World Wide Web (2014) 17:1051–1079
DOI 10.1007/s11280-013-0229-1

Integration of scientific and social networks

Mahmood Neshati · Djoerd Hiemstra · Ehsaneddin Asgari · Hamid Beigy

Received: 31 August 2012 / Revised: 7 April 2013 /
Accepted: 3 June 2013 / Published online: 21 June 2013

Abstract In this paper, we address the problem of scientific-social network integration to find a matching relationship between members of these networks (i.e. The DBLP publication network and the Twitter social network). This task is a crucial step toward building a multi environment expert finding system that has recently attracted much attention in Information Retrieval community. In this paper, the problem of social and scientific network integration is divided into two sub problems. The first problem concerns finding those profiles in one network, which presumably have a corresponding profile in the other network and the second problem concerns the name disambiguation to find true matching profiles among some candidate profiles for matching. Utilizing several name similarity patterns and contextual properties of these networks, we design a focused crawler to find high probable matching pairs, then the problem of name disambiguation is reduced to predict the label of each candidate pair as either true or false matching. Because the labels of these candidate pairs are not independent, state-of-the-art classification methods such as logistic regression and decision tree, which classify each instance separately, are unsuitable for this task. By defining matching dependency graph, we propose a joint

M. Neshati (✉) · H. Beigy
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: mahmood.neshati@gmail.com

H. Beigy
e-mail: beigy@sharif.edu

D. Hiemstra
Database Research Group, Electrical Engineering, Mathematics and Computer Science (EEMCS) Department, University of Twente, Enschede, The Netherlands
e-mail: d.hiemstra@utwente.nl

E. Asgari
School of Computer and Communication Science (IC), Ecole Polytechnique Fédérale de Lausanne - EPFL, Lausanne, Switzerland
e-mail: ehsaneddin.asgari@epfl.ch

label prediction model to determine the label of all candidate pairs simultaneously. Two main types of dependencies among candidate pairs are considered for designing the joint label prediction model which are quite intuitive and general. Using the discriminative approaches, we utilize various feature sets to train our proposed classifiers. An extensive set of experiments have been conducted on six test collection collected from the DBLP and the Twitter networks to show the effectiveness of the proposed joint label prediction model.

Keywords Social network integration · Twitter · DBLP · Collective classification

1 Introduction

As the large portion of the web provides information for various kinds of real-world objects (i.e. entities), more search engines provide object level search result. Typical objects are products, people, papers, organizations, and the like. If these objects and their attributes can be extracted from the web, powerful object-level search engines can be built to more precisely meet users' information needs. Well-known examples of object level search engines include scientific expert search [29], book search [17] and product search [24].

People search is one of the most interesting and challenging types of object level search. In the information retrieval community, people search is also known as expert search (i.e. expert finding). Expert finding addresses the problem of identifying individuals who are knowledgeable in a given topic. State-of-the-art algorithms for expert finding rank persons based on the content of their associated documents and relations. Although most of the proposed algorithms for expert finding restrict their analysis to the documents and relations exist in a single environment [2, 21, 25], recent studies [12, 26] suggest that analysis of personal expertise should not be necessarily undertaken only using the date of one single environment. For example, while [2, 21, 25], simply use the information collected from the interant of an organization to rank people, recent approaches [12, 26] also consider information extracted from web pages (i.e. homepage) and social networks (i.e. personal weblogs) to rank them. In fact, besides the degree of expertise, there are some other important factors, which should be taken into account for ranking of experts. These factors such as contextual factors [16], the availability of an expert [27] and the authority of experts in their specialization area [9] are generally independent of the content of the documents and can be extracted from multiple sources of information.

Recently, user generated data is growing rapidly and becoming one of the most important sources of information on the web. This data contains a lot of information such as opinion, experience, etc., which can be useful for expert finding. Microblogs are one of the such valuable and reliable sources of information since they usually contain up-to-date and relatively well-formatted data as well as meaningful relationships between people. Expert's microblogs can be used to estimate the effective factors for ranking (e.g. temporal, geographical and contextual factors) and this makes automatic discovery of expert's microblogs an important step toward building a multi environment expert finding system.

In this paper, we address the problem of scientific-social network integration towards building such multi environment expert finding system. We propose an

automatic method to integrate two networks of experts. One of them is an official academic network (i.e. the DBLP[1] publication database) that indexes the name of experts and their publications, and the other one is the network (i.e. the Twitter[2] social network) of expert's microblogs.

There are some services, which can map people's name with its social entity (e.g. its Twitter profile) [31]. Those services are built on an assumption that full names uniquely identify social entities, and therefore, they focus on textual name matching. However, this assumption is not valid for many cases. In general, integration of scientific and social networks is a challenging task because of the following reasons:

- **Nicknaming**: According to a recent research study [33], about 11 % of people use nicknames in microblogs, which cannot be reached by the naive name matching. In these cases, slightly different names in multiple networks refer to the same person. This problem is known as the identification problem in the name disambiguation literature [5].
- **Name ambiguity**: The second main challenge in social network integration is distinguishing those entities that have very similar and sometimes exactly the same name and yet refer to different people. This problem is known as the disambiguation problem in name disambiguation literature [5].
- **Multiple reference**: Although the majority of people have only one profile in the social and scientific networks, in some cases, more than one profile may exist for an individual in a network.
- **Local access to profiles**: In many cases, it is impossible to access the whole profiles in a network simultaneously, therefore we need a crawling method to access the profiles in a network.

We design a focused crawler to collect high probable matching profile pairs in scientific and social networks. Using a bootstrapping method, the crawler starts to collect people' profiles form common profiles of the two networks and in each step, it collects those social profiles[3] which follow (directly or indirectly) these common profiles (i.e. seed profiles). Using several name matching patterns, each social profile can be potentially matched to a limited number of corresponding profiles in the scientific network. The social-scientific network integration problem is then reduced to finding true matching pairs among the collected candidate pairs.

Using discriminative features extracted from the candidate profiles of matching, we can utilize state-of-the-art classification methods (e.g. logistic regression, SVM, decision tree, etc.) to classify candidate pairs and find true matching profiles in the two networks. These methods basically assume that the label of each instance is independent of other instances and do not use the relations between them. However, in scientific-social integration problem, the profiles in each network are related to each other, and the label (either true or false) of each matching candidate pair is not independent of the label of other pairs.

[1] http://www.informatik.uni-trier.de/~ley/db/

[2] https://twitter.com

[3] In this paper, each social profile is equivalent to a Twitter user page and each scientific profile is equivalent to a DBLP user page.

We consider two main types of dependencies between candidate pairs:

- **Common friend dependency:** People are related to each other in scientific and social networks. An individual can have co-author relationship with other people in a scientific network and also be a friend of others in social networks. In many cases, scientific collaborators are also social friends. So, if for a matching candidate pair, a *common friend* exists in both networks, it will be more likely to be a true match, but finding a common friend in both networks is not possible until we resolve all matching pairs. It means that we should jointly predict the label of all candidate pairs.
- **One-to-One matching dependency:** Scientific networks (e.g. digital libraries) use sophisticated algorithms [5, 13] and manual effort [20] to identify and disambiguate people with similar names. So, if one specific social profile is a candidate for matching with two or more scientific profiles, it is less likely to be a true match for more than one of them. On the other hand, the majority of people have at most one profile in a social network. Therefore, if a scientific profile is already determined as a true match for a specific social profile, the probability of matching other social profiles (for the same scientific profile) should be reduced.

To utilize the above-mentioned dependencies in network integration problem, we transform the initial view of each network as well as their relationships into a new graph structure called *Matching Dependency Graph*. In this graph, nodes indicate matching candidate pairs and edges represent their label dependencies. Following the idea of conditional random fields [19], we use a discriminative graphical model to determine the label of all candidate pairs simultaneously in order to integrate the scientific and social networks. To learn the parameters of the model, we optimize the conditional likelihood of the assigned labels for all candidate pairs given several discriminative features extracted from scientific and social profiles. Since it is impossible to use exact inference algorithms for the dependent variables in the matching dependency graph, approximate belief propagation is used to find the most probable label assignment for all candidate pairs simultaneously.

We apply our proposed algorithm to integrate DBLP and Twitter networks. To measure the performance of our matching algorithm, we build an automatically generated test collection and five manually annotated topical test collections. We conduct extensive sets of experiments to compare the performance of the proposed collective classification model with independent classification models as well as the algorithm proposed in [33].

The main contribution of this paper includes:

- Building a multi-environmental expert finding system to integrate the social and scientific activities of experts.
- Proposing a relational learning method that jointly integrates profiles of the two networks of people.
- Testing our algorithm on an automatic generated test collection. In addition, we build five test collections to test social-scientific network integration algorithms.

2 Related work

As a retrieval task, expert finding has attracted much attention, mostly due to the launching of the Enterprise track of TREC [3]. The previously proposed approaches

for expert finding obtain their estimator of personal expertise by aggregating the relevance scores of the documents related to a person [2, 21]. Most of these methods estimate the expertise score of each person according to the relations and documents existing in a single environment. For example, [2, 21] and [1] estimate the expertise score of each person based on his related documents collected from the intranet of an organization and the intranet of a university respectively. However, recently proposed models such as [27] and [9] consider heterogeneous sources of information to improve the quality of expert finding results. Smirnova and Balog [27] considered geographical location of experts to rank them based on their accessibility, and Deng et al. [9] suggested to rank each expert based on his authority in the research communities. The usefulness of heterogeneous sources of information for expert finding is also reported in [12] and [8]. Similar to the idea of heterogeneous information sources for expert finding, our goal is to build a multi environment (i.e. social and scientific) expert finding system.

