Knowledge and Information Systems
https://doi.org/10.1007/s10115-018-1246-2

REGULAR PAPER

Collective entity resolution in multi-relational familial networks

Pigi Kouki[1] · Jay Pujara[1] · Christopher Marcum[2] · Laura Koehly[2] · Lise Getoor[1]

Received: 15 January 2018 / Revised: 21 April 2018 / Accepted: 9 June 2018

Abstract
Entity resolution in settings with rich relational structure often introduces complex dependencies between co-references. Exploiting these dependencies is challenging—it requires seamlessly combining statistical, relational, and logical dependencies. One task of particular interest is entity resolution in familial networks. In this setting, multiple partial representations of a family tree are provided, from the perspective of different family members, and the challenge is to reconstruct a family tree from these multiple, noisy, partial views. This reconstruction is crucial for applications such as understanding genetic inheritance, tracking disease contagion, and performing census surveys. Here, we design a model that incorporates statistical signals (such as name similarity), relational information (such as sibling overlap), logical constraints (such as transitivity and bijective matching), and predictions from other algorithms (such as logistic regression and support vector machines), in a collective model. We show how to integrate these features using probabilistic soft logic, a scalable probabilistic programming framework. In experiments on real-world data, our model significantly outperforms state-of-the-art classifiers that use relational features but are incapable of collective reasoning.

Keywords Entity resolution · Data integration · Familial networks · Multi-relational networks · Collective classification · Family reconstruction · Probabilistic soft logic

✉ Pigi Kouki
pkouki@soe.ucsc.edu

Jay Pujara
jay@cs.umd.edu

Christopher Marcum
chris.marcum@nih.gov

Laura Koehly
koehlyl@mail.nih.gov

Lise Getoor
getoor@soe.ucsc.edu

[1] School of Engineering, University of California Santa Cruz, Santa Cruz, USA

[2] National Human Genome Research Institute, National Institutes of Health, Bethesda, USA

Published online: 15 September 2018

1 Introduction

Entity resolution, the problem of identifying, matching, and merging references corresponding to the same entity within a dataset, is a widespread challenge in many domains. Here, we consider one particularly compelling application: the problem of entity resolution in familial networks, which is an essential component in applications such as social network analysis [17], medical studies [24], family health tracking and electronic healthcare records [18], genealogy studies [12,25] and areal administrative records, such as censuses [34]. Familial networks contain a rich set of relationships between entities with a well-defined structure, which differentiates this problem setting from general relational domains such as citation networks that contain a fairly restricted set of relationship types.

As a concrete example of entity resolution in familial networks, consider the healthcare records for several patients from a single family. Each patient supplies a family medical history, identifying the relationship to an individual and their symptoms. One patient may report that his 15-year-old son suffers from high blood sugar, while another patient from the same family may report that her 16-year-old son suffers from type 1 diabetes. Assembling a complete medical history for this family requires determining whether the two patients have the same son and are married.

In this setting, a subset of family members independently provide a report of their familial relationships. This process yields several ego-centric views of *a portion* of a familial network, i.e., persons in the family together with their relationships. Our goal is to infer the entire familial network by identifying the people that are the same across these ego-centric views. For example, in Fig. 1 we show two partial trees for one family. In the left tree, the patient "Jose Perez" reported his family tree and mentioned that his 15-year-old son, also named "Jose Perez," has high blood sugar. In the right tree, the patient "Anabel Perez" reported her family tree and mentioned that her 16-year-old son suffers from type 1 diabetes. In order to assemble a complete medical history for this family, we need to infer which references refer to the same person. For our example trees, we present in Fig. 2 the resolved entities indicated by the same shades. For example, "Ana Maria Perez" from the left tree is the same person with "Anabel Perez" from the right tree. Our ultimate goal is to reconstruct the underlying family tree, which in our example is shown in Fig. 3.

