A Joint Classification Method to Integrate Scientific and Social Networks

Mahmood Neshati[1], Ehsaneddin Asgari[3], Djoerd Hiemstra[2], and Hamid Beigy[1]

[1] Department of Computer Engineering, Sharif University of Technology
{neshati,beigy}@ce.sharif.edu
[2] Database Research Group, University of Twente
d.hiemstra@utwente.nl
[3] School of Computer and Communication Science (IC),
Ecole Polytechnique Fédéral de Lausanne - EPFL
ehsaneddin.asgari@epfl.ch

Abstract. In this paper, we address the problem of scientific-social network integration to find a matching relationship between members of these networks. 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. By defining matching dependency graph, we propose a joint label prediction model to determine the label of all candidate pairs simultaneously. An extensive set of experiments have been conducted on six test collections obtained from the DBLP and the Twitter networks to show the effectiveness of the proposed joint label prediction model.

1 Introduction

Expert finding addresses the problem of identifying individuals who are knowledgeable in a given topic. Although most of the proposed algorithms for expert finding restrict their analysis to the documents and relations exist in a single environment[1], recent studies [2, 3] suggest that besides the degree of expertise, there are some other important factors, which should be taken into account for ranking of experts. These factors such the availability of an expert [4] and the authority of experts in their specialization area [5] are generally independent of the content of the documents and can be extracted from multiple sources of information. Experts' 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 experts. 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 integration of the DBLP and Twitter networks towards building such multi environment expert finding system.

P. Serdyukov et al. (Eds.): ECIR 2013, LNCS 7814, pp. 122–133, 2013.

Generally, integration of scientific and social networks is a challenging task because of the following reasons: Firstly, according to a recent research study [6], about 11% of people use nicknames in microblogs, which cannot be reached by the naive name matching. 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 [7].

In order to find the matching relationship between DBLP and Twitter networks, we use several name matching patterns to find high probable matching pairs in these networks. While these matching pairs are collected using a focused crawling mechanism, due to name ambiguity, a lots of collected pairs are not valid matches. Therefore, Our matching problem is reduced to find true matching pairs among the collected candidate pairs by the crawler. We use several features extracted from Twitter and DBLP profiles to train the state-of-the-art classifiers (e.g. logistic regression, SVM, decision tree, etc.). While these classifiers basically assume label independency between instances, However, in our matching 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. We consider two main types of dependencies between candidate pairs:

1) Common friend dependency: In many cases, scientific collaborators are also social friends. Thus, 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. 2) One-to-One matching dependency: Scientific networks (e.g. digital libraries) use sophisticated algorithms [7] and manual effort 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 DBLP profile is already determined as a true match for a specific Twitter profile, the probability of matching other Twitter profiles (for the same DBLP 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*. Using relational learning method, we simultaneously predict the label of dependent candidate pairs. Our experiments on an automatically generated test collection and five manually annotated topical test collections shows significant improvement in comparision with state of the art classification methods.

2 Related Work

Recent methods for expert finding [4, 5], consider heterogeneous sources of information to improve the quality of expert ranking. Smirnova and Balog [4]

considered geographical location of experts to rank them based on their accessibility, and Deng et al. [5] suggested to rank each expert based on his authority in the research communities. 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.

As the most similar research to our work, You et al.[6], proposed a method to integrate two networks of people namely, EntityCube [1] and Twitter networks. They addressed the problem of finding Twitter pages (i.e. social profile) of a group of related celebrities. They 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(i.e. SVM) to model this property.

Another related line of research to our work is relational learning. Some previous research [7–9], reported significant accuracy improvement of relational learning methods (e.g. collective learning) in comparison with independent learning methods in interdependent decision making problems. While our matching algorithm, 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).

3 Integration of Social-Scientific Networks

We divided the problem of social and scientific network integration into two sub problems. The first problem (i.e. selection) concerns finding those profiles in one network, which presumably have a corresponding profile in the other network and the second problem (i.e. matching) concerns the name disambiguation to find true matching profiles among some candidate profile pairs for matching.

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 (e.g. @sigir2011, @ecir2011, @siggraph_ic in Twitter) 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.

The crawler starts collecting profile of people who directly follow event profiles (i.e. seed profiles) and uses follow[2] relation between people to find new profiles. For each collected profile, it uses some name similarity patterns to find the

[1] `http://entitycube.research.microsoft.com/`

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

candidate scientific profiles for matching. If it cannot find any candidate for a given social profile, it will continue crawling from other paths. It continues until a predetermined number of candidate pairs is collected. Using several name matching patterns introduced in [6], output of the crawler is a set of matching profile pairs.

3.1 Matching Problem

The output of 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 among the set of collected candidate pairs. Using several discriminative features associated with each candidate pair, we can train a classifier to determine the label of each candidate pair.

