Probabilistic Entity Linkage for Heterogeneous Information Spaces

Ekaterini Ioannou, Claudia Niederée, and Wolfgang Nejdl

L3S Research Center/Leibniz Universität Hannover
Appelstr. 9a, 30167 Hannover, Germany
{ioannou,niederee,nejdl}@L3S.de

Abstract. Heterogeneous information spaces are typically created by merging data from a variety of different applications and information sources. These sources often use different identifiers for data that describe the same real-word entity (for example an artist, a conference, an organization). In this paper we propose a new probabilistic *Entity Linkage* algorithm for identifying and linking data that refer to the same real-world entity.

Our approach focuses on managing entity linkage information in heterogeneous information spaces using probabilistic methods. We use a Bayesian network to model evidences which support the possible object matches along with the interdependencies between them. This enables us to flexibly update the network when new information becomes available, and to cope with the different requirements imposed by applications build on top of information spaces.

Keywords: entity linkage, data integration, metadata management.

1 Introduction

Many applications rely on rich *information spaces* - collections of data from a variety of different applications and information sources. Information spaces are found in various systems of different research areas. For example Semantic Web with applications analyzing social networks such as identifying conflicts of interests [1], or researcher's influence in a community [16]. Also, Personal Information Management (PIM) with systems such as Beagle^{++} [8], and Haystack[1], as well as information integration approaches for Information Systems [19]. The success of these applications depends on a clear picture about the data and their relationships that the underling information spaces provide.

When compiling information spaces from heterogeneous sources, data referring to the same real-world entity (for example to a researcher or an artist, a conference or an organization) often use different identifiers. Different attribute sets (e.g. 'hasName', 'author'), the use of naming variants (e.g., 'Wolfgang Nejdl', 'Nejdl W.'), and most importantly, the lack of a global coordination for identifier assignment, forces each source to create and use its own identifiers.

[1] http://haystack.lcs.mit.edu/

Furthermore, each source describes entities in a way most adequate for its purpose. A publication will describe a person using name and affiliation, whereas an email will use the email address. It is the goal of *Entity Linkage* to identify data describing the same real-world entity, and link their corresponding identifiers.

In order to cope with uncertainties inherent in entity linkage, in this paper we propose a new probabilistic entity linkage algorithm, which is able to compute the probability for each possible match between data according to the evidences currently available in the information space. Our approach addresses the special characteristics and resulting **challenges** of entity linkage in information spaces for PIM:

– We can distinguish two main directions for identifying possible matches in information spaces. The first is based on observing similarities between text values of the data participating in a potential match. The second direction relies on identifying relationships between the data. $\xmapsto{C1}$ Entity linkage should be able to follow both directions, and incorporate the observed evidences.
– Information spaces for PIM constantly change and evolve through interactions of the user with his/her desktop. This changes the information available to the entity linkage algorithm. $\xmapsto{C2}$ An entity linkage solution should support incremental computation and adaptation of entity matching information. Also, since entity linkage results are never finalized, the original data of the information space should not be modified.
– A wide variety of applications can be executed on top of integrated information spaces. Each application might have different requirements for the entity linkage solution. For example, one application might need only certain matches, whereas other applications might accept uncertain matches based on only few evidences. $\xmapsto{C3}$ Matches should be accompanied with a metrics, indicating the belief we have that the corresponding identifiers refer to the same real-world entity, based on the evidences in the current information space.

The strong interconnections in information spaces are a valuable source of entity matching evidence. Previous approaches operating in such information spaces, such as [10] and [3], did not restrict themselves to entity attributes but also systematically exploit the context of the entity, taking into account associations with other entities. In particular, [10] uses association properties of entities in combination with normal attributes for computing record linkage. Also, [3] exploits the link structure of Web pages about persons as an indicator for entity (person) relationships. Our approach is most similar to the approach presented in [10], which uses entity context in the form of relationships and propagation of matching evidence information. However, we go further by also addressing the other two PIM characteristics described above, while achieving comparable precision and recall performance.

The main contribution of our work is an innovative entity linkage algorithm, that addresses all three challenges listed, by: (i) clearly separating data of the information space with data representing decisions for matches, (ii) enabling incremental update of matches, when new information becomes available in the information space, and (iii) associating each match with a probability indicating the belief (confidence) we have for the existence of the specific match. Our algorithm receives entity metadata and computes matching probabilities by constructing and maintaining a Bayesian network with matching evidences. As a further contribution, we introduce and explain the problem as it appears in heterogeneous PIM information spaces, as a set of requirements to be addressed by our algorithm.

The rest of the document is organized as follows. Section 2 gives an overview over related work. Section 3 provides a formal formulation of the entity linkage problem for PIM, and Section 4 presents our algorithm. Section 5 presents our evaluation experiments, showing good precision and recall on real-life test collections. Finally, Section 6 concludes and discusses future extensions.

