0.01$)



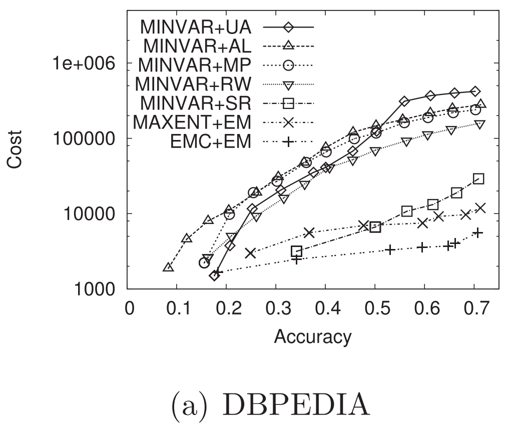
(d) Varying μ ($p = 0.03$)

Fig. 15. Varying the connectivity of synthetic graphs generated with $p = 0.01$ and 0.03.

formances of our algorithm were very similar to each other and thus, we set the default value of μ to 10 in our experiments with IMDB data set.

Varying the connectivity of graphs: To experiment with varying the connectivity of graphs, we generated two other synthetic data sets with $p = 0.01$ and 0.03. We next ran all implemented algorithms with the synthetic data sets. We show the results of *MINVAR+SR, MAXENT+EM* and *EMC+EM* only because they always outperform the other algorithms.

We plotted the accuracy with growing accumulated cost in Fig. 15(a) and (b). In each round, we repeatedly asked 5 queries until we achieved the accuracy of 0.9. The result confirms that our proposed algorithm *EMC+EM* outperforms the



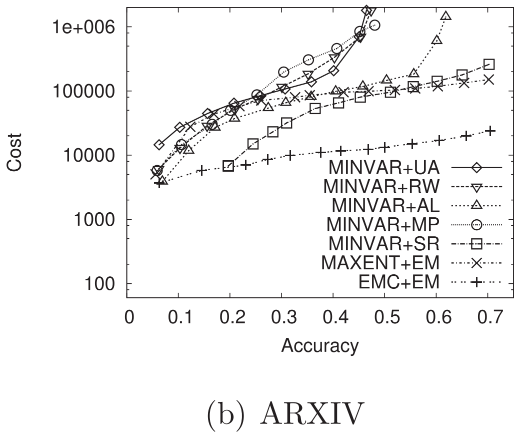


Fig. 16. Accumulated costs with DBPEDIA and ARXIV.

	EMC+EM	MAXENT+EM	RAND+EM	MINVAR+SR	RAND+SR
T1	5,136	2,799	4,846	101	190
T2	162.8	0.5	2.4	4.5	2.3
AC	147,924	170,410	1,186,215	533,190	1,222,984

Fig. 17. Execution time (s) and accumulated cost.

other algorithms for all synthetic data sets with different connectivities. Furthermore, we found that *EMC+EM* can integrate the given graphs more precisely with a smaller number of questions when the graphs have large connectivity. The reason is that the more edges they have, the more clues we have for discovering the matching nodes.

We next provide the accumulated cost of *EMC+EM* with varying μ and p together in Fig. 15(c) and (d). As the connectivity of graphs grows, we can see that computing matching probability with large values of μ shows better performance. For sparsely connected graphs, however, a large value of μ is not always desirable because the incorrect answers do not have enough chances to be fixed by neighboring nodes.

Performance evaluations with DBPEDIA and ARXIV: We presented the performance of the graph integration algorithms with DBPEDIA and ARXIV in Fig. 16. The trends are similar to those graphs in Fig. 12 where *EMC+EM* is the best performer and *MAXENT+EM* is the second best one. In Fig. 16(a), *EMC+EM* 5.52 times better with regard to cost compared to *MINVAR+SR* for DBPEDIA. The performance gaps between *EMC+EM* and the other algorithms were bigger for ARXIV whose size is bigger than DBPEDIA. In Fig. 16(b), we can see that *MINVAR+SR* costs 10.83 times more than *EMC+EM* to achieve an accuracy of 0.7. Furthermore, the other algorithms based on *MINVAR* fail to reach the accuracy of 0.7. The results show that the proposed algorithm *EMC+EM* is more scalable with increasing graph sizes than the other algorithms.