Typical approaches to performing entity resolution use attributes characterizing a reference (e.g., name, occupation, age) to compute different statistical signals that capture similarity, such as string matching for names and numeric distance for age [34]. However, relying

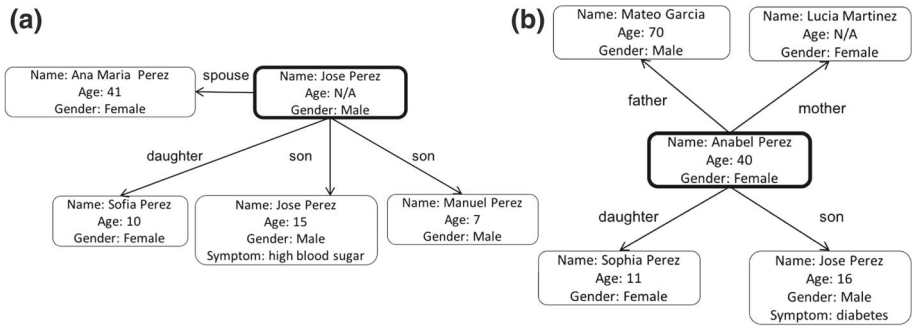

Fig. 1 Two familial ego-centric trees for family F. Bold black borders indicate the root of the tree, i.e., the root of tree **a** is "Jose Perez" and the root of tree **b** is "Anabel Perez"

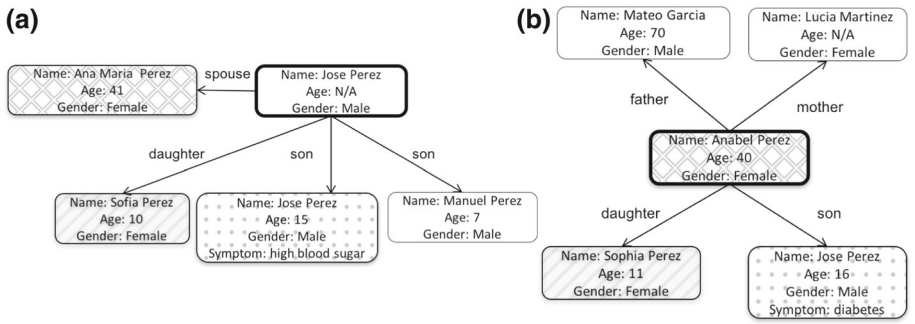

Fig. 2 The two familial ego-centric trees for family F with resolved entities. Persons in same shade represent same entities, e.g., "Ana Maria Perez" from tree (**a**) and "Anabel Perez" in tree (**b**) are co-referent. White means that the persons were not matched across the trees

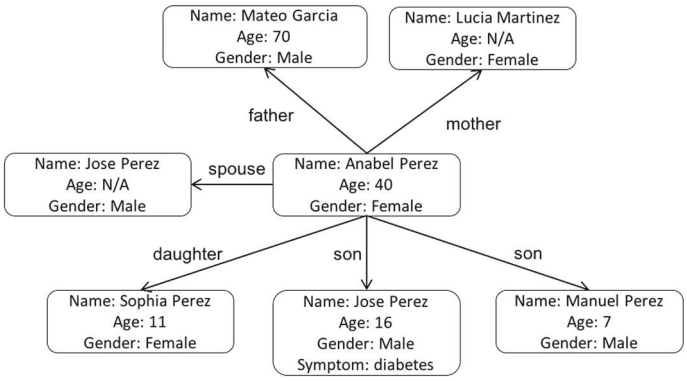

Fig. 3 The aggregated family tree for family F

only on attribute similarity to perform entity resolution in familial networks is problematic since these networks present unique challenges: attribute data are frequently incomplete, unreliable, and/or insufficient. Participants providing accounts of their family frequently forget to include family members or incorrectly report attributes, such as ages of family members. In other cases, they refer to the names using alternate forms. For example, consider the two ego-centric trees of Fig. 1. The left tree contains one individual with the name "Ana Maria Perez" (age 41) and the right one an individual with the name "Anabel Perez" (age 40). In this case, using name and age similarity only, we may possibly determine that these persons are not co-referent, since their ages do not match and the names vary substantially. Furthermore, even when participants provide complete and accurate attribute information, this information may be insufficient for entity resolution in familial networks. In the same figure, the left tree contains two individuals of the name "Jose Perez," while the right tree contains only one individual "Jose Perez." Here, since we have a perfect match for names for these three individuals, we cannot reach a conclusion which of the two individuals of the left tree named after "Jose Perez" match the individual "Jose Perez" from the right tree. Additionally, using age similarity would help in the decision; however, this information is missing for one person. In both cases, the performance of traditional approaches that rely on attribute similarities suffers in the setting of familial trees.