2 Related Work

Variants of the problem we address in this paper have been investigated in different research areas. Traditionally, the database community proposed algorithms to detect database tuples that refer to the same real-world entity [20,12]; a problem known as record linkage, data integration, and merge/purge. More recently, algorithms in the data mining community propose identifying real-world entities as a way to perform data cleaning, and clustering tasks. These algorithms try to identify the data that describe the same real-world through interconnections. Merging objects is done by calculating the *distance* between data that possibly describe the same real-world entity [5,6], or computing the interconnection strength of their alternative connection paths [15,14].

The most relevant algorithms related to our approach are the ones that identify entities through interconnections between objects found in a given dataset. Ananthakrishna et al. [2] exploit dimensional hierarchies to detect fuzzy duplicates in dimensional tables. The hierarchies are build by following the links between the data of one table to data of other tables. Entities are matched when the information along these generated hierarchies is found similar. The most recent algorithm, motivated by a Personal Information Management scenario, is the *Reference Reconciliation* algorithm by Dong et al. [10]. The authors use interconnections to identify and merge data that possibly describe the same real-world entity. Information about these merges is propagated into the rest of the dataset (reconciliation propagation), along with the exchange of information between the two merged references (reference enrichment). A modified version of this algorithm [1] is used for detecting conflict of interests in paper reviewing processes.

The DBLP system[2] faces a similar problem with author names. To solve it, they construct a co-author graph (nodes show authors, links show common publications). Merging authors is done using *edit distance* algorithms, based on

[2] http://dblp.uni-trier.de/

comparisons such as Levenshtein distance and soundex. The TAP system [11] uses a *Semantic Negotiation* process through which the common descriptions (if any) between the different resources are identified. These common descriptions are then used to create a unified view of a given data set. Swoosh [4] is another related system. Here, the authors focus on identifying the different properties that affect the efficiency of such algorithms, and introduce different approaches to address the possible combinations of the found properties.

3 Problem Formulation

Entity linkage is concerned with relations of objects described in heterogeneous information spaces, with corresponding entities existing in the real-world. We start with an information space consisting of metadata describing resources, for example emails or publications. The metadata include descriptions of *objects* such as persons or conferences. Objects that refer to the same real-world entities will often have different identifiers. Relating objects of the information space to entities of the real-world allows us to discover the objects which should have the same identifier. We will give a formal definition of this problem in the following paragraphs, and provide a summary of the used notation Table 1.

Table 1. A summary of the algorithm's notation

Symbol	Description
$\mathcal{D}$	The heterogeneous information space
r	A Resource (for example an email, a publication)
$\mathcal{M}(r)$	The metadata describing resource r
$e(t_i, t_j, \phi)$	Evidence showing the similarity of t_i with t_j, as given by function ϕ
$rep_{r,k}$	The representation of object k in resource r
$\mathrm{d}(rep_{r,k})$	The entity mapping for representation $rep_{r,k}$
$\mathrm{P}(\mathrm{d}(rep_{r_i,k})=\mathrm{d}(rep_{r_j,m}))$	A match between two entity mappings

Definition 1. *The information space $\mathcal{D}$, on which we execute our algorithm, is a set of metadata describing resources $\mathcal{R}_D$. It is defined as $\mathcal{D} = \{\mathcal{M}(r_i) \mid r_i \in R_{\mathcal{D}}\}$, where $\mathcal{M}(r_i)$ denotes the metadata describing an individual resource r_i.*

Definition 2. *Metadata for resource r_i is represented as a set of tuples t. These tuples describe the resource along with the objects found in the context of the resource. It is defined as $\mathcal{M}(r_i) = \{\, t_j \mid j \leq sizeof(t_{r_i})\}$.*

In this paper, we assume that $\mathcal{D}$ complies with the RDF data model, and thus a tuple t_i is a triple of the form $\langle u, p, o\rangle$. Symbol u denotes a URI, p denotes a property, and o an RDF-object which can either be a literal, or a URI. URIs are used as identities of resources. Ideally, the same identity would be used whenever describing the same object. However, since the URI assignment is not globally coordinated, multiple URIs are used for single real-world entities (see

(a)

Metadata for publ. #77 ($M(r_{77})$)**={...,**
⟨ file:///P77, type, 'publication'⟩,
⟨ file:///P77, title, ... ⟩,
⟨ file:///P77/a_1, name, `K. Marriott`⟩,
⟨ file:///P77/a_2, name, `P. J. Stuckey`⟩}

Metadata for publ. #127 ($M(r_{127})$)**={...,**
⟨ file:///P127/a_1, name, `'Marriott, K'`⟩,
⟨ file:///P127/a_2, name, `'Sndergaard, H'`⟩,
⟨ file:///P127/a_3, name, `'Kelly, A'`⟩}