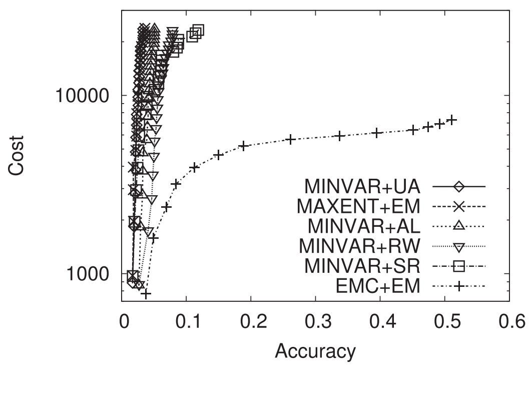
Execution time v.s. crowdsourcing cost: We next presented the execution time of similarity computation (T1), query and candidate selection (T2) with accumulated cost (AC) until we achieve the accuracy 0.7 in Fig. 17. Note that the actual time consumed by human annotators in crowdsourcing systems is not shown here. As we can expect, the time needed for query selection by *EMC* is larger than that needed by *MAXENT, MINVAR* and *RAND* while *EMC* reduces the crowdsourcing cost by at least 15%. Considering that it takes much longer for annotators to answer the questions in crowdsourcing systems than the computation time, such computation overheads are very affordable.

8.4. Graph matching of real-life data

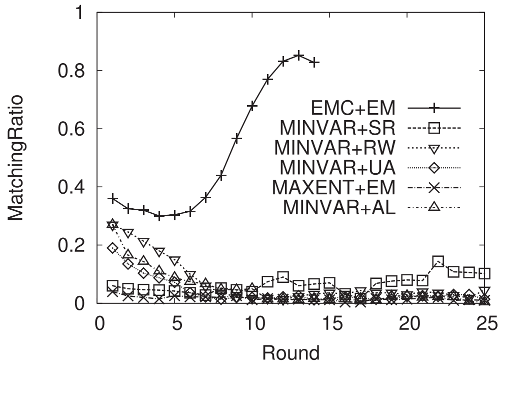
We evaluated the performance of active graph matching using *IMDB* and *DBpedia* collected from different data sources. Because we do not have the ground truth of the matching pairs between those graphs, for every node in the small data set *DBpedia*, we downloaded its matching node in *IMDB* by asking 5 human annotators and used *DBpedia* as the left graph from which we choose the query nodes for active graph matching.

We first performed active matching with human annotators on our own crowdsourcing system implemented for graph matching. We selected 100 query nodes at each round (i.e., $k_q = 100$) and suggested 10 candidate nodes to an annotator for each query node (i.e., $k_c = 10$). Moreover, we also did active matching of those two graphs with simulated answers, in the previous section. In the experiments, we show the performance of all implemented algorithms except those with *RAND*.

Active matching with human annotators: We plotted the accumulated cost and the accuracy of 14 rounds for *EMC+EM* and 25 rounds for *MINVAR+SR* and *MAXENT+EM* in Fig. 18(a). We stopped crowdsourcing for *MINVAR+SR* and *MAXENT+EM* at the 25-th round. The graph shows that *EMC+EM* achieves an accuracy of 0.5 at the 14-th rounds with a cost of 7277 while *MINVAR+SR* has an accuracy of 0.1 at the 22-nd rounds with a cost of 21,372. We also presented the ratio of early-matched query nodes in Fig. 18(b). Recall that it presents the ratio of the queries to be answered only with the top-10 candidates. The