In a similar approach to our approach, You et al. [33], proposed a method to integrate two networks of people namely, EntityCube [10] and Twitter networks. They addressed the problem of finding Twitter pages (i.e. social profile) of a group of related celebrities. Entity Cube is a virtual network of celebrities in which each node corresponds to the name of a celebrity, and each edge represents co-occurrence of names in web pages.

You et al. [33] used several name similarity patterns to find matching Twitter profile for each name in EntityCube. Using a couple of indicative features, they used a discriminative approach to rank Twitter candidate profiles for each name in EntityCube. They considered the *common friend* property (introduced in Section 1) to improve the accuracy of integration. However, they used independent learning approach to model this property. Specifically, for a candidate pair of matching nodes in Twitter and EntityCube, as indicated in Figure 1, they used the number of neighborhood nodes with similar names as a discriminative feature.

Some previous research [5, 30], reported significant accuracy improvement of relational learning methods (e.g. collective learning) in comparison with independent learning methods in interdependent decision making problems. Our proposed algorithm of matching, models the *common friend* property using relational learning method. The main benefit of our proposed relational learning model is its flexibility that can help us to consider various types of dependencies between candidate matching profiles (e.g. *one-to-one* matching property).

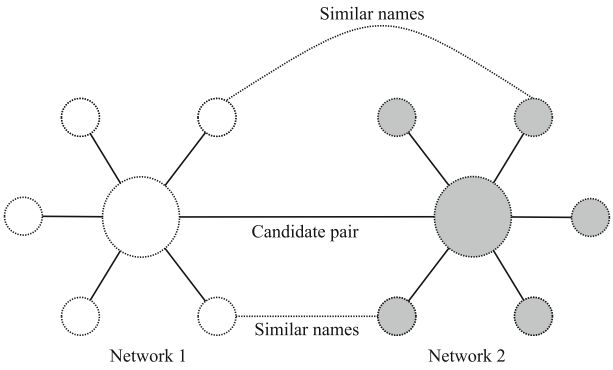

Figure 1 Modeling the *common friend* property in [33]

Record deduplication (also known as entity matching [18]) is a related research problem which concerns recognizing and discriminating ambiguous records/references in a database referring to a same underlying entity. These ambiguous references occur in databases due to the lack of unique identifier attributes. People name disambiguation is an important and well investigated type of such problems. Similar to this problem, our proposed algorithm can find matching profiles in social-scientific networks despite the lack of unique identifier attributes. Previously proposed methods for name disambiguation is generally based on learning a pair wise similarity function for ambiguous name/references. A couple of supervised [14, 23] and unsupervised [5, 15] methods are used to learn such similarity functions while each method uses a different set of textual and relational (i.e. co-occurrence based) features. For example, [5] and [6], used several features to measure co-occurrence and textual similarity of two ambiguous names/references respectively. Pursuing the same goal, specific properties of our problem impose some constraints on reference/name disambiguation, which is not considered in previous proposed methods for name disambiguation. Specifically, *one to one* matching property (introduced in Section 1) is a natural constraint in our problem, which is not considered in previous research.

Another related line of research is the problem of finding related web pages for a given query. It has attracted increasing attention within the IR research community since its first run as part of TREC-10 Web track. Craswell et al. [7] suggest the importance of anchor text in comparison with full-text search and Xi et al. [32] indicate the importance of query independent features in the homepage finding task. Recently, Fang et al. [11] proposed a discriminative graphical model to find the homepage of faculty members. Even though this problem is closely related to ours, homepage finding is more general in scope and does not consider the name disambiguation problem directly. Bekkerman and McCallum [4] proposed a more related task in which the aim is categorizing the web pages related to a set of ambiguous names. However, in their work, each person can have an arbitrary number of related webpages that contradicts the assumption of *one-to-one matching* in the network integration problem.

3 Integration of social-scientific networks

The goal of social-scientific network integration is to find a matching relationship between members of these networks. Membership of people in these networks happens for different reasons and usages. So, for a large number of members of a network, the corresponding profile may not exist in the other network. In other words, the matching relation between members of these two networks is neither surjective nor injective. On the other hand, due to name ambiguities, for people who have profiles in both networks, finding matching profiles is not a trivial task. The problem of social and scientific network integration can be divided into two sub problems. The first problem concerns finding those profiles in one network, which presumably have a corresponding profile in the other network and the second problem concerns the name disambiguation to find true matching profiles among some candidate profiles for matching. We refer to the first problem as the *selection* problem and the second as the *matching* problem.

Table 1 Twitter and DBLP common events

Topic of event	DBLP event	Twitter event
Information retrieval	SIGIR 2011	@sigir2011
	ECIR 2011	@ecir2011
	WSDM 2011	@wsdm2011
Programming languages	PLDI	@pldi
	OOPSLA 2009	@oopsla2009
Computer graphics	SIGGRAPH	@siggraph_ic
Operating systems	SOSP 2009	@sosp09
Databases	SIGMOD 2010	@sigmod2010
	SIGMOD 2011	@sigmod2011
Data mining	IJCAI 2011	@ijcai11
	KDD 2011	@kdd2011
HCI	CHI 2011	@chi2011
	UIST 2011	@uist2011

3.1 Selection problem

A simple solution to the selection problem is to search the social (or scientific) network with the name of all people in the other network and collect retrieved profiles as selected profiles for matching. However, this method has two main drawbacks. Firstly, in general, it is impossible to access the whole list of names in the social (or scientific) network. Secondly, this method only relies on exact name matching, and therefore, it is not able to collect profiles with slightly different names, that refer to the same person. To overcome these problems, we use a focused crawler to collect those social profiles that presumably have a corresponding scientific profile.

To find the social profiles appropriate for matching, we try to find the profiles of those people who have common scientific interests. There are some profiles in social networks, which correspond to scientific events (e.g. workshops, conferences, etc.). People with common interests are members of these events and share their news and opinions about them. Those individuals who follow these social events (directly or indirectly) are more likely to have a corresponding profile in the scientific network. Table 1 shows example mappings of social and scientific events in DBLP and Twitter networks.

We use a focused crawler to collect the social profile of those people who follow these events directly or indirectly. It starts crawling from people who directly follow event profiles (i.e. seed profiles) and uses follow[4] relation between people to find new profiles. For each collected profile, it uses some name similarity patterns to find the candidate scientific profiles for matching. If it cannot find any candidate for a given social profile, it will continue crawling from other paths. As indicated in Algorithm 1, the crawler continues until a maximum number of candidate pairs is collected. Using name matching patterns introduced in [33], we use *exact*, *prefix* and *all* patterns to find candidate pairs. Table 2, indicates these patterns for a person name with three name parts (in this table each NP represents a name part).

[4]follow relationship is a directed relationship between profiles of the Twitter, but for generality and simplicity, we ignore its direction.

Table 2 Name matching patterns

This example indicates three types of pattern (i.e. exact match, prefix match and all match) for searching a three-part name. Name = NP_1 NP_2 Np_3

Pattern name	Regular expression of matching pattern
Exact match	^NP_1 NP_2 NP_3\$
Prefix match	^NP_1(\w)$^+$\s$NP_2$(\w)$^+$\s$NP_3$(\w)$^+$\$
	^NP_1(\w)$^+$\s$NP_2$(\w)$^+$\s$NP_3$(\w)$^+$\$
	^NP_1(\w)$^+$\s$NP_3$(\w)$^+$\s$NP_2$(\w)$^+$\$
	^NP_2(\w)$^+$\s$NP_1$(\w)$^+$\s$NP_3$(\w)$^+$\$
All match	...
	^NP_3(\w)$^+$\s$NP_2$(\w)$^+$\s$NP_1$(\w)$^+$\$

Algorithm 1 Focused Crawler Pseudo-Code

```
function FocusedCrawler(list l)
                                   ▷ l is list of common scientific and social events
    for all s ∈ l do
enqueue(queue, extract_follower(s))          ▷ queue is a ordinary queue structure
    end for
    visited ← 0
    while visited ≤ MAX_VISIT do
        v_S ← dequeue(queue)
        namePatterns ← generateNamePattern(name(v_S))
        v_DS ← Φ
                    ▷ v_DS is the list of scientific candidate profiles to match with v_S
        for all p ∈ namePatterns do
            v_DS ← v_DS ∪ searchSientificNetwork(p)
     ▷ searchSientificNetwork finds a list of scientific profiles with names compatible
with pattern p
        end for
        if ||v_DS|| > 0 then
            enqueue(queue,extract_follower(v_S))
            visited ← visited+1
            for all v_D ∈ v_DS do
                addMatchingPair(v_D,v_S)
                                                   ▷ add a new candidate pair
            end for
        end if
    end while
end function
```

3.2 Matching problem

In previous section, we explained how to collect social and scientific profiles for matching. The output from the selection phase is a set of social and scientific candidate pairs, which match to each other according to a name similarity pattern. Due to name ambiguity, a large portion of collected candidate pairs is not actual matching pairs. For the matching sub-problem, the goal is to find true matching pairs from the set of collected candidate pairs. Let $V_D = \{d_1, d_2, ..., d_n\}$, $V_S = \{s_1, s_2, ..., s_n\}$ and $C_{DS} = \{(d_i, s_j) | d_i \in V_D, s_j \in V_S\}$ be the set of scientific profiles, social profiles and candidate pairs collected during the selection phase correspondingly. The matching problem can be reduced to labeling each member of C_{DS} as a true or false matching pair.

Note that the prior probability of the candidate pair (d_i, s_j) for being a true pair of matching, is inversely related to the distance of s_j from its seed profile. As mentioned before, the main criteria for selecting the pair (d_i, s_j) as a candidate pair is the existence of 1) a path between s_j and its parent seed profile and 2) a name similarity pattern between d_i and s_j. Considering these criteria, we expect a fairly high prior probability for the true labels. However, there are two important cases that cause false matching pairs. Firstly, follow-relationship between social profiles does not necessarily indicate a scientific relation. Hence, it is very probable that a social profile candidate for matching may have some friends who have not a corresponding scientific profile and vice versa. In these cases, the candidate social profile is not an appropriate profile for matching. Intuitively, the more distant the social profile from its seed, the less probability of being a valid selection. Secondly, the selected social profile may be an appropriate one for matching, while because of the name ambiguity, the selected pair is not a valid matching.