Metadata for email #128 ($M(r_{128})$)**={...,**
⟨ file:///E128/to_1, name, `Kelly A.` ⟩,
⟨ file:///E128/from, name, `Sndergaard H.`⟩,
⟨ file:///E128/to_2, name, `Stuckey P.`⟩}

(b)

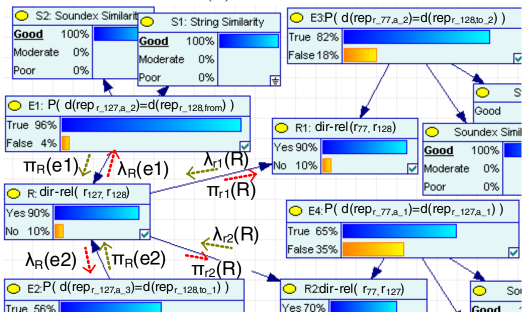

Fig. 1. (a) Metadata for desktop resources, and (b) corresponding Bayesian network

[7] for more details). Therefore, even if URIs provide an appropriate formalism for unique identifiers, the entity linkage problem is still present.

Definition 3. *Let object representation $rep_{r,k}$ denote the subset of the tuples from $\mathcal{M}(r)$ which refer to/describe the same object k in r. It is defined as $rep_{r,k} = \{t \mid t = \langle id_k, p, o\rangle \vee t = \langle s, p, id_k\rangle\}$, where id_k is the local identifier (URI) assigned to object k.*

Figure 1 (a) shows some example metadata describing three desktop resources together with several representations for the contained objects. Two examples of object representation are:
– $rep_{r_{77},a\_1} = \{\langle$ file:///P77/a_1, name, K. Marriott$\rangle\}$, and
– $rep_{r_{127},a\_1} = \{\langle$ file:///P127/a_1, name, 'Marriott, K'$\rangle\}$

Definition 4. *The entity mapping $d(rep)$ semantically maps an object representation rep to the corresponding real-world entity.*

Obviously, $\mathcal{D}$ may consist of different representations $rep_{r_i,k}, \ldots, rep_{r_j,m}$ which semantically refer to the same real-world entity, i.e., $\text{d}(rep_{r_i,k}) = \ldots = \text{d}(rep_{r_j,m})$. The challenge is that the mapping d is not known to the entity linkage algorithm. To overcome this, we have to compute the *match*, $\text{P}(\text{d}(rep_{r_i,k}) = \text{d}(rep_{r_j,m}))$, for pairs of object representations. Each match expresses the probability with which specific entity mappings refer to the same real-world entity.

Definition 5. *The probability of each match depends on the supporting information in $\mathcal{D}$. We represent each such supporting information item with an evidence $e(t_i, t_j, \phi)$, where t_i and t_j denote metadata tuples of the entity mappings in our match, and ϕ the function which reports similarity between t_i and t_j.*

Going back to our example, the previous representations correspond to the real-world entities $\text{d}(rep_{r_{77},a\_1})$ and $\text{d}(rep_{r_{127},a\_1})$. To decide whether these correspond to the same entity we apply our algorithm and calculate the probability of match $\text{P}(\text{d}(rep_{r_{77},a\_1}) = \text{d}(rep_{r_{127},a\_1}))$, based on evidences for this match from the information space. An evidence for this match can be obtained using a

string similarity function. For example:
$e(\langle\text{...P77/a\_1, name, K. Marriott}\rangle, \langle\text{...P127/a\_1, name, `Marriott, K'}\rangle, \text{StringSim})$.

4 The Entity Linkage Algorithm

4.1 A Brief Reminder of Bayesian Networks and Inference

Bayesian networks [18,13] are probabilistic graphical models for reasoning under uncertainty, using cause-effect relationships modeled as a directed acyclic graphs. Each node in the graph represents a variable with two or more possible states. Each edge from parent node X to child node Y, represents a cause-effect relationship with X being the cause and Y the effect, whenever the state of Y is directly influenced by the state of X. Each node X is accompanied with a *local probability distribution* $P(X \mid U_1, .., U_m)$, showing the conditional probability of all states in X given the states of its parents U_1,...,U_m. Nodes without parents are associated with an unconditional $P(X)$, representing prior probabilities.

Bayesian networks represent the *joint probability distribution* over all variables, defined as the product of all local probability distributions, as follows:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i \mid parent(X_i)),$$

where $P(X_i \mid parent(X_i))$ corresponds to the local probability distribution of node X_i, and $parent(X_i)$ to the parent nodes of X_i.