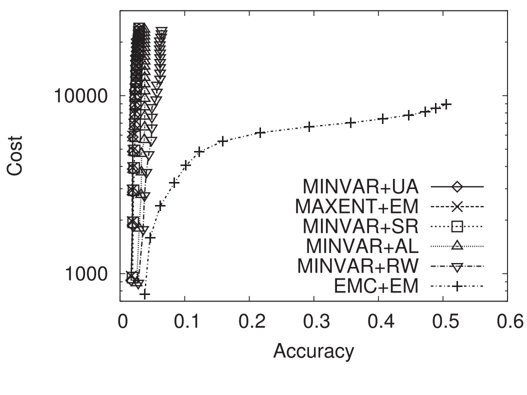


(a) Accumulated cost

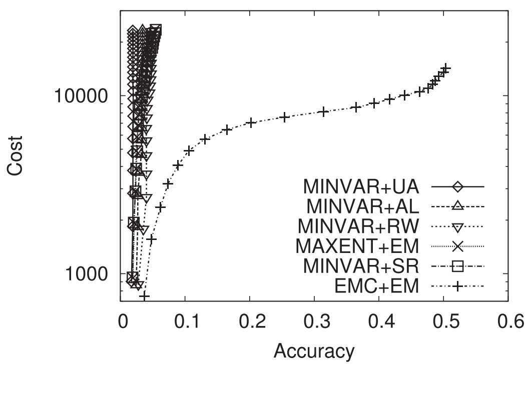


(b) Matching ratio

Fig. 18. Active matching with human annotators.



(a) $\epsilon = 0.01$



(b) $\epsilon = 0.05$

Fig. 19. Accumulated cost with DBpedia.

result shows that the ranking of candidates is not precise in the early rounds for every algorithm, but *EMC+EM* gradually improves the accuracy round by round.

Simulated active matching: We plotted the accumulated cost at each round with the probability of choosing wrong answers $\epsilon = 0.01$ and 0.05 in Fig. 19. The graphs show that *EMC+EM* definitely outperforms the other algorithms with every value of ϵ. We found that the performance of *EMC+EM* is much better than the other algorithms compared to the other experiments with *IMDB* in Fig. 13. The reason is that the right graph *IMDB* is relatively much larger than the left graph *DBpedia*, and we must select the candidates from a larger set of nodes in the right graph. The result confirms that *EMC* chooses a query that is likely to give a high rank to the correct answer, and achieves higher accuracy than *MAXENT* and *MINVAR*, which simply select the query nodes with the greatest uncertainty.

9. Conclusion

In this paper, we proposed an active graph matching algorithm that efficiently exploits a crowdsourcing system. We proposed a probabilistic random walk model to compute the matching probability between every two nodes each from different graphs by modeling the behavior of annotators to detect the matching node of a given query node. We developed an efficient inference algorithm to calculate the matching probability based on the Expectation-Maximization (EM) algorithm. To consider the possibility that annotators may give incorrect labels, our model and inference algorithm borrowed from the approach of semi-supervised learning which does not definitely fix the matching probability on the observed label. For efficient active learning, we also devised a method to select the most informative query node, which maximizes the proposed expected model change. Experimental results with both synthetic and real-life data sets confirmed that our proposed algorithm $EMC + EM$ discovers matching nodes most accurately with the lowest cost compared to the baseline algorithms.

Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2012M3C4 A7033342). This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-H8501-16-1013) supervised by the IITP (Institute for Information & communication Technology Promotion). This work was also supported by Korea Institute of Energy Technology Evaluation and Planning (KETEP) granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea (No. 20153010011980).