Using discriminative features associated with each candidate pair, we can utilize the well-known classification methods (e.g. logistic regression, SVM, decision tree, etc.) to classify each candidate pair. These methods independently predict the label (either true or false) of each candidate pair (i.e. independent prediction). In the next section, we introduce logistic regression as a representative method of independent classification algorithms.

3.3 Independent label prediction using logistic regression

Logistic regression is a widely used classification algorithm which predicts the label of each instance independent of other instances. For scientific-social networks integration, a logistic regression classifier can be used to predict the label of each candidate pair. We can use several indicative features associated with each candidate pair to train the classifier utilizing a set of training instances $TrainSet = \{(x_1; t_1)...(x_n; t_n)\}$ where x_i is the feature vector associated with the candidate pair i, $t_i \in \{true, false\}$ is its corresponding label and n is the number of training instances. While each candidate pair i is associated with a social profile $s \in V_S$ and a scientific profile $d \in V_D$, the classifier determines if the profile d is a match for s. We use the parametric form of logistic regression to predict the label of each candidate pair $p(t_i|x_i)$:

$$p(t_i = 1|x_i) = \frac{1}{1 + \exp(\theta x_i)} \tag{1}$$

$$p(t_i = 0|x_i) = \frac{\exp(\theta x_i)}{1 + \exp(\theta x_i)} \tag{2}$$

Where vector x_i is the feature vector of the candidate pair i and vector θ represents the corresponding weights for each feature. Training in this model is to find the vector θ that maximizes the conditional log likelihood of the training data:

$$\log \mathcal{L}(\theta \mid X, T) = \sum_{i=1}^{n} \log P(t_i|x_i; \theta) \tag{3}$$

In this equation, $T = \{t_1, t_2, ..., t_n\}$ and $X = \{x_1, x_2, ..., x_n\}$ represent the set of labels and the set of feature vectors for each candidate pair in the training set respectively. The above likelihood function is convex and has a unique global maximum which

can be found numerically [28]. After learning the parameter θ, we can use (1) and (2) to predict the most probable label for a given test instance (i.e. a candidate pair for matching).

3.4 Candidate pairs label dependence

Logistic regression model is one of the most effective techniques for binary classification. However, it makes its decisions only based on the features of individual candidate pairs and do not utilize the dependencies between them. In social-scientific network integration, the label of each candidate pair is not independent of other pairs. We consider two cases of dependencies between candidate pairs.

First of all, in many cases, scientific collaborators are also social friends. So, if a common friend (in both social and scientific networks) exists for a candidate pair $(d_i, s_j) \in C_{DS}$, the probability of classifying this pair as a true matching pair should be increased, but finding a common friend is impossible until we resolve all matching pairs. It means that we should jointly decide the labels of two pairs (d_i, s_j) and (d_k, s_l), if d_i and d_j are scientific collaborators and s_j and s_l are social friends. We refer to this type of dependency between candidate pairs as dependency *type* 1.

Secondly, since scientific networks (e.g. digital libraries) use sophisticated algorithms and manual effort to disambiguate people names, we expect that in most cases each person has at most one scientific profile. On the other hand, the majority of people have at most one profile in a social network. These assumptions mean that the label of two candidate pairs (d_i, s_k) and (d_j, s_k) are dependent on each other. Specifically, if d_i is already determined as a true match for s_k, the probability of matching (d_j, s_k) should be reduced.

We refer to this type of dependency between candidate pairs as dependency *type* 2. Likewise, the label of two candidate pairs (d_l, s_m) and (d_l, s_t) are dependent to each other. If d_l is already determined as a true match for s_m, the probability of matching (d_l, s_t) should be reduced. We refer to this type of dependency between candidate pairs as dependency *type* 3.

Figure 2 illustrates a real example of the *type* 1 dependency. In this figure, three candidate pairs have been shown. White and gray nodes indicate social and scientific profiles respectively, and the label of each node indicates the key attribute of the profile (e.g. username for Twitter network and URL key for DBLP network). As indicated in this figure, *Marshini Chetty*'s candidate social profile is followed by *Andrea Grimes Parker*'s one and also their candidate scientific profiles are related to each other by co-author relationship. Intuitively, this dependency can increase the matching probability for both candidate pairs. The mentioned dependency also exists between *Desney S. Tan*'s and *Andrea Grimes Parker*'s candidate profiles.

Figure 3 illustrates an example of the *type* 2 dependency. Four candidate pairs are illustrated, which are dependent on each other by *type* 2 dependency (e.g. a single social profile is simultaneously the candidate for four scientific profiles). If a classifier independently predicts the label of each candidate pair, it will assign all of them the same label because the social and scientific names are very similar in this case (Social name of *@nlpnoah* is Noah Smith and scientific candidate names are Noah W. Smith, Noah A. Smith, Noah H. Smith and Noah Torp-Smith). However, if we jointly predict the labels of these candidate-pairs, the set of labels will most probably contain at most one pair with true label.

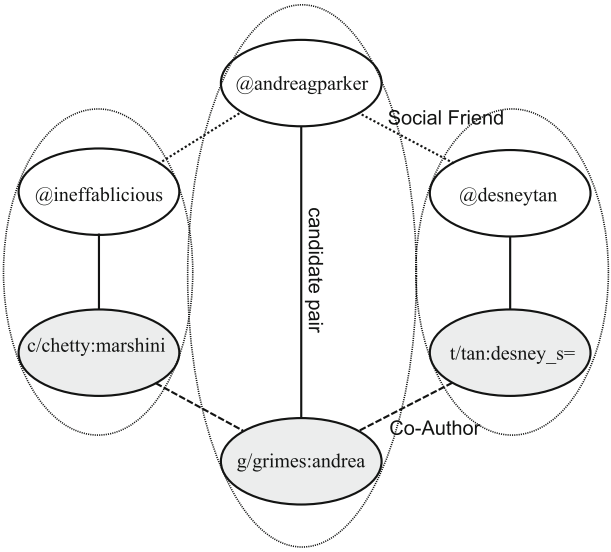

Figure 2 *Type* 1 dependency between candidate pairs. *White and gray nodes* indicate social and scientific profiles respectively. The labels of the nodes are Twitter UserName and DBLP unique URL key. Co-author and follow relationships are indicated by dash and candidate pairs are indicated by *solid lines*

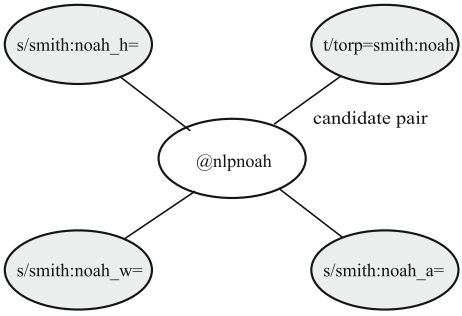

Figure 3 *Type* 2 dependency between candidate pairs. *White and gray nodes* indicate social and scientific profiles respectively. The labels of the nodes are Twitter UserName and DBLP unique URL key

Figure 4 *Type* 3 dependency between candidate pairs. *White and gray nodes* indicate social and scientific profiles respectively. The labels of the nodes are Twitter UserName and DBLP unique URL key

Figure 4 illustrates an example of the *type* 3 dependency. In this figure, the labels of two candidate pairs are dependent on each other because a single scientific profile is simultaneously the candidate for matching with two social profiles. In this example, social names are exactly the same (e.g. *Jonathan Bowen*) and both social profiles have the same chance to match with *Jonathan P. Bowen*'s scientific profile. However, if we jointly predict the label of candidate pairs, at most one of them will be selected as the true matching pair.

3.5 Matching dependency graph

In the previous section, we mentioned some examples that indicate the dependencies among candidate pairs. These dependencies are quite intuitive and could help us to make a joint prediction. In this section, we introduce a graph representation which captures and models these dependencies. We refer to this graph as the matching dependency graph (i.e. MDG).

As explained before, each instance of the matching problem can be formulated by the following set of profiles and relationships: $V_D = \{d_1, d_2, ..., d_k\}$ and $V_S = \{s_1, s_2, ..., s_m\}$ are the set of scientific and social profiles. Within each network, there exist relationships that indicate social friendship among members of V_S and co-author relationship among members of V_D. $E_D = \{(d_i, d_j)|d_i, d_j \in V_D \wedge Co-author(d_i, d_j)\}$ indicates the co-authorship relation between scientific profiles and $E_S = \{(s_l, s_n)|s_l, s_n \in V_S \wedge Follow(s_l, s_n)\}$ indicates the social friendship between social profiles. During selection phase, the focused crawler finds for each social profile some few matching candidates in the scientific network. We can indicate the candidate pairs by:

$$C_{SD} = \{(s_i, d_j)|s_i \in V_S \wedge d_j \in V_D \wedge CandidMatch(s_i, d_j)\}$$

Figure 5 illustrates an instance of matching problem (i.e. output of focused crawler). In this figure, the nodes in V_D and the edges in E_D are indicated by red color while the nodes in V_S and the edges in E_S are indicated by blue color. Each candidate pair is also represented by a black edge. As mentioned before, in the matching problem we should decide the label of each candidate pair as either a true or a false matching pair.