Bayesian networks successfully determine the conditional probabilities of cause nodes based on the current probabilities of the effect nodes, a task called *probabilistic inference*. Given any new effects (evidences), probabilistic inference recomputes the probability of the cause nodes which are responsible for these effects. One well-known algorithm for probabilistic inference is *message-passing* by Pearl [18]. Pearl's algorithm is iterative, and in each iteration calculates the belief of a node based on messages exchanged by the node X with its parents $U_1, ..., U_m$ and its children $Y_1, ..., Y_n$. When node X is activated, and receives all messages $\pi_X(U_i)$ from its parents, and $\lambda_{Y_j}(X)$ from its children, it calculates its own belief as:

$$BEL(X) = \alpha\lambda(X)\pi(X), \quad \text{where } \alpha \text{ is a normalization constant,}$$
$$\pi(X) = \sum_{U_1,...,U_m} P(X|U_1, ..., U_m) \prod_{i=1}^{m} \pi_X(U_i), \text{ and } \lambda(X) = \prod_{j=1}^{n} \lambda_{Y_j}(X) \; .$$

After calculating its belief, node X computes and sends new messages $\lambda_X(U_i)$ to its parents, and $\pi_{Y_j}(X)$ to its children. These messages are:

$$\pi_{Y_j}(X) = \alpha\, \pi(X) \prod_{k \neq j} \lambda_{Y_k}(X)$$
$$\lambda_X(U_i) = \sum_{X} \lambda(X) \sum_{U_k : k \neq i} P(X|U_1, ..., U_k) \prod_{k \neq i} \pi_x(U_k)$$

4.2 Structure of Our Bayesian Network

The goal of our algorithm, is to compute the probability of each match. We perform this task using a Bayesian network constructed from information related to the matches. The information is encoded using the following node types:

Entity Nodes. These nodes represent a match, e.g., P(d(rep_{r_{77},a_1}) = d(rep_{r_{127},a_1})). As explained in Section 3, $\mathcal{D}$ does not have the information to directly compute matches, thus we can not specify the states of entity nodes. The probability of their states is computed through probabilistic inference based on the cause-effect relationships in the network.

Evidence Nodes. These nodes represent evidences for entity nodes. It is the first type of nodes we use to represent evidence for entity nodes. Each evidence node represents an evidence $e(t_i, t_j, \phi)$, with t_i and t_j being tuples from the two representations constructing the match. Evidence nodes form the prior probabilities in our network, since they have no parent nodes upon which they depend. The unconditional probability distribution is determined by the similarity function ϕ used for creating them. In our current approach, we rely on the comparison of literals from t_i and t_j to measure compatibility between the corresponding properties. An extension of our approach for operation in more heterogeneous contexts is the inclusion of a comparison of the properties themselves and the inclusion of these similarities into the aggregation of the matching probabilities.

Direct-Relation Nodes. These nodes rely on the fact that two resources are related when their descriptions contain the same object. It is the second type of nodes that influence the states of entity nodes. For example, match P(d(rep_{r_{77},a_1}) = d(rep_{r_{127},a_1})) implies a relation between resources r_{77} and r_{127}, which we encode in the direct-relation node dir-rel(r_{77}, r_{127}). Finding evidences for more shared objects between these resources (e.g., additional common authors) increases the belief for the relation of r_{77} with r_{127}, and consequently the belief in the corresponding matches. Direct-relation nodes are the effect we can observe for entity nodes and for deductive-relation nodes (explained bellow). The local probability distribution of their states is directly influenced by their relationships with these node types.

Deductive-Relation Nodes. These nodes represent the indirect relation between two resources, inferred by combining the information of two nodes, either direct-relation or deductive-relation. Combining direct-relation nodes dir-rel(r_{77}, r_{127}) and dir-rel(r_{77}, r_{128}) for example, implies a new relation between r_{127} with r_{128} (due to the common resource r_{127}), which we encode in deductive-relation node del-rel(r_{127}, r_{128}).

An important aspect of the Bayesian network is the cause-effect relationships between the nodes. We have already explained these together with the different types of nodes. Table 2 gives a summary.

Table 2. The possible cause-effect relationships used in our Bayesian network

Effect Nodes:		(1) Evidence	(2) Dir.-Rel.	(3) Ded.-Rel.
Cause	(1) Entity	√	√	
Nodes:	(2) Ded.-Rel.		√	√

4.3 Incremental Computation of the Network

To compute the probability of matches we collect evidence, positive and negative. Then, we calculate the probability of matches by constructing a Bayesian network, modeling matches and related evidence. Starting point for this computation is an incrementally growing metadata set, which are added to $\mathcal{D}$. We have to update the Bayesian network incrementally, after addition of new metadata.