References

[1] Amazon Mechanical Turk, 2014, http://www.mturk.com/.
[2] DBpedia Knowledge Base, 2015, http://wiki.dbpedia.org.
[3] Internet Movie Database, 2015, http://imdb.com.
[4] N. Agarwal, M.G. Oliveras, H. Liu, S.B. Subramanya, Wiscoll: collective wisdom based blog clustering, Inf. Sci. 180 (1) (2010) 39–61.
[5] M. Bayati, M. Gerritsen, D. Gleich, A. Saberi, Y. Wang, Algorithms for large, sparse network alignment problems, in: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, 2009, pp. 705–710.
[6] M. Bayati, D.F. Gleich, A. Saberi, Y. Wang, Message-passing algorithms for sparse network alignment, ACM Trans. Knowl. Discov. Data 7 (1) (2013) 3.
[7] A. Bhattacharyya, On a measure of divergence between two statistical populations defined by their probability distributions, Bull. Calcutta Math. Soc. 35 (1943) 99–109.
[8] P. Buneman, S.B. Davidson, A. Kosky, Theoretical aspects of schema merging, in: Proceedings of the International Conference on Extending Database Technology, 1992, pp. 152–167.
[9] E. Choi, C. Lee, Feature extraction based on the bhattacharyya distance, Pattern Recognit. 36 (8) (2003) 1703–1709.
[10] G. Demartini, D.E. Difallah, P. Cudré-Mauroux, Large-scale linked data integration using probabilistic reasoning and crowdsourcing, VLDB J. 22 (5) (2013) 665–687.
[11] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. R. Stat. Soc. Ser. B 39 (1977) 1–38.
[12] A. Feng, M.J. Franklin, D. Kossmann, T. Kraska, S. Madden, S. Ramesh, A. Wang, R. Xin, CrowdDB: query processing with the VLDB crowd, Proc. VLDB Endow. 4 (12) (2011) 1387–1390.
[13] J. Gemmell, B.I.P. Rubinstein, A.K. Chandra, Improving Entity Resolution with Global Constraints, arXiv preprint 1108.6016 (2011).
[14] L. Getoor, A. Machanavajjhala, Entity resolution: theory, practice & open challenges, Proc. VLDB Endow. 5 (12) (2012) 2018–2019.
[15] G. Jeh, J. Widom, Simrank: a measure of structural-context similarity, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 538–543.
[16] H. Jeong, B. Yoon, Effective estimation of node-to-node correspondence between different graphs, IEEE Signal Process. Lett. 22 (6) (2015) 661–665.
[17] Y. Kim, W. Kim, K. Shim, Latent ranking analysis using pairwise comparisons, in: Proceedings of the 2014 IEEE International Conference on Data Mining, 2014, pp. 869–874.
[18] X. Kong, J. Zhang, P.S. Yu, Inferring anchor links across multiple heterogeneous social networks, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM, 2013, pp. 179–188.
[19] D. Koutra, H. Tong, D. Lubensky, BIG-ALIGN: fast bipartite graph alignment, in: Proceedings of the 2013 IEEE 13th International Conference on Data Mining, 2013, pp. 389–398.
[20] H.W. Kuhn, B. Yaw, The Hungarian method for the assignment problem, Naval Res. Logist. Q. (1955) 83–97.
[21] N. Lao, W.W. Cohen, Fast query execution for retrieval models based on path-constrained random walks, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 881–888.
[22] J. Leskovec, J.M. Kleinberg, C. Faloutsos, Graph evolution: densification and shrinking diameters, ACM Trans. Knowl. Discov. Data 1 (1) (2007).
[23] D.D. Lewis, W.A. Gale, A sequential algorithm for training text classifiers, in: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1994, pp. 3–12.
[24] J. Liu, X.X. Zhang, Data integration in fuzzy XML documents, Inf. Sci. 280 (2014) 82–97.
[25] S. Liu, S. Wang, F. Zhu, J. Zhang, R. Krishnan, HYDRA: large-scale social identity linkage via heterogeneous behavior modeling, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 51–62.
[26] M.S. Mahmoud, H.M. Khalid, Expectation maximization approach to data-based fault diagnostics, Inf. Sci. 235 (2013) 80–96.