We formally define matching dependency graph $MDG(V_{MDG}, E_{MDG})$ as follows. Each node in MDG corresponds to exactly one candidate pair in C_{SD} as defined by:

$$V_{MDG} = \{(s_i, d_j)|s_i \in V_S, d_j \in V_D, (s_i, d_j) \in C_{SD}\}$$

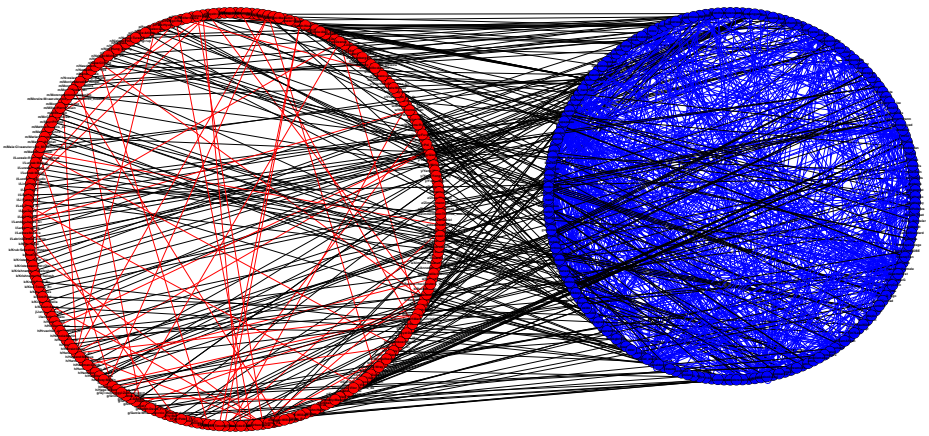

Figure 5 A sample of matching problem instance V_S and E_S are indicated in *blue*, V_D and E_D are indicated in *red* and C_{SD} are indicated in *black*

According to those three types of dependencies introduced in Section 3.4, we define three types of edges in *MDG* graph as

$$E_{MDG} = E_1 \cup E_2 \cup E_3$$

The edges in E_1 capture the *type* 1 dependency between nodes in V_{MDG} and can be defined as

$$E_1 = \{((s_i, d_j), (s_m, d_n))|s_i, s_m \in V_S \wedge d_j, d_n \in V_D \wedge (s_i, s_m) \in E_S \wedge (d_j, d_n) \in E_D\}$$

The *type* 2 dependency between nodes of V_{MDG} is indicated using the edges in E_2 and it can be defined as

$$E_2 = \{((s_i, d_j), (s_m, d_n))|s_i, s_m \in V_S \wedge d_j, d_n \in V_D \wedge s_i = s_m \wedge d_j \neq d_n\}$$

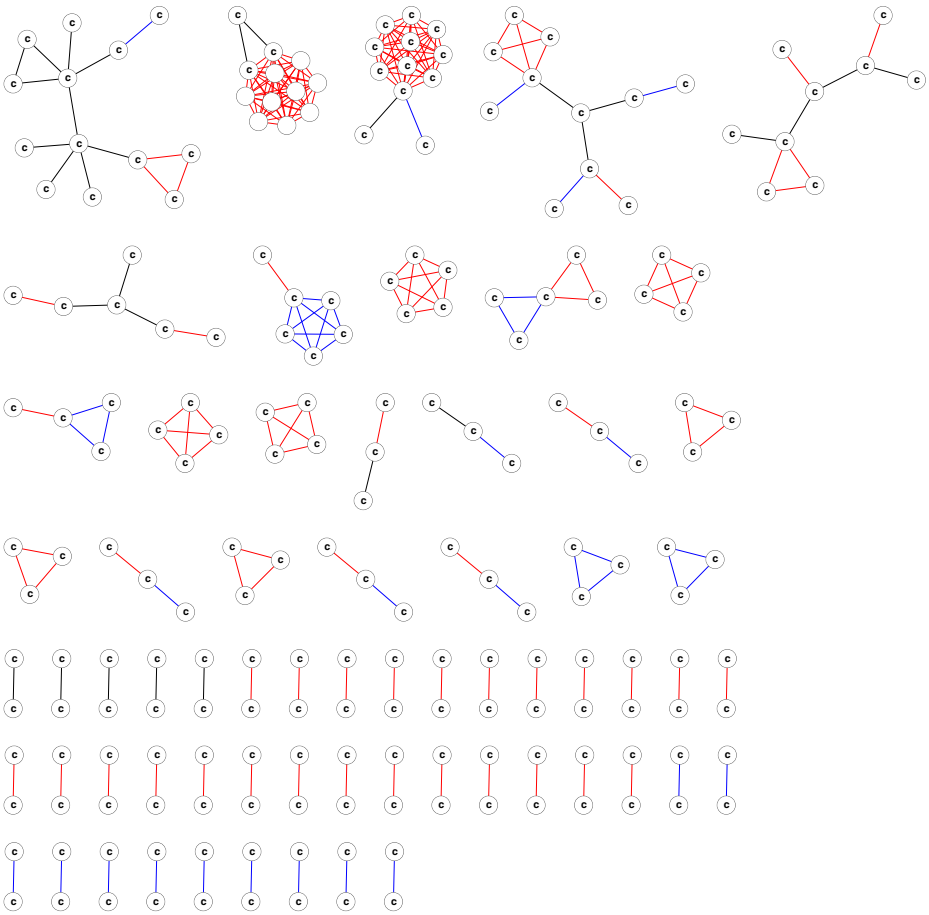

Figure 6 Corresponding MDG graph of Figure 5 E_1, E_2 and E_3 are *black*, *blue* and *red* respectively

The edges in E_3 represent the *type* 3 dependency between nodes of V_{MDG} and can be defined as

$$E_3 = \{((s_i, d_j), (s_m, d_n))|s_i, s_m \in V_S \wedge d_j, d_n \in V_D \wedge s_i \neq s_m \wedge d_j = d_n\}$$

Figure 6 illustrates corresponding *MDG* graph of Figure 5. In this figure each node represents a candidate pair and edges indicate different types of dependencies between them. Dependencies of *type* 1, *type* 2 and *type* 3 are indicated by black, blue and red edges respectively. In next section, we introduce a joint prediction model for labeling the *MDG*'s nodes.

3.6 Joint label prediction using conditional random field

Given the *MDG* graph defined above, the matching problem can be reduced to *jointly* predict the label (either true or false) of all candidate pairs (i.e all nodes in *MDG*) simultaneity. The label of each node in *MDG* graph indicates if its corresponding profiles (refer to the definition of V_{MDG} in Section 3.5) are true matches of each other and each edge in *MDG* graph indicates the label dependency between two candidate pairs (i.e. two neighbor node). As descried in Section 3.3, the label of each candidate pair (i.e the label t_i of each node v_i in *MDG*) is dependent on the observed feature vector (x_i) of its corresponding profiles. On the other hand, as described in Section 3.5, the label of each node in *MDG* is dependent to the label of its neighbor nodes. Therefore, we expect better label prediction by considering both effects of node features and their label dependency's simultaneity. In this section, we propose a joint label prediction method that captures both mentioned effect simultaneity.

Relational classification is a natural solution for our joint label prediction problem. By definition [28], relational data has two characteristics: first, statistical dependencies exist between the entities, and second; each entity has a rich set of features that can aid classification. The main idea of relational classification is to represent the distribution of target random variables (i.e. the label of each node in *MDG*) by a product of local functions (i.e. potential function) that each depends on only a small number of variables.

Considering two main effective factors on label prediction in *MDG* graph (i.e. node feature set and label dependency among neighbor nodes), following the idea of conditional random field [19], we can define two types of potential function in our model namely, node potential function and edge potential function. Node potential function is responsible to capture the dependency of the label t_i on the observed feature x_i for each node v_i of *MDG* and edge potential is responsible to model the label dependency among neighbor nodes in *MDG* graph.

According to the definition of Conditional Random Field [19], we can estimate the joint conditional probability of a particular label assignment T given observed feature X as a normalized product of the following set of non-negative potential functions.

$$P(T|X) = \frac{1}{Z} \prod_{i=1}^{n} \Phi_1(t_i, x_i) \prod_{e_{lm} \in E_1} \Phi_2(t_l, t_m) \prod_{e_{kn} \in E_2} \Phi_3(t_k, t_n) \prod_{e_{jh} \in E_3} \Phi_4(t_j, t_h) \qquad (4)$$

In this equation, $T = \{t_1, t_2, ..., t_n\}$ is the set of assigned labels for all nodes of *MDG* where n is the number of nodes and $t_i \in \{true, \ false\}$ is the random variable indicating

the assigned label for node v_i. $X = \{x_1, x_2, ..., x_n\}$ is the set of observed feature vectors, where x_i is the feature vector associated with node v_i and e_{ij} indicates the edge connecting two nodes v_i and v_j. Z is a normalizing factor that guarantees $P(T|X)$ is a valid distribution.

In (4), the node potential function (i.e. Φ_1) gives a non-negative weight to each possible outcome of the random variable t_i (i.e. true or false) according to the observed feature vector x_i. Corresponding to three different types of dependencies among neighbor nodes in *MDG*, we define three edge potential functions in (4) where Φ_2, Φ_3 and Φ_4 capture *type* 1, *type* 2 and *type* 3 label dependency respectively. The edge potential functions, defined on the edge e_{ij} connecting node v_i and v_j, give a non-negative weight to all four combinations that labels of v_i and v_j can take. (i.e. (t_i,t_j)=(false,false), (false,true),(true,false) or (true,true)).

Each type of edge potential functions defined above, assigns different weights for these combinations. For example, as discussed in Section 3.6, two neighbor nodes v_i and v_j which are connected by a *type*1 edge are more likely to have the same label, so we expect that Φ_2 gives higher weight/potential to the configurations in which t_i and t_j take the same label (especially for (true, true) combination). In contrast, *one-to-one* matching dependency discussed in Section 3.6, suggests that two neighbor nodes v_i and v_j that are connected by a *type*2 or *type*3 edge are more likely to have different labels. In other words, we expect that at most one of them take the true label. Therefore, we expect that Φ_3 and Φ_4 give lower weight/potential to the configurations in which t_i and t_j take the same label (especially for (true, true) combination).