Upon addition of a new set of metadata, the algorithm performs the following four steps: (a) Process the object representations contained in the metadata added. By comparing the new representations with previous representations, the algorithm identifies similarities, and updates the network with new evidence and entity nodes. (2) Create direct-relation nodes to represent the effects we observed which could cause these new entity nodes. (3) Analyze the updated network and generate new information about the relation of resources, represented using deductive-relation nodes. (4) Perform probabilistic inference on the Bayesian network, and generate the updated probabilities for the matches. The remaining paragraphs of this section describe these steps in more detail.

Step 1 - Adding Entity & Evidence Nodes. The new metadata contain one or more object representations. In the first step, the algorithm updates the Bayesian network with new evidences generated using these representations. We start with similarity computations to identify resemblance between tuples from the new representation $rep(r_{new}, k)$ with compatible tuples from the existing representations $rep(r_{exists}, m)$. In the current version of our algorithm, similarities are detected using two functions. The first algorithm is *String Similarity*, detecting string resemblance between literals of tuples[3]. The second algorithm is *Soundex Similarity*, which detects the resemblance in pronunciation between literals[4]. Whenever similarity is above a given threshold, we consider it as evidence for the match P(d($rep(r_{new,k})$)=d($rep(r_{exists,m})$)).

An evidence node is created for each similarity identified. Since the current version of our algorithm includes two similarity algorithms, we create one or two evidence nodes for each match. All evidence nodes have three states, *Good*, *Moderate*, and *Poor*, which we set based on computed similarity.

An entity node is created to represent the identified match P(d($rep(r_{new,k})$) = d($rep(r_{exists,m})$)), if such node does not yet exists. The relation between the newly created evidence nodes with the entity node is represented by introducing cause-effect relationships. All entity nodes have two possible states, *Exists* to

[3] For String Similarity we use the JaroWinkler method from the SecondString API [9].
[4] For Soundex Similarity we use the Apache Codec API.

indicate that the corresponding match exists, and *Exists_Not* to indicate that the match does not exist. The probabilities of these states are computed by probabilistic inference.

Step 2 - Adding Direct-Relation Nodes. Direct-relation nodes represent the observed effect that entity nodes could cause. They are created using only information from the matches. For each match P(d($rep(r_{new,k})$)=d($rep(r_{exists,m})$)) we extract its resources, and use them to create a direct-relation node del-rel(r_{new},r_{exists}), if this node does not yet exist. If there is more than one match referring to the same two resources, we represent this through a cause-effect relationships created between the entity nodes and the corresponding direct-relation node. The direct-relation nodes have two possible states, *Yes* to indicate that the two resources are related, and *No* to indicate that the resources are not related. The probabilities of these states are again computed by probabilistic inference.

Step 3 - Adding Deductive-Relation Nodes. This step analyzes the current status of the network to extract indirect relations between the resources. The underlying idea is similar to the one represented by the direct-relation nodes. To identify possible indirect relations, our algorithm inspects the direct-relation and deductive-relation nodes. Each node is considered as a transitive, binary relation (b-relation) between the two participating resources. For example, dir-rel(r_{77},r_{127}) corresponds to b-relation (r_{77},r_{127}), and dir-rel(r_{77},r_{128}) to b-relations (r_{77},r_{128}). The algorithm extracts more relations by transitively combining b-relations. For example, b-relation (r_{127},r_{127}) is the transitive combination of our two previous b-relations. We encode the new b-relation using a ded-rel node, for example del-rel(r_{127},r_{127}).

Since computing transitive b-relations is a recursive process, we need an appropriate stopping criterium. In the current version of our algorithm, we enforce a fixed ratio between entity nodes and deductive-relation nodes. This approach allows us handle specific characteristics possibly present in $\mathcal{D}$. If $\mathcal{D}$ contains only few matches, the algorithm will be forced to search for evidence by incorporating many deductive-relation nodes. On the other hand, if $\mathcal{D}$ contains a relatively big number of matches, the algorithm will include only a small subset of them, enough to increase the belief for the specific node without overloading the network with nodes.

Step 4 - Updating the Matches. Once the network is updated with nodes representing new matches and evidences, we need to recalculate the probability for the states of each node. This task is performed through probabilistic inference which updates all nodes according to the current status of the network. To minimize the time needed for doing this, we execute probabilistic inference only on the newly added nodes and nodes related to them.