In this scenario, there is a clear benefit from exploiting *relational information* in the familial networks. Approaches incorporating relational similarities [5,10,20] frequently outperform those relying on attribute-based similarities alone. *Collective* approaches [32] where related resolution decisions are made jointly, rather than independently, showed improved entity resolution performance, albeit with the tradeoff of increased time complexity. General approaches to collective entity resolution have been proposed [30], but these are generally appropriate for one or two networks and do not handle many of the unique challenges of familial networks. Accordingly, much of the prior work in collective, relational entity resolution has incorporated only one, or a handful, of relational types, has limited entity resolution to one or two networks, or has been hampered by scalability concerns.

In contrast to previous approaches, we develop a scalable approach for collective relational entity resolution across multiple networks with multiple relationship types. Our approach is capable of using incomplete and unreliable data in concert with the rich multi-relational structure found in familial networks. Additionally, our model can also incorporate input from other algorithms when such information is available. We view the problem of entity resolution in familial networks as a collective classification problem and propose a model that can incorporate statistical signals, relational information, logical constraints, and predictions from other algorithms. Our model is able to collectively reason about entities across networks using these signals, resulting in improved accuracy. To build our model, we use *probabilistic soft logic* (PSL) [2], a probabilistic programming framework which uses soft constraints to specify a joint distribution over possible entity matchings. PSL is especially well suited to entity resolution tasks due to its ability to unify attributes, relations, constraints such as bijection and transitivity, and predictions from other models, into a single model.

We note that this work is an extended version of [22]. Our contributions mirror the structure of this paper:

- We formally define the problem of entity resolution for familial networks (Sect. 2).
- We introduce a process of *normalization* that enables the use of relational features for entity resolution in familial networks (Sect. 3).
- We develop a scalable entity resolution framework that effectively combines attributes, relational information, logical constraints, and predictions from other baseline algorithms (Sect. 4).
- We perform extensive evaluation on two real-world datasets, from real patient data from the National Institutes of Health and Wikidata, demonstrating that our approach beats state-of-the-art methods while maintaining scalability as problems grow (Sect. 5).
- We provide a detailed analysis of which features are most useful for relational entity resolution, providing advice for practitioners (Sect. 5.3.1).
- We experimentally evaluate the state-of-the-art approaches against our method, comparing performance based on similarity functions (Sect. 5.4), noise level (Sect. 5.5), and number of output pairs (Sect. 5.6).
- We provide a brief survey of related approaches to relational entity resolution (Sect. 6).
- We highlight several potential applications for our method and promising extensions to our approach (Sect. 7).

2 Problem setting

We consider the problem setting where we are provided a set of ego-centric *reports* of a familial network. Each report is given from the perspective of a *participant* and consists

of two types of information: family members and relationships. The participant identifies a collection of family members and provides personal information such as name, age, and gender for each person (including herself). The participant also reports their relationships to each family member, which we categorize as first-degree relationships (mother, father, sister, daughter, etc.) or second-degree relationships (grandfather, aunt, nephew, etc.). Our task is to align family members *across* reports in order to reconstruct a complete family tree. We refer to this task as *entity resolution in familial networks* and formally define the problem as follows:

Problem definition We assume there is an underlying family $\mathbf{F} = \langle \mathbf{A}, \mathbf{Q} \rangle$ which contains (unobserved) actors $\mathbf{A}$ and (unobserved) relationships $\mathbf{Q}$ among them. We define $\mathbf{A} = \{A_1, A_2, \ldots, A_m\}$ and $\mathbf{Q} = \{r_{t_a}(A_i, A_j), r_{t_a}(A_i, A_k), r_{t_b}(A_k, A_l) \ldots r_{t_z}(A_k, A_m)\}$. Here, $t_a, t_b, t_z \in \tau$ are different relationship types between individuals (e.g., son, daughter, father, aunt). Our goal is to recover $\mathbf{F}$ from a set of k participant reports, $\mathcal{R}$.