Basically, each potential function can be an arbitrary non-negative function, but according to [28], the most widely-used type of potential functions are log-linear functions. Log-linear potential functions can be defined as the weighted combination of the observed feature variables. This type of potential function is appealing since it is jointly convex in the parameters of the model. Using log-linear potential functions, we can re-write conditional probability of the label set T given the observed feature variable X as follows:

$$P(T|X) = \frac{1}{Z'} exp\left\{ \sum_{i=1}^{n} \psi_1(x_i, t_i) + \sum_{e_{lm} \in E_1} \psi_2(t_l, t_m) + \sum_{e_{kn} \in E_2} \psi_3(t_k, t_n) + \sum_{e_{jh} \in E_3} \psi_4(t_j, t_h) \right\}$$

In this equation where the parameters are similar to (4), the exponential function guarantees that $P(T|X)$ yields a positive value, and Z' is the normalization constant which guarantees that $P(T|X)$ sums to 1 defined as:

$$Z' = \sum_{t'_1 \in \{\text{true,false}\} \ldots t'_n \in \{\text{true,false}\}} \left(exp\left\{ \sum_{i=1}^{n} \psi_1(x_i, t'_i) + \sum_{e_{lm} \in E_1} \psi_2(t'_l, t'_m) + \sum_{e_{kn} \in E_2} \psi_3(t'_k, t'_n) + \sum_{e_{jh} \in E_3} \psi_4(t'_j, t'_h) \right\} \right)$$

Using log-linear potential functions [28], each potential function ψ_1, ψ_2, ψ_3, ψ_4 is represented by weighted combinations of feature vectors in the following form:

$$\psi_1(x_i, t_i) = \sum_{m=1}^{M_1} \theta_m f_m(x_i, t_i)$$

$$\psi_2(t_i, t_j) = \sum_{m=1}^{M_2} \alpha_m g_m(t_i, t_j)$$

$$\psi_3(t_i, t_j) = \sum_{m=1}^{M_3} \beta_m h_m(t_i, t_j)$$

$$\psi_4(t_i, t_j) = \sum_{m=1}^{M_4} \zeta_m s_m(t_i, t_j)$$

Where θ, α, β and ζ represent trainable weight vectors, f, g, h and s represent features vectors and M_1, M_2, M_3 and M_4 represent the number of features for each potential function.

Similar to the logistic regression label prediction method introduced in Section 5.3, we use an extensive set of features to train ψ_1 potential function and for edge potential functions (i.e. ψ_2, ψ_3 and ψ_4), we define a set of binary/indicative features that captures the compatibility of labels among two neighbor nodes. Binary features associated with ψ_2 are defined as follows:

$$g_1(t_i, t_j) = \neg t_i \wedge \neg t_j$$

$$g_2(t_i, t_j) = \neg t_i \wedge t_j \vee \neg t_j \wedge t_i$$

$$g_3(t_i, t_j) = t_i \wedge t_j$$

For each combination of labels assigned to two neighbor nodes t_i and t_j, the value of one of the above-mentioned features is 1 and other features will be zero. For example, if both t_i and t_j take true labels, then the value of g_1, g_2 and g_3 will be zero, zero and one respectively. Specifically, feature g_2 indicates conflicting label assignment and g_1 and g_3 indicate homogenous label assignment for two neighbor nodes t_i and t_j . Since MDG is an undirected graph, only three features are sufficient to model all combinations of labels assigned to t_i and t_j. In other words, the value of g_2 will be 1 for conflicting combinations regardless of order of nodes. We define the binary features of ψ_3 and ψ_4 analogously.

Training in the proposed model is to find vectors θ, α, β and ζ that maximize the conditional log likelihood of the training data as defined below. In our proposed

model, training data is an instance of *MDG* graph with known values of labels and features for each node.

$$\log \mathcal{L}(\theta, \alpha, \beta, \zeta \mid X, T) = \sum_{i=1}^{n} \log P(t_i|x_i; \theta) + \sum_{e_{lm} \in E_1} \log P(t_l, t_m; \alpha) + \sum_{e_{kn} \in E_2} \log P(t_k, t_n; \beta) + \sum_{e_{jh} \in E_3} \log P(t_j, t_h; \zeta)$$

In this equation, the unknown parameters are θ, α, β and ζ while the value of each t_i and x_i are given as an instance of *MDG* graph (e.g. training instance).

Despite there is no closed-form solution for the above maximization problem, the above log likelihood function is convex and can be efficiently maximized by iterative searching algorithms such as BFGS [22]. It is worth noting that if we remove all edges of the *MDG* graph, the above equation will be the same as (3). In other words, logistic regression is a special case of the aforementioned model where there is no edge between the nodes of *MDG* graph.

After learning the parameters of the model using an instance of *MDG* graph, we can jointly predict the label of all nodes for a given test instance of *MDG* graph. (i.e. an *MDG* graph with unknown values of labels and known values of features for each node.) The prediction (also known as inference [28]) in our conditional model is to compute the posterior distribution over the label variables T given a test instance of *MDG* graph with observed values of node features X, i.e., to compute the following most probable assignment of labels:

$$T^* = argmax_T P(T|X)$$

Exact inference can be done efficiently for a set of dependent variables with simple graph topologies such as sequences and trees [22]. However, the proposed *MDG* graph goes beyond these simple topologies, and exact inference is usually intractable in this case. Belief propagation is an exact inference algorithm for simple topologies such as sequences and trees, which generalizes the forward-backward algorithm [28]. Although this algorithm is neither exact nor guaranteed to converge for loopy graph structures, it is still well-defined and relatively efficient, specifically for sparse graphs. Therefore, we resort to approximate loopy belief propagation [28] to find the most probable assignment of labels for a given *MDG* test. As indicated in Figure 6, the *MDG* graphs resulting from the three cases of dependencies are usually not densely connected in real cases. So, the inference task can be done efficiently by belief propagation for the proposed graphical model.

4 Experiments

4.1 Data

We test our proposed models on six test collections collected from the Twitter social network and the DBLP scientific network. To build these test collections, we use the crawler (described in Section 3.1) to collect Twitter profiles and their corresponding candidate DBLP profiles. General statistics of the crawled profiles are reported

Table 3 General statistics of the crawled profiles form Twitter and DBLP

Number of crawled Twitter profiles	61863
Number of crawled DBLP Profiles	136596
Average number of candidate DBLP profiles for each Twitter profile	3.65
Average number of candidate Twitter profiles for each DBLP profile	1.65

in Table 3. According to this table, for each Twitter profile there are on average 3.65 candidate DBLP profiles and for each DBLP profile there are on average 1.65 candidate Twitter profiles for matching. We build one automatically generated and five manually annotated test collections from these collected candidate pairs.

We automatically generate a test collection, which is called *URL* test collection in our experiments. Twitter and DBLP profiles have an optional field that can be filled with the homepage URL address of the profile owner. 56.97 % of all the collected Twitter profiles and 1.8 % of all the collected DBLP profiles have a valid URL address, which can be used as a unique identifier of the profile owner. We used a simple string matching method to find Twitter and DBLP profile pairs with exactly the same URL address. We found 173 Twitter profiles, which have a unique corresponding DBLP profile with the same URL address and used these pairs as positive instances. For this set of automatically matched Twitter and DBLP profiles, we used all other candidates found by the crawler as the negative instances. The set of negative instances includes non-matching DBLP and non-matching Twitter profiles.

Apart from the automatically generated test collection, we also build five other manually annotated test collections to evaluate the proposed matching algorithms. As mentioned in Section 3.1, we used several seed profiles to collect profiles form the Twitter social network. Each of these seed profiles (refer to Table 1) is related to a well-known computer science event such as a conference, workshop or journal. We can topically categorize the set of collected Twitter profiles according to the topic of their seed parent. Selected seed profiles cover a broad range of topics in computer science research community. In order to categorize seed profiles according to their topics, we used the WikiCFP topical tags related to each of these seed profiles.[5] In our experiments, the total number of seed profile is 152, and we extract 67 related tags from WikiCFP, while each seed profile is associated with 1.66 tags on average. Table 4 illustrates a few examples of seed profiles and their associated tags. While some of the extracted tags are very specific (e.g. object-oriented programming, virtual reality, etc.), some others cover broader topics in computer science (e.g. software engineering, semantic web, etc.). After removing tags with less than two occurrences, we categorized the remaining tags into five main topics in computer science. The main topics and the number of associated seeds and tags are illustrated in Table 5.[6]

Table 6 illustrates the number of the collected Twitter profiles for each main topic. In this table, the number of the collected Twitter profiles is separated by the distance of the Twitter profile from its parent seed. This table also illustrates the fraction of ambiguous Twitter profiles (i.e. profiles with more than one candidate DBLP profile)

[5] WikiCFP is a forum for researchers to share news about call for papers. http://www.wikicfp.com.

[6] Due to several common tags in the Information Retrieval and Data Mining topics; we combine them in a single topic as DM-IR topic.