As explained in Section 4.1, computing the probability of a node requires information from its neighbor nodes. Computed results are propagated back to the neighbor nodes to allow them to recompute their probability. For example, consider node R from the Bayesian network of Figure 1 (b). Once node R is activated, and receives messages $\lambda_{r_1}(R)$, $\lambda_{r_2}(R)$, $\pi_R(e1)$, and $\pi_R(e2)$ from its parent and children nodes, it computes: (i) its own belief as $BEL(R) = \alpha\lambda(R)\pi(R)$ (marked as eq. 1 in the following equation list), and (ii) new messages to send to its parent nodes (eq. 2), and children nodes (eq. 3). These messages are as follow:

$$\lambda(R) = \lambda_{r_1}(R)\lambda_{r_2}(R), \text{ and } \pi(R) = P(R|e1, e2)\pi_R(e1)\pi_R(e2) \tag{1}$$

$$\lambda_R(e1) = P(R|e1, e2)\pi_R(e2)\lambda(R) \tag{2}$$

$$\pi_{r2}(R) = \pi(R)\lambda_{r1}(R) \tag{3}$$

The message computation in this example shows the main benefit of using cause-effect relationships between nodes. Although node $e1$ is not directly connected to node $e2$, the algorithm is able to propagate information from one node to the other, through their cause-effect relationships with node R. Consequently, a high belief of node $e2$ affects the belief of node R (eq. 1), which is reflected in the message node R sends to node $e1$ (eq. 2). Finally, node $e1$ is affected when it recomputes its belief using the message sent to it by node R.

Entity Linkage information for $M(r_{127})$:
⟨ el:///E1, object_rep, file:///P127/a_2 ⟩,
⟨ el:///E1, object_rep, file:///P128/from ⟩,
⟨ el:///E1, belief, 0.96 ⟩, ...

Fig. 2. Part of the entity linkage information generated by our algorithm, for the metadata of Figure 1

After executing probabilistic inference, we have an updated set of matches that reflect the metadata present in the information space. Different representations of the results matches are possible (i.e., include or do not include the probability of each match), and the selected representation depends on the needs of the specific system. Figure 2 shows one possible representation for part of the results of the metadata from Figure 1. In this example, additional metadata are generated to represent each match in the Bayesian network using the corresponding object representations and belief.

5 Experimental Evaluation

We evaluated our approach using a JAVA implementation of the entity linkage algorithm, including all features we described in the previous sections. For performing probabilistic inference[5]on the Bayesian network we used the *jSMILE API*[6], and for creating a database to store internal information we used *MySQL* 5.0[7]. The following paragraphs present the effectiveness of our algorithm on two datasets, the Cora and a PIM dataset.

[5] For efficiency reasons we use the 'Backward simulation' algorithm; a modified version of Pearl's algorithm that performs approximate inference.

[6] `http://genie.sis.pitt.edu/`

[7] `http://www.mysql.com/`

5.1 Cora Dataset

The *Cora dataset*[8] is a collection of publications collected from CiteSeer. Each publication contains title and author names, using different forms for the names (e.g., 'J. Antonisse', 'Antonisse , H. J.', 'Antonisse', 'Jim Antonisse'). The dataset was manually processed to accompany each publication author with an identifier that indicates the corresponding real-world entity.

We processed the Cora dataset and converted each publication into RDF triples. Our process generated 14392 triples describing title and authors for 1563 resources (publications). A total of 2882 triples described authors, with 9768 matches between these authors. Following the definitions of Section 3, we use as object representation rep_{A_i,C_k}, the triples describing author A_i as given in the triples generated for publication C_k. The task of our algorithm is then to compute the probability of entity mappings of author A_i from publication C_k with author A_j from publication C_n, represented by P(d(rep_{A_i,C_k})=d(rep_{A_j,C_n})).

The goals of our Cora dataset experiments were twofold: (i) evaluate the effectiveness of our algorithm in identifying the entities, and (ii) compare the effectiveness of our algorithm with the one given by the basic similarity functions we use for generating the evidence nodes. We measured effectiveness, as usual in information retrieval, by computing precision and recall. These measures were calculated in respect to the actual real-world entities, as specified by the unique identifier given for the authors of each publication in the Cora dataset.

We executed the experiments, by adding the triples generated from the Cora dataset incrementally into an information space, which uses our algorithm for entity linkage. After adding triples for 100 publications, we performed probabilistic inference on the Bayesian network generated by our algorithm. The following table shows the number of the matches that correspond to the different numbers of publications.

Publications	1000	1100	1200	1300	1400	1563
Matches	4129	4620	5050	6036	7337	9774

Entity Linkage Effectiveness. Figure 3 shows the plots for precision and recall under different probability thresholds, for several publications groups. The plots do not include groups that contain less than 1000 publications because the number of the corresponding matches is too small. Small values of the probability threshold (θ <0.4) are not included in the plots since the results are similar to θ=0.4.

As shown in Figure 3, our algorithm is able to maintain the same values for precision and recall for the different probability thresholds. For lower probability thresholds (i.e., θ=0.4, and θ=0.5) we see that recall is very high and precision is already quite satisfactory (around 0.9). Moving toward higher probability thresholds (i.e. θ=0.6, θ=0.7,) we see precision values increasing and, as expected, decreasing recall values. Precision does not 'automatically' increase with groups that have more publications —more data, and thus entities are available— but rather reflects our belief for the entities in the current data.