Independent Label Prediction. Given a set of training instances $TrainSet = \{(x_1; t_1)...(x_n; t_n)\}$, we can use several indicative features associated with each candidate pair to train a classifier, 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 Twitter profile $s \in V_S$ and a DBLP profile $d \in V_D$, the classifier determines if the profile d is a valid match for s. We use the parametric form of logistic regression (as an independent classification model) to predict the label of each candidate pair $p(t_i|x_i)$:

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

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. The likelihood function is convex and has a unique global maximum which can be found numerically [10]. After learning the parameter θ, we can use equations 1 to predict the most probable label for a given test instance (i.e. a candidate pair of matching). As the baseline matching model, the classifier determines the label of each candidate pair independently and does not utilize various dependencies between candidate pairs of matching.

Candidate Pairs Label Dependence. Logistic regression as an independent label prediction model is a naive solution for our matching problem. In fact, In our matching problem, the label of each candidate pair is not independent of other pairs. We consider two cases of dependencies between candidate pairs.

First of all, if a common friend (in both Twitter and DBLP networks) exists for a candidate pair, 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 co-author in DBLP and s_j and s_l are Twitter friends. We refer to this type of dependency between candidate pairs as dependency *type 1*. Secondly, since DBLP network uses sophisticated algorithms and manual effort to disambiguate people names, we expect that in most cases each person has at most one profile in DBLP network. On the other hand, the majority of people have at most one profile in the Twitter 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*.

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 DBLP and Twitter profiles respectively. 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 DBLP profiles and $E_S = \{(s_l, s_n) | s_l, s_n \in V_S \wedge Follow(s_l, s_n)\}$ indicates the social tie between Twitter profiles. During selection phase, the focused crawler finds for each Twitter profile some few matching candidates in the DBLP network. We can indicate the set of candidate pairs by:

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

In order to model mentioned dependencies between candidate pairs, we 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: $V_{MDG} = \{(s_i, d_j) | s_i \in V_S, d_j \in V_D, (s_i, d_j) \in C_{SD}\}$. According to those three types of dependencies between candidate pairs, 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 *type1* 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 *type2* dependency between nodes of V_{MDG} is indicated using the edges in E_2 and it can be defined as $E_2 = \{((s_i, d_j), (s_m, d_n)) | s_i, s_m \in V_S \wedge d_j, d_n \in V_D \wedge s_i = s_m \wedge d_j \neq d_n\}$. The edges in E_3 represent the *type3* dependency between nodes of V_{MDG} and can be defined as $E_3 = \{((s_i, d_j), (s_m, d_n)) | s_i, s_m \in V_S \wedge d_j, d_n \in V_D \wedge s_i \neq s_m \wedge d_j = d_n\}$

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.

Relational classification is a natural solution for our joint label prediction problem. By definition[10], 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[11], 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 [11], we can estimate the joint conditional probability of a particular label assignment T given observed feature X as a normalized product of a set of non-negative potential functions. Although, each potential function can be an arbitrary non-negative function, but according to [10], 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:

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

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.

Using log-linear potential functions [10], each potential function ψ_1, ψ_2, ψ_3, ψ_4 is represented by weighted combinations of feature vectors in the following form:

$$\psi_1(x_i, t_i) = \sum_{m=1}^{M_1} \theta_m f_m(x_i, t_i) \qquad \psi_2(t_i, t_j) = \sum_{m=1}^{M_2} \alpha_m g_m(t_i, t_j)$$
$$\psi_3(t_i, t_j) = \sum_{m=1}^{M_3} \beta_m h_m(t_i, t_j) \qquad \psi_4(t_i, t_j) = \sum_{m=1}^{M_4} \zeta_m s_m(t_i, t_j)$$

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 method, 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 [12]. 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[10]) 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:

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

Although, due to loopy structure of *MDG*, exact inference is not applicable we can use Belief Propagation [10] to approximatly predict the most probable label assignement for a given (i.e. test instance) *MDG*. The *MDG* graphs resulting from the three cases of dependencies are usually not densely connected in real cases. Thus, 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 and the DBLP networks. To build these test collections, we use the crawler (described in section 3) to collect Twitter profiles and their corresponding candidate DBLP profiles.

The first testcollection (i.e. URL collection) is generated automatically by exact URL matching between homepage field of Twitter and DBLP. 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. According to the topic of each seed event introduced in 3, we categorized them into five main topics in computer science.[3]

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, 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 1 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.

[3] Five main topics related to the seed profiles. DB= Database, DM-IR= Data mining and Information Retrieval, HCI = Human Computer Interaction, OS = Operating Systems, SF= Software.

Table 1. Detailed statistics of the test collections. *URL* is the automatically generated test collection and other test collections are named by the abbreviations introduced in the page 8.