Table 4 Examples of seed profiles and their associated tags extracted from the WikiCFP

Seed profile	WikiCFP tag	Seed profile	WikiCFP tag
@aamas2012	Agents	@KDD2011	Data mining
@clef2011	Information retrieval	@acmchi2012	HCI
@sosp2011	Operating systems	@PLDI	Programming language

Table 5 Five main topics related to the seed profiles

Topical collection	DB	DM-IR	HCI	OS	SF
# associated tags	2	27	2	7	8
# associated seeds	3	116	11	10	12

DB = Database, DM-IR = Data mining and Information Retrieval, HCI = Human Computer Interaction, OS = Operating Systems, SF = Software

Table 6 Number of collected Twitter profiles and fraction of ambiguous profiles for each main topic separated by distance form parent seed

Collection	# of collected Twitter profiles			Fraction of ambiguous profiles		
	Distance 1	Distance 2	Distance 3	Distance 1	Distance 2	Distance 3
DB	222	2011	8592	0.26	0.37	0.54
DM_IR	2034	8362	23453	0.40	0.44	0.55
HCI	589	4495	19340	0.33	0.44	0.55
OS	217	1781	12067	0.39	0.44	0.55
SF	443	3116	18667	0.33	0.46	0.55

Profiles of a topic may have overlap with profiles of other topics

Table 7 Detailed statistics of the test collections

Statistics/Dataset	DB	DM-IR	HCI	OS	SF	URL
Number of candidate pairs collected by crawler	540	873	617	800	732	619
Number of Twitter having no DBLP	145	305	197	256	264	35
Number of edges of *type*1 in *MDG*	28	8	27	9	38	31
Number of edges of *type*2 in *MDG*	383	807	433	656	597	290
Number of edges of *type*3 in *MDG*	132	515	201	308	353	205

URL is the automatically generated test collection and other test collections are named by the abbreviations introduced in the Table 5

separated by distance from the seed parent. As mentioned before, the fraction of the ambiguous Twitter profiles grows with the distance from the parent seed.

400 Twitter profiles are randomly chosen for each main topic to build the topical test collections. For these randomly selected Twitter profiles and their corresponding DBLP candidate profiles, which are collected by the crawler, two human assessors are asked to determine the label of each candidate pair. They used several external evidence to determine the label of each candidate pair. For example, they used the information on the web (e.g. homepage) as well as other social-networking websites (e.g. the Facebook social network, the LinkedIn professional network) to decide the label of each pair. In some cases, they also decided the label of candidate pairs based on the topic similarity of their associated Tweets and papers. Table 7 gives detailed statistics of the data collections.

We can notice that the test collections have different characteristics. In particular, the number of the Twitter profiles which do not have any DBLP matching profile is smaller in the *URL* test collection in comparison with other test collections. It comes from the method, we select the Twitter profiles for the *URL* test collection. As mentioned before, we use exact URL matching to select Twitter profiles (positive instances) for this test collection, but for other test collections, we randomly select the Twitter profiles from the output of the focused crawler. Furthermore, there are more edges of type two and three in the DM-IR test collection in comparison with other test collections. This may come from the fact that in this collection, more ambiguous names are occurred.

In our experiments, we used the negative and the positive candidate pairs of five collections to train each proposed discriminative model and used the candidate pairs of the remaining collection as the test set. Precision, Recall and the F-measure are used to evaluate the proposed models, which are defined as follows:

$$P = \frac{\text{Number of correctly (true) predicted matching pairs}}{\text{Number of predicted matching pairs}}$$

$$R = \frac{\text{Number of correctly (true) predicted matching pairs}}{\text{Number of correct(true) matching pairs}}$$

$$F = \frac{2PR}{P+R}$$

4.2 Experiments setup

In our experiments, we compared the matching performance of 1) a simple heuristic method, 2) independent label predication methods, 3) proposed joint label prediction method and 4) the method proposed in the [33].

Simple heuristic method which is called SIMPLE method in our experiments, matches each Twitter profile to *exactly one* DBLP profile. For each Twitter profile, the SIMPLE method selects the DBLP profile with most name similarity as the true match between the set of DBLP candidate profiles found by the crawler. In other words, the SIMPLE method assumes that each Twitter profile has exactly one matching profile in DBLP and selects it based on the name similarity.[7]

To train the independent and joint classification models, we use three sets of features as presented in Table 8. The total features are divided into five groups based on where the feature comes from, namely, 1) Twitter homepage URL features, 2) Twitter location feature, 3) Twitter-DBLP name features, 4) Twitter Description features and 5) Twitter-DBLP profile features. For example, the Twitter homepage URL features include some binary and real features that indicate the properties of the homepage URL reported by the Twitter profile owner. Specifically, feature *URL* indicates whether or not the symbol tilda ~ is occurred in the homepage field of a Twitter profile. The Twitter-DBLP name similarity features indicate the similarity of names in a candidate pair, and the Twitter-DBLP profile features include

[7]We used the edit distance algorithm to measure the name similarity between DBLP and Twitter names.

Table 8 Feature group and feature sets used in discriminative learning

Category	Feature name	Minimal	Minimal+ Description	All	Type
Twitter homepage URL	Homepage URL exist?	✔	✔	✔	Binary
	Homepage URL Length	✔	✔	✔	Real
	# of slashes in URL	✔	✔	✔	Real
	Homepage URL has ∼?	✔	✔	✔	Binary
Twitter location	Twitter location exist?	✔	✔	✔	Binary
Name imilarity	Distance based (2 features)	✔	✔	✔	Real
	Pattern based (8 features)	✔	✔	✔	Binary
	Overlap based (2 features)	✔	✔	✔	Binary
Twitter description	Description Exists?	–	✔	✔	Binary
	Word (10 features)	–	✔	✔	Binary
Twitter-DBLP profile	Distance from parent seed	–	–	✔	Real
	Number of papers	–	–	✔	Real
	Number of followers	–	–	✔	Real
	Number of co-authors	–	–	✔	Real

crawling and profile information of candidate pairs. To examine the performance of each feature group, we used three sets of features to train proposed discriminative models. The *Minimal* feature set includes 1) the Twitter homepage URL, 2) the Twitter location features and 3) the Twitter-DBLP name similarity features. *Minimal+Description* includes the *Minimal* features as well as the Twitter description features. The Twitter description feature set includes some binary features that indicate the occurrence of some informative words such as "student", "computer", "research", "PhD" and so on in the description field of a Twitter profile. Finally, the *All* feature set includes *Minimal+Description* features as well as the Twitter-DBLP profile features. There are totally 32 features used in our experiments, and all the feature scores are normalized by the maximum score in that feature. In addition, for the joint prediction method, we used the binary features introduced in Section 3.6.

5 Results

An extensive set of experiments were conducted on the six test collections to address the following questions:

- How good are the discriminative independent label prediction approaches compared with the SIMPLE heuristic method? The experiments described in the Section 5.1 are conducted on SIMPLE method as well as logistic regression (LR), Support Vector Machine (SVM) and decision tree methods trained on the *Minimal* feature set.
- Can the prediction performance be improved by considering the dependency between the labels of the candidate pairs? Experiments in the Section 5.2 are conducted to compare the performance of the proposed joint prediction model with the logistic regression method (as an independent label prediction method).
- What is the impact of the different features on prediction? Experiments in the Section 5.3 are conducted to compare the performance of the proposed joint

prediction model, and the logistic regression method trained on different feature sets introduced in the Table 8.

– How good is the proposed joint classification approach in comparison with the method proposed in [33]? Experiments in the Section 5.4 are conducted to compare the matching performance of the proposed model with the method proposed in [33].

5.1 SIMPLE Heuristic method versus independent label prediction

In this experiment, we compare the SIMPLE heuristic method described in the Section 4.2 with the independent label predication methods (i.e. logistic regression, support vector machine and decision tree.) Table 9 contains the comparisons in F-score. In these experiments, we used the *Minimal* feature set described in Table 8.

We can see that all the independent classification methods improve upon the SIMPLE approach and usually LR, SVM and Decision Tree have almost the same performance. The SIMPLE method has almost the same behavior on all test collections except for two cases. Its F-score on the DM-IR collection is very low and on the *URL* test collection is very high. It may come from the ambiguity level of these test collections. As mentioned in Section 4.1, the DM-IR collection is the most ambiguous and the URL collection is the least ambiguous collection among other collections. Therefore, it seems that matching problem is easier to solve for the URL collection in comparison with other collections. In contrast, independent classification methods have almost the same performance on all test collections.

Figures 7 and 8 depict the precision and recall scores for SIMPLE and independent classification methods. We can see that the SIMPLE method usually has large recall in comparison with the independent classification methods, but it has very low precision (except for *URL* test collection). The high recall property of the SIMPLE method can be explained by the fact that people usually use very similar names in Twitter and DBLP networks. Therefore, if multiple DBLP candidates exist for a given Twitter profile, the most likely DBLP profile for matching will be the one with the most similar name to that Twitter name (exactly the same heuristic is used by the SIMPLE method). In contrast, the SIMPLE method has very low precision, which means that it is not able to recognize non-matching pairs that have very similar names. The independent classification methods can improve the F-score by enhancing the precision score, but this methods decrease the recall score substantially. It means that these methods tend to select only candidate pairs with

Table 9 SIMPLE method versus independent label prediction

Collection/Method	Simple	Decision tree	SVM	LR
DB	0.619	0.775	0.782	**0.804**
DM-IR	0.382	0.683	0.689	**0.709**
HCI	0.569	0.635	0.632	**0.678**
OS	0.474	0.767	0.764	**0.775**
SF	0.405	0.707	0.699	**0.731**
URL	0.794	**0.802**	0.789	0.785

Comparisons are based on F-measure

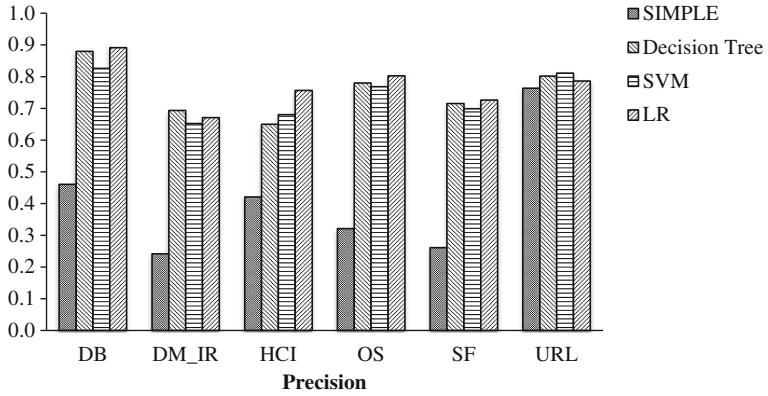

Figure 7 Precision comparison of SIMPLE method and independent label prediction approaches. Abbreviations are defined in the Table 5