[8] We used the version from `http://www.cs.umd.edu/~indrajit/ER/index.html`

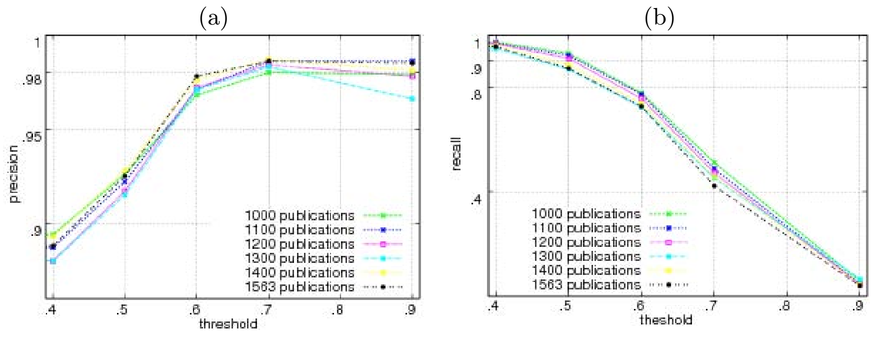

Fig. 3. (a) Precision, and (b) Recall vs. different thresholds

The results of these plots follow exactly the behavior explained in the analysis of our algorithm. It is clear that external algorithms are able to control the precision/recall of the entities by selecting an appropriating value of the probability threshold. For example, an application that needs only very certain matches will choose a high probability threshold, whereas an application that accepts uncertain matches a lower.

We also used our Cora dataset experiments to compare with previous approaches described in the literature. The authors of [10] reported precision 0.994 with recall 0.985, the authors of [17] had precision 0.842 with recall 0.909. To compare these numbers with our results, we considered only matches generated by our algorithm that exceed a preselected low probability threshold (e.g., θ=0.5). As shown in our two plots, these matches have high precision and high recall, similar to the ones given by these other algorithms. Our algorithm offers two additional advantages: (i) identified matches do not alter original metadata, (ii) our algorithm is able to further classify these matches according to the belief we have for their existence.

Comparison with basic similarity functions. In this experiment we performed a comparison of the effectiveness of our algorithm with the basic similarity functions used for generating evidence nodes. The algorithms we considered were Soundex Similarity and String Similarity, as described in Section 4.3.

Table 3 shows precision and recall values given by the two similarity functions on different publications groups. In all cases we assume that the real-world entities are the ones those probability is above threshold 0.7. Our evaluation shows our entity linkage clearly outperforms the effectiveness of the basic similarity functions.

Table 3. Precision/Recall of the entity linkage, and the basis similarity functions we used for generating the evidence nodes (θ=0.7)

Publications	Entity Linkage	String Similarity	Soundex Similarity
200	0.219/0.969	0.892/0.081	0.482/0.362
400	0.218/0.977	0.422/0.065	0.246/0.346
600	0.358/0.982	0.329/0.05	0.181/0.220

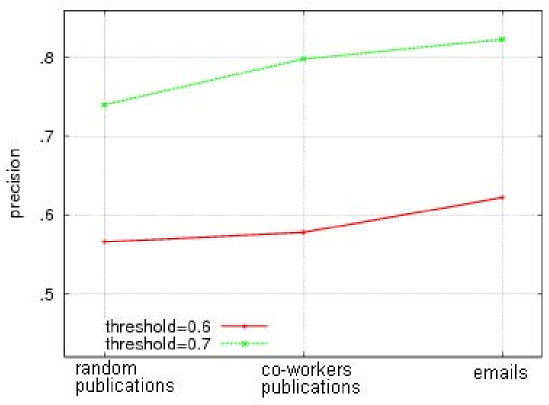

Fig. 4. (a) Precision, and (b) Recall vs. different thresholds

5.2 PIM Dataset

As a second dataset for evaluating our algorithm we use metadata generated in a personal information management environment, for desktop resources. As there is no publicly available PIM dataset, we created a suitable collection of metadata by simulating the behavior of a PIM application. Our PIM dataset included metadata describing desktop resources from three groups:

- The first group contains publications randomly selected from the DBLP system, to simulate arbitrary publications downloaded from the Web. This group resulted in metadata describing 700 imported resources, with 1326 triples corresponding to authors.
- The second group contains publications imported into the PIM environment from the DBLP system for which one of the authors is our co-worker at L3S. The results of this import were metadata for 250 resources, with 480 triples describing authors.
- The third group contains personal emails from one of the author's email client. Our goal was to identify and link authors from the publications with the corresponding person sending emails. The entity linkage problem in this case is somehow limited since persons are usually accompanied with email addresses which can act as unique identifiers for them. For this reason, we applied our entity linkage algorithm only on the existing email address, person name pairs. To capture the connections between these people, we randomly selected a small portion of available emails. The result was metadata for 200 resources, with 400 triples describing people. This group contains the most heterogeneous data, since each person has various email addresses and name variants.