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 *type1* in *MDG*	28	8	27	9	38	31
Number of edges of *type2* in *MDG*	383	807	433	656	597	290
Number of edges of *type3* in *MDG*	132	515	201	308	353	205

4.2 Experiments Setup

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

To train the independent and joint classification models, we use five groups of features including 1) Twitter homepage URL features (2-features), 2) Twitter location feature (1-feature), 3) Twitter-DBLP name similarity features (5 features), 4) Twitter Description features (10 features) and 5) Twitter-DBLP crawling information features (10 features).

5 Results

In this section, an extensive set of experiments were conducted on the six test collections to address the following questions: 1) How good are the discriminative independent label prediction approaches compared with the SIMPLE heuristic method? 2) Can the prediction performance be improved by considering the dependency between the labels of the candidate pairs?

5.1 SIMPLE Heuristic Method versus Independent Label Prediction

In this section, we compare the matching performance of 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 2 contains the comparisons in precision, recall and F-score.

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

Table 2. SIMPLE method versus independent label prediction. Comparisons are based precision(P), recall(R) and on F-measure(F).

Collection/Method	Simple			Decision Tree			SVM			LR		
Measure	P	R	F	P	R	F	P	R	F	P	R	F
DB	0.460	0.944	0.619	0.879	0.693	0.775	0.826	0.743	0.782	0.891	0.732	**0.804**
DM-IR	0.242	0.908	0.382	0.693	0.674	0.683	0.652	0.730	0.689	0.671	0.752	**0.709**
HCI	0.420	0.875	0.569	0.650	0.620	0.635	0.680	0.590	0.632	0.760	0.615	**0.678**
OS	0.321	0.899	0.474	0.780	0.754	0.767	0.768	0.760	0.764	0.802	0.749	**0.775**
SF	0.261	0.902	0.405	0.715	0.699	0.707	0.699	0.699	0.699	0.726	0.737	**0.731**
URL	0.763	0.826	0.794	0.802	0.802	**0.802**	0.811	0.768	0.789	0.786	0.783	0.785

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. On the other hand, 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 these methods decrease the recall score substantially. It means that these methods tend to select only candidate pairs with 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, we compare the matching performance of the logistic regression method (as an independent label prediction model) with the joint label prediction method trained on the dependency *type 1*, *type 2*, *type 3* and the combination of them. Table 3 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 3. 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***

Table 3 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, 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. In other words, this method improves the recall score but retains the precision in the same level in comparison with logistic regression. In fact, 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. According to this experiment, 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.

6 Conclusions and Future Work

In this paper, we designed a focused crawler to collect high probable matching profile pairs in the DBLP and the Twitter networks. The network integration problem is then reduced to finding true matching pairs among these collected candidate pairs. We introduced a joint label predication method to predict the label of candidate pairs simultaneously. Our experiments indicate that the joint label prediction method can improve the F-score of matching up to 8.7% in comparison with the independent classification methods.

Acknowledgement. This work was in part supported by a grant from Iran telecommunication research center (ITRC). We would like to acknowledge Mr. Hessameddin Akhlaghpour for his helps in this research project.

References

1. Balog, K., Azzopardi, L., de Rijke, M.: A language modeling framework for expert finding. Inf. Process. Manage. 45(1), 1–19 (2009)
2. Serdyukov, P.: Search for expertise: going beyond direct evidence. PhD thesis, Enschede (June 2009)
3. Fang, Y., Si, L., Mathur, A.P.: Discriminative probabilistic models for expert search in heterogeneous information sources. Inf. Retr. 14, 158–177 (2011)
4. Smirnova, E., Balog, K.: A User-Oriented Model for Expert Finding. In: Clough, P., Foley, C., Gurrin, C., Jones, G.J.F., Kraaij, W., Lee, H., Mudoch, V. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 580–592. Springer, Heidelberg (2011)
5. Deng, H., King, I., Lyu, M.R.: Enhanced models for expertise retrieval using community-aware strategies. IEEE Transactions on Systems, Man, and Cybernetics, Part B 42(1), 93–106 (2012)
6. 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)
7. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Transactions on Knowledge Discovery from Data 1(1) (2007)
8. Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: UAI, pp. 485–492 (2002)
9. Fang, Y., Si, L., Mathur, A.P.: Discriminative graphical models for faculty homepage discovery. Inf. Retr. 13(6), 618–635 (2010)
10. Sutton, C., McCallum, A.: An Introduction to Conditional Random Fields for Relational Learning, pp. 93–128. MIT Press (2006)
11. 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 2001, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco (2001)
12. McCallum, A.: Efficiently inducing features of conditional random fields. In: Nineteenth Conference on Uncertainty in Artificial Intelligence, UAI 2003 (2003)