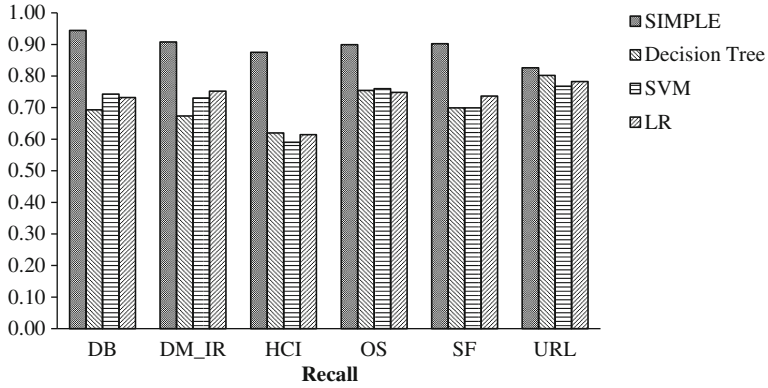

Figure 8 Recall comparison of SIMPLE method and independent label prediction approaches. Abbreviations are defined in the Table 5

very similar names as true matches. As a result, these methods miss a lots of true matching pairs (i.e. low recall).

5.2 Independent versus joint label prediction

In this experiment, using the *Minimal* feature set, we compare the matching performance of the logistic regression method with the joint label prediction method trained on the dependency *type* 1, *type* 2, *type* 3 and the combination of them. Table 10 contains the comparisons in F-score. In this table, CRF-1, CRF-2 and CRF-3 indicate the joint label prediction method for the *MDG* graph that has only edges of *type* 1, *type* 2 and *type* 3 respectively. CRF-123 indicates the joint label prediction method for the *MDG* graph with all mentioned dependency types. Table 10 shows that the method CRF-2 substantially improves the F-score in all test collections in comparison with the logistic regression method. Inspired from the SIMPLE method,

Table 10 Independent versus joint label prediction

Comparisons are based on F-measure. The * symbol indicates statistical significance at 0.9 confidence interval

Collection/Method	LR	CRF-1	CRF-2	CRF-3	CRF-123
DB	0.804	0.842	0.846	0.817	**0.861***
DM-IR	0.709	0.710	0.760	0.718	**0.774***
HCI	0.678	0.692	0.732	0.682	**0.736***
OS	0.775	0.763	0.783	0.752	**0.797***
SF	0.731	0.739	0.793	0.751	**0.812***
URL	0.785	0.796	0.871	0.785	**0.891***

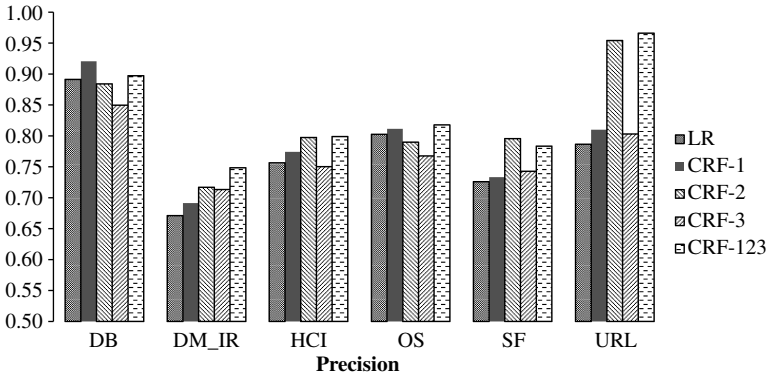

Figure 9 Precision comparison of the joint and the independent label prediction methods

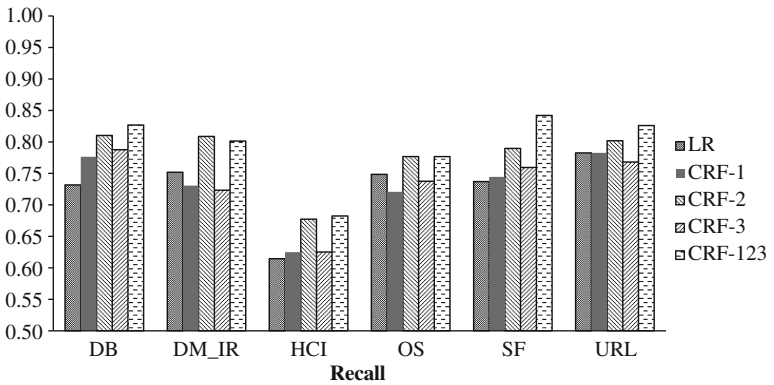

Figure 10 Recall comparison of the joint and the independent label prediction methods

CRF-2 only selects the most probable DBLP candidate for each Twitter profile as a true match, but using discriminative features it also prevents from many false negatives. Figures 9 and 10, indicate that this method improves the recall score but retains the precision in the same level in comparison with logistic regression. In other words, CRF-2 brings together the advantages of the SIMPLE method (i.e. high recall) and the logistic regression method (i.e. high precision). The average improvement of F-score using CRF-2 is 6.8 % for all test collections in comparison with logistic regression.

CRF-1 and CRF-3 generally increase the F-score, but their improvement is less than the CRF-2. CRF-3 improves the F-score 0.6 % on average and CRF-1 can improve it up to 1.3 % on average. Specifically, CRF-1 improves the precision on all the collections, but in two cases, slightly reduces the recall score (i.e. the DM-IR and the OS collections). CRF-123 considers all the dependency types in the *MDG* graph to predict the label of each candidate pair. In all the test collections, CRF-123 improves the precision and recall scores in comparison with logistic regression method, and it also has the best performance in F-score in comparison with other methods in all collections. The improvement of F-score using CRF-123 is 8.7 % averaged on all the test collections in comparison with logistic regression.

5.3 Impact of using different feature sets

In this experiment, we compare the matching performance of the logistic regression method with the best joint label prediction methods (i.e. CRF-123) for different sets of features defined in Table 8. Figure 11, indicates the precision of LR and CRF-123 methods on the different set of features. The general trend is that the precision can be improved for some test collections by adding the Twitter description and the Twitter-DBLP profile features. This improvement is more obvious for the DM-IR test collection which is the most ambiguous test collection. Another interesting trend is that in all cases, the precision of the CRF-123 is better than its corresponding logistic regression method (i.e. LR) in the same level of the feature set. It means that joint prediction model never decreases the precision measure, but it can also improve the recall measure and accordingly, the F-score.

Figure 12, indicates the recall score of the LR and CRF-123 methods on the different set of features. The general trend is that the recall score can be improved by adding the Twitter-DBLP profile features, but the Twitter description features may decrease the recall score on some collections. Similar to the precision score, for all feature sets, the recall score of CRF-123 is better than LR method.

Table 11 contains the F-score comparisons of LR and CRF-123 on the different feature sets. According to this table, the overall performance of matching can be

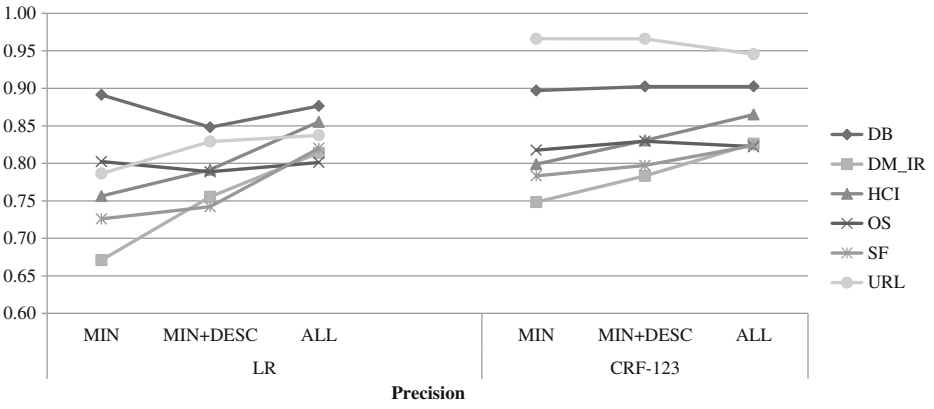

Figure 11 Precision comparison of joint and independent prediction methods for different set of features defined in the Table 8

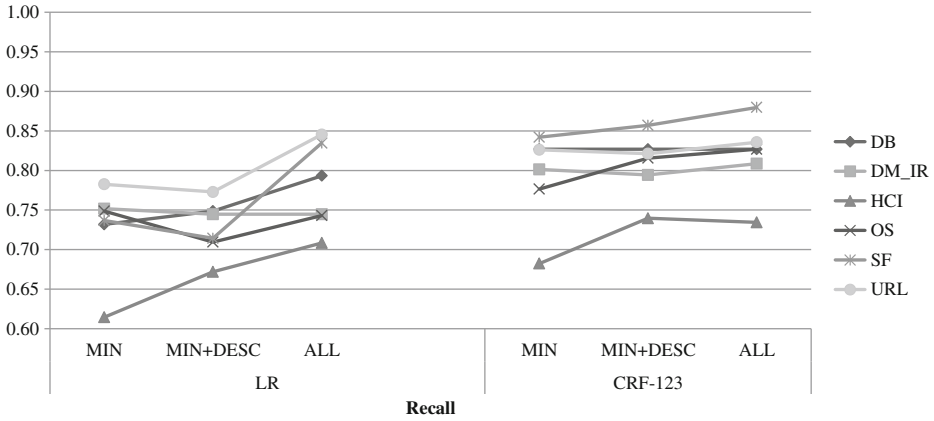

Figure 12 Comparison of joint and independent prediction methods for different set of features defined in Section 4.1- comparisons are based on recall

Table 11 Independent versus joint label prediction

Test collection	LR			CRF-123		
Feature set	MIN	MIN+DESC	ALL	MIN	MIN+DESC	ALL
DB	0.804	0.795	0.833	0.861	0.863	**0.863***
DM_IR	0.709	0.750	0.778	0.774	0.789	**0.817***
HCI	0.678	0.727	0.775	0.736	0.782	**0.794***
OS	0.775	0.747	0.771	0.797	0.823	**0.825***
SF	0.731	0.728	0.838	0.812	0.826	**0.851***
URL	0.785	0.800	0.841	**0.891**	0.888	0.887

F-scores are reported for different set of features introduced in Section 4.1. The * symbol indicates statistical significance at 0.9 confidence interval

improved by using the complete set of features (i.e. *All* feature set introduced in Table 8) for both CRF-123 and LR methods. It is worth noting that in all collections, and for all levels of the feature sets the F-score of CRF-123 surpass its corresponding LR in the same level of the feature set. The improvement is more salient on the *Minimal* feature set, which means that the proposed joint model of label prediction can be very useful in matching problems in which there is not enough set of features to integrate profiles of two networks.