We evaluated our algorithm on this PIM dataset, by adding metadata incrementally into an information space, which uses our algorithm for entity linkage.

We performed probabilistic inference on the Bayesian network generated by our algorithm, after adding the metadata of these three groups. Figure 4 shows the precision of the results generated by our algorithm. As shown, adding emails does not reduce the precision of the generated results, which is what we would have excepted since the emails contain the most heterogeneous data. As such, the results indicate that our algorithm is able to handle the heterogeneous instances of persons referenced in emails, and successfully link them with the author instances gained from the publications.

6 Conclusions

In this paper, we addressed the problem of identifying and linking heterogeneous data referring to the same real-world entity. This problem appears in a variety of situations, where we have to merge and integrate heterogeneous data from different information sources. Our algorithm uses a Bayesian network to explicitly model evidences supporting possible matches between different references, along with interconnections between these matches. The algorithm runs incrementally and does not modify existing data. Our evaluations showed that our algorithm successfully achieves our goal of efficiently and effectively linking data in heterogeneous information spaces.

We are currently investigating several directions for additional improvements and extensions of our entity linkage algorithm. First, we want to continue our experiments with other data sets and information sources. This will also include more general scenarios including unknown data schemes and data extracted from the Web. Finally, we will examine the use of other similarity functions for generating evidence nodes, and investigate the possibility of using different similarity functions for different data scenarios.

Acknowledgements

This work was supported by the NEPOMUK project, funded by the European Commission under the 6th Framework Programme (IST Contract No. 027705).

References

1. Aleman-Meza, B., Nagarajan, M., Ramakrishnan, C., Ding, L., Kolari, P., Sheth, A.P., Arpinar, I.B., Joshi, A., Finin, T.: Semantic analytics on social networks: experiences in addressing the problem of conflict of interest detection. In: WWW 2006 (2006)
2. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: VLDB (2002)
3. Bekkerman, R., McCallum, A.: Disambiguating web appearances of people in a social network. In: WWW 2005 (2005)
4. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widomr, J., Jonas, J.: Swoosh: A generic approach to entity resolution. Technical report, Stanford InfoLab (2006)

5. Bhattacharya, I., Getoor, L.: Deduplication and group detection using links. In: Workshop on Link Analysis and Group Detection, ACM SIGKDD 2004 (2004)
6. Bhattacharya, I., Getoor, L.: Iterative record linkage for cleaning and integration. In: DMKD (2004)
7. Bouquet, P., Stoermer, H., Mancioppi, M., Giacomuzzi, D.: OkkaM: Towards a Solution to the "Identity Crisis" on the Semantic Web. In: Italian Semantic Web Workshop, SWAP (2006)
8. Brunkhorst, I., Chirita, P.A., Costache, S., Julien Gaugaz, E.I., Iofciu, T., Minack, E., Nejdl, W., Paiu, R.: The beagle^{++} toolbox: Towards an extendable desktop search architecture. In: Semantic Desktop Workshop, ISWC (2006)
9. Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string distance metrics for name-matching tasks. In: Workshop on Inf. Integration on the Web (2003)
10. Dong, X., Halevy, A.Y., Madhavan, J.: Reference reconciliation in complex information spaces. In: SIGMOD Conference (2005)
11. Guha, R.V., McCool, R.: Tap: a semantic web platform. Computer Networks (2003)
12. Hernández, M.A., Stolfo, S.J.: Real-world data is dirty: Data cleansing and the merge/purge problem. Data Min. Knowl. Discov. (1998)
13. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer, New York (2001)
14. Kalashnikov, D.V., Mehrotra, S.: Domain-independent data cleaning via analysis of entity-relationship graph. ACM Trans. Database Syst. (2006)
15. Kalashnikov, D.V., Mehrotra, S., Chen, Z.: Exploiting relationships for domain-independent data cleaning. In: SDM (2005)
16. Li, J.-Z., Tang, J., Zhang, J., Luo, Q., Liu, Y., Hong, M.: Eos: expertise oriented search using social networks. In: WWW (2007)
17. Parag, Domingos, P.: Multi-relational record linkage. In: MRDM (2004)
18. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
19. Weis, M., Manolescu, I.: Declarative xml data cleaning with xclean. In: CAiSE (2007)
20. Winkler, W.E.: The state of record linkage and current research problems. Technical report (1999)