We define these reports as $\mathcal{R} = \{\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^k\}$, where superscripts will henceforth denote the participant associated with the reported data. Each report, $\mathbf{R}^i = \langle p^i, \mathbf{M}^i, \mathbf{Q}^i \rangle$ is defined by the reporting participant, p^i, the set of family members mentioned in the report, $\mathbf{M}^i$, and the participant's relationships to each mention, $\mathbf{Q}^i$. We denote the mentions, $\mathbf{M}^i = \{p^i, m_1^i, \ldots, m_{l_i}^i\}$, where each of the l_i mentions includes (possibly erroneous) personal attributes and corresponds to a distinct, unknown actor in the family tree (note that the participant is a mention as well). We denote the relationships $\mathbf{Q}^i = \{r_{t_a}(p^i, m_x^i), \ldots, r_{t_b}(p^i, m_y^i)\}$, where $t_a, t_b \in \tau$ denote the types of relation, and m_x^i and m_y^i denote the mentioned family members with whom the participant p^i shares the relation types t_a and t_b, respectively. A participant p^i can have an arbitrary number of relations of the same type (e.g., two daughters, three brothers, zero sisters). Our goal is to examine all the mentions (participants and non-participants) and perform a matching across reports to create sets of mentions that correspond to the same actor. The ultimate task is to construct the unified family $\mathbf{F}$ from the collection of matches.

Entity resolution task A prevalent approach to entity resolution is to cast the problem as a *binary, supervised classification* task and use machine learning to label each pair of entities as matching or non-matching. In our specific problem setting, this corresponds to introducing a variable $\textsc{Same}(x, y)$ for each pair of entities x, y occurring in distinct participant reports. Formally, we define the variable $\textsc{Same}(m_x^i, m_y^j)$ for each pair of mentions in distinct reports, i.e., $\forall_{i \neq j} \forall_{m_x^i \in \mathbf{M}^i} \forall_{m_y^j \in \mathbf{M}^j}$. Our goal is to determine for each pair of mentions whether they refer to the same actor.

In order to achieve this goal, we must learn a decision function that, given two mentions, determines whether they are the same. Although the general problem of entity resolution is well studied, we observe that a significant opportunity in this specific problem setting is the ability to leverage the familial relationships in each report to perform relational entity resolution. Unfortunately, the available reports, $\mathcal{R}$ are each provided from the perspective of a unique participant. This poses a problem since we require relational information for each *mention* in a report, not just for the reporting participant. As a concrete example, if one participant report mentions a son and another report mentions a brother, comparing these mentions from the perspectives of a parent and sibling, respectively, is complex. Instead, if relational features of the mention could be reinterpreted from a common perspective, the two mentions could be compared directly. We refer to the problem of recovering mention-specific relational features from participant reports as *relational normalization* and present our algorithm in the next section.

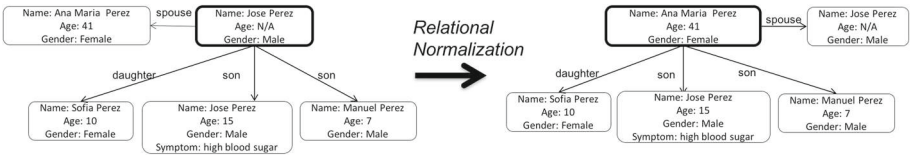

Fig. 4 Left: the tree corresponding to a participant report provided by "Jose Perez." Right: the derived normalized tree from the perspective of "Ana Maria Perez"

3 Preprocessing via relational normalization

Since the relational information available in participant reports is unsuitable for entity resolution, we undertake the process of normalization to generate mention-specific relational information. To do so, we translate the relational information in a report $\mathbf{R}^i$ into an *ego-centric tree*, $\mathbf{T}^i_j$, for each mention m^i_j. Here, the notation $\mathbf{T}^i_j$ indicates that the tree is constructed from the perspective of the jth mention of the ith report. We define $\mathbf{T}^i_j = \langle m^i_j, \mathbf{Q}^i_j \rangle$, where $\mathbf{Q}^i_j$ is a set of relationships. Constructing these trees consists of two steps: relationship inversion and relationship imputation.

Relationship inversion The first step in populating the ego-centric tree for m^i_j is to invert the relationships in $\mathbf{R}^i$ so that the first argument (subject) is m^i_j. More formally, for each relation type $t_j \in \tau$ such that $r_{t_j}(p^i, m^i_j)$, we introduce an inverse relationship $r_{t'_i}(m^i_j, p^i)$. In order to do so, we introduce a function $inverse(\tau, m^i_j, p^i) \rightarrow \tau$ which returns the appropriate inverse relationship for each relation type. Note that the inverse of a relation depends both on the mention and the participant, since in some cases mention attributes (e.g., father to daughter) or participant attributes (e.g., daughter to father) are used to determine the inverse.

Relationship imputation The next step in populating $\mathbf{T}^i_j$ is to impute relationships for m^i_j mediated through p^i. We define a function $impute(r_x(p^i, m^i_j), r_y(p^i, m^i_k)) \rightarrow r_k(m^i_j, m^i_k)$. For example, given the relations $\{r_{father}(p^i, m^i_j), r_{mother}(p^i, m^i_k)\}$ in $\mathbf{T}^i(p^i)$, then we impute the relations $r_{spouse}(m^i_j, m^i_k)$ in $\mathbf{T}^i_j$ as well as $r_{spouse}(m^i_k, m^i_j)$ in $\mathbf{T}^i_k$.

Figure 4 shows an example of the normalization process. We begin with the left tree centered on "Jose Perez" and after applying inversion and imputation we produce the right tree centered on "Ana Maria Perez." Following the same process, we will produce three more trees centered on "Sofia Perez," "Manuel Perez," and "Jose Perez" (with age 15). Finally, we note that since initially we have relational information for just one person in each tree, it will be impossible to use any relational information if we do not perform the normalization step.

4 Entity resolution model for familial networks

After recovering the mention-specific relational features from participant reports, our next step is to develop a model that is capable of collectively inferring mention equivalence using the attributes, diverse relational evidence, and logical constraints. We cast this entity resolution task as inference in a graphical model, and use the probabilistic soft logic (PSL) framework to define a probability distribution over co-referent mentions. Several features of this problem setting necessitate the choice of PSL: (1) entity resolution in familial networks is inherently collective, requiring constraints such as transitivity and bijection; (2) the multitude of relationship types require an expressive modeling language; (3) similarities

between mention attributes take continuous values; (4) potential matches scale polynomially with mentions, requiring a scalable solution. PSL provides collective inference, expressive relational models defined over continuously valued evidence, and formulates inference as a scalable convex optimization. In this section, we provide a brief primer on PSL and then introduce our PSL model for entity resolution in familial networks.

4.1 Probabilistic soft logic (PSL)

Probabilistic soft logic is a probabilistic programming language that uses a first-order logical syntax to define a graphical model [2]. In contrast to other approaches, PSL uses continuous random variables in the [0, 1] unit interval and specifies factors using convex functions, allowing tractable and efficient inference. PSL defines a Markov random field associated with a conditional probability density function over random variables $\mathbf{Y}$ conditioned on evidence $\mathbf{X}$,

$$P(\mathbf{Y}|\mathbf{X}) \propto \exp\left(-\sum_{j=1}^{m} w_j \phi_j(\mathbf{Y}, \mathbf{X})\right), \tag{1}$$

where ϕ_j is a convex potential function and w_j is an associated weight which determines the importance of ϕ_j in the model. The potential ϕ_j takes the form of a *hinge-loss*:

$$\phi_j(\mathbf{Y}, \mathbf{X}) = (\max\{0, \ell_j(\mathbf{X}, \mathbf{Y})\})^{p_j} . \tag{2}$$

Here, ℓ_j is a linear function of $\mathbf{X}$ and $\mathbf{Y}$, and $p_j \in \{1, 2\}$ optionally squares the potential, resulting in a *squared-loss*. The resulting probability distribution is log-concave in $\mathbf{Y}$, so we can solve maximum a posteriori (MAP) inference exactly via convex optimization to find the optimal $\mathbf{Y}$. We use the alternating direction method of multipliers (ADMM) approach of Bach et al. [2] to perform this optimization efficiently and in parallel. The convex formulation of PSL is the key to efficient, scalable inference in models with many complex interdependencies.

PSL derives the objective function by translating logical rules specifying dependencies between variables and evidence into hinge-loss functions. PSL achieves this translation by using the *Lukasiewicz* norm and co-norm to provide a relaxation of Boolean logical connectives [2]:

$$\begin{aligned} p \wedge q &= \max(0, p + q - 1) \\ p \vee q &= \min(1, p + q) \\ \neg p &= 1 - p . \end{aligned}$$

Recent work in PSL [2] provides a detailed description of PSL operators. To illustrate PSL in an entity resolution context, the following rule encodes that mentions with similar names and the same gender might be the same person:

$$\textsc{SimName}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) , \tag{3}$$

where $\textsc{SimName}(\mathtt{m}_1, \mathtt{m}_2)$ is a continuous observed atom taken from the string similarity between the