5.4 Comparison with previous method

In this experiment, we compare the matching performance of the CRF-1 and CRF-123 methods with the proposed method in [33]. As mentioned in Section 2, You et al. [33] used numerical features to model the *common-friend* property. In contrast, CRF-1 and CRF-123 uses relational learning to model the dependencies between candidate pairs. Similar to [33], CRF-1 considers *common-friend* dependency between candidate pairs but CRF-123 considers both *common-friend* dependency and *one-to-one* matching dependency simultaneously. We implemented the same

Table 12 Comparison of Joint prediction and method proposed in [33]

Comparisons are based on F-scores. The * symbol indicates statistical significance at 0.9 confidence interval

Method	SS [33]	CRF-1	CRF-123
DB	0.833	0.833	**0.863***
DM_IR	0.769	0.769	**0.817***
HCI	0.778	0.778	**0.794***
OS	0.773	0.773	**0.825***
SF	0.850	0.844	**0.851**
URL	0.840	0.840	**0.887***

feature set proposed in [33] to predict the label of the collected candidate pairs by the crawler. Table 12 indicates that CRF-1 and SS [33] method have almost the same performance on all data collections, but CRF-123 surpass the SS method by a large margin. As mentioned above, SS and CRF-1 models the same property and almost have the same result, but our proposed relational learning model is more flexible to consider *one-to-one* matching dependency and it can improve the matching performance substantially.

6 Conclusions and future work

Social and Scientific network integration is a crucial step toward building a multi environment expert finding system. It is an important information retrieval task because each network has its own properties and characteristics, and the integration of them can help us to improve the quality of expert finding. This task is also closely related to other IR problems such as named disambiguation and homepage finding as presented in TREC Web Track. In this paper, we designed a focused crawler to collect high probable matching profile pairs in the DBLP and the Twitter networks. Using a bootstrapping method, the crawler starts to collect people' profiles form common profiles of the two networks and in each step using name similarity patterns, it collects profile pairs with high prior probability of matching. The network integration problem is then reduced to finding true matching pairs among these collected candidate pairs. We noticed that two important types of dependency exist between collected candidate pairs namely *common friend* dependency and *one-to-one* dependency. Considering these dependencies between candidate pairs of matching, we introduced a joint label predication method to predict the label of candidate pairs simultaneously. We tested our algorithm on six test collections collected from The DBLP and the Twitter networks, and our experiments indicate that the joint label prediction method can improve the F-score of matching up to 8.7 % averaged on all the test collections. Furthermore, utilizing the different sets of features for training, we concluded that the proposed profile matching method can be especially useful for matching problem with few available discriminative features. It is worth nothing that the mentioned dependencies in network integration problem are quite general and can be utilized to integrate other social and scientific networks. In the future, we intend to use the aligned profiles of The Twitter and The DBLP to obtain a better relevance score estimation for each expert according to his social and scientific interactions.

References

1. Balog, K., Bogers, T., Azzopardi, L., de Rijke, M., van den Bosch, A.: Broad expertise retrieval in sparse data environments. In: SIGIR '07: Proceedings of the 30th Annual international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 551–558. ACM Press, New York, NY, USA (2007)
2. Balog, K., Azzopardi, L., de Rijke, M.: A language modeling framework for expert finding. Inf. Process. Manage. **45**(1), 1–19 (2009)
3. Balog, K., Soboroff, I., Thomas, P., Craswell, N., de Vries, A.P., Bailey, P.: Overview of the TREC 2008 enterprise track. In: The Seventeenth Text Retrieval Conference Proceedings (TREC 2008), NIST Special Publication (2009)
4. Bekkerman, R., McCallum, A.: Disambiguating web appearances of people in a social network. In: Ellis, A., Hagino, T. (eds.), pp. 463–470. WWW, ACM (2005)
5. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Discov. Data **1**(1), 1–36 (2007)
6. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of IJCAI-03 Workshop on Information Integration, pp. 73–78 (2003)
7. Craswell, N., Hawking, D., Robertson, S.: Effective site finding using link anchor information. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01, pp. 250–257. ACM, New York, NY, USA (2001)
8. Deng, H., Han, J., Lyu, M.R., King, I.: Modeling and exploiting heterogeneous bibliographic networks for expertise ranking. In: 2012 ACM/IEEE Joint Conference on Digital Libraries (JCDL 2012). IEEE (2012)
9. Deng, H., King, I., Lyu, M.R.: Enhanced models for expertise retrieval using community-aware strategies. IEEE Trans. Syst. Man Cybern. Part B **42**(1), 93–106 (2012)
10. EntityCube: http://entitycube.research.microsoft.com/. Accessed April 2012
11. Fang, Y., Si, L., Mathur, A.P.: Discriminative graphical models for faculty homepage discovery. Inf. Retr. **13**(6), 618–635 (2010)
12. Fang, Y., Si, L., Mathur, A.P.: Discriminative probabilistic models for expert search in heterogeneous information sources. Inf. Retr. **14**,158–177 (2011)
13. Ferreira, A.A., Veloso, A., Gonçalves, M.A., Laender, A.H.: Effective self-training author name disambiguation in scholarly digital libraries. In: Proceedings of the 10th Annual Joint Conference on Digital Libraries, JCDL '10, pp. 39–48. ACM, New York, NY, USA (2010)
14. Han, H., Giles, L., Zha, H., Li, C., Tsioutsiouliklis, K.: Two supervised learning approaches for name disambiguation in author citations. In: Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '04, pp. 296–305. ACM, New York, NY, USA (2004)
15. Han, H., Zha, H., Giles, C.L.: Name disambiguation in author citations using a k-way spectral clustering method. In: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '05, pp. 334–343. ACM, New York, NY, USA (2005)
16. Hofmann, K., Balog, K., Bogers, T., de Rijke, M.: Contextual factors for finding similar experts. J. Am. Soc. Inform. Sci. Technol. **61**(5), 994–1014 (2010)
17. Kazai, G., Doucet, A.: Overview of the inex 2007 book search track (booksearch'07). In: Focused access to XML documents, Sixth International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007. Lecture Notes in Computer Science, vol. 4862, pp. 148–161. Springer (2008)
18. Kopcke, H., Rahm, E.: Frameworks for entity matching: a comparison. Data Knowl. Eng. **69**(2), 197–210 (2010)
19. 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, ICML '01, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
20. Ley, M.: Dblp: some lessons learned. Proc. VLDB Endow. **2**, 1493–1500 (2009)
21. Macdonald, C., Ounis, I.: Voting techniques for expert search. Knowl. Inf. Sys. **16**(3), 259–280 (2008). doi:10.1007/s10115-007-0105-3
22. McCallum, A.: Efficiently inducing features of conditional random fields. In: Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03) (2003)

23. Mccallum, A., Wellner, B.: Conditional models of identity uncertainty with application to noun coreference. In: NIPS 2004 (2004)
24. Moghaddam, S., Ester, M.: Ilda: interdependent lda model for learning latent aspects and their ratings from online product reviews. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '11, pp. 665–674. ACM, New York, NY, USA (2011)
25. Rode, H., Serdyukov, P., Hiemstra, D.: Combining document- and paragraph-based entity ranking. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, pp. 851–852. ACM, New York, NY, USA (2008)
26. Serdyukov, P.: Search for expertise: going beyond direct evidence. PhD thesis, Enschede (2009)
27. Smirnova, E., Balog, K.: A user-oriented model for expert finding. In: 33rd European Conference on Information Retrieval (ECIR 2011), vol. 6611, pp. 580–592. Springer (2011)
28. Sutton, C., McCallum, A.: 4. In: An Introduction to Conditional Random Fields for Relational Learning, pp. 93–128. MIT Press (2006)
29. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, pp. 990–998. ACM, New York, NY, USA (2008)
30. Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: UAI, pp. 485–492 (2002)
31. WebMynd: http://www.webmynd.com (2012)
32. Xi, W., Fox, E., Tan, R., Shu, J.: Machine learning approach for homepage finding task. In: Laender, A., Oliveira, A. (eds.) String Processing and Information Retrieval. Lecture Notes in Computer Science, vol. 2476, pp. 169–174. Springer Berlin / Heidelberg (2002)
33. You, G.w., Park, J.w., Hwang, S.w., Nie, Z., Wen, J.R.: Socialsearchs+: enriching social network with web evidences. World Wide Web 1–27 (2012). doi:10.1007/s11280-012-0165-5