[27] A. Marcus, E. Wu, D.R. Karger, S. Madden, R.C. Miller, Demonstration of qurk: a query processor for humanoperators, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, 2011, pp. 1315–1318.
[28] L. Nie, X. Song, T. Chua, Learning from Multiple Social Networks, Morgan & Claypool Publishers, 2016.
[29] E. Rahm, P.A. Bernstein, A survey of approaches to automatic schema matching, VLDB J. 10 (4) (2001) 334–350.
[30] B. Settles, M. Craven, An analysis of active learning strategies for sequence labeling tasks, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2008, pp. 1070–1079.
[31] B. Settles, M. Craven, S. Ray, Multiple-instance active learning, in: Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, 2008, pp. 1289–1296.
[32] R. Singh, J. Xu, B. Berger, Pairwise global alignment of protein interaction networks by matching neighborhood topology, in: Proceedings of the Annual International Conference on Research in Computational Molecular Biology, 2007, pp. 16–31.
[33] X. Song, Z. Ming, L. Nie, Y. Zhao, T. Chua, Volunteerism tendency prediction via harvesting multiple social networks, ACM Trans. Inf. Syst. 34 (2) (2016) 10.
[34] X. Song, L. Nie, L. Zhang, M. Liu, T. Chua, Interest inference via structure-constrained multi-source multi-task learning, in: Proceedings of the Twenty–Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, 2015, pp. 2371–2377.
[35] N. Vesdapunt, K. Bellare, N.N. Dalvi, Crowdsourcing algorithms for entity resolution, Proc. VLDB Endow. 7 (12) (2014) 1071–1082.
[36] J. Wang, T. Kraska, M.J. Franklin, J. Feng, CrowdER: crowdsourcing entity resolution, Proc. VLDB Endow. 5 (11) (2012) 1483–1494.
[37] J. Wang, G. Li, T. Kraska, M.J. Franklin, J. Feng, Leveraging transitive relations for crowdsourced joins, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, 2013, pp. 229–240.
[38] J. Wang, G. Li, T. Kraska, M.J. Franklin, J. Feng, The Expected Optimal Labeling Order Problem for Crowdsourced Joins and Entity Resolution, (2014). ArXiv preprint arXiv:1409.7472.
[39] J.T. Wang, K. Zhang, G. Chirn, Algorithms for approximate graph matching, Inf. Sci. 82 (1–2) (1995) 45–74.
[40] M. Wang, X. Hua, Active learning in multimedia annotation and retrieval: a survey, ACM Trans. Intell. Syst. Technol. 2 (2) (2011) 10.
[41] M. Wang, X. Hua, R. Hong, J. Tang, G. Qi, Y. Song, Unified video annotation via multigraph learning, IEEE Trans. Circuits Syst. Video Technol. 19 (5) (2009) 733–746.
[42] M. Wang, H. Li, D. Tao, K. Lu, X. Wu, Multimodal graph-based reranking for web image search, IEEE Trans. Image Process. 21 (11) (2012) 4649–4661.
[43] T. Yan, V. Kumar, D. Ganesan, Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, 2010, pp. 77–90.

[44] J. Yu, Y. Rui, Y.Y. Tang, D. Tao, High-order distance-based multiview stochastic learning in image classification, IEEE Trans. Cybern. 44 (12) (2014a) 2431–2442.
[45] J. Yu, D. Tao, J. Li, J. Cheng, Semantic preserving distance metric learning and applications, Inf. Sci. 281 (2014b) 674–686.
[46] J. Yu, X. Yang, F. Gao, D. Tao, Deep multimodal distance metric learning using click constraints for image ranking, IEEE Trans. Cybern. 99 (2016) 1–11.
[47] R. Zafarani, H. Liu, Connecting users across social media sites: a behavioral-modeling approach, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 41–49.
[48] Y. Zhang, J. Tang, Z. Yang, J. Pei, P.S. Yu, COSNET: connecting heterogeneous social networks with local and global consistency, in: Proceedings of International Conference on Knowledge Discovery and Data Mining, SIGKDD, 2015, pp. 1485–1494.
[49] X. Zhu, Semi-supervised Learning Literature Survey, Computer Sciences, University of Wisconsin-Madison, 2005 Technical report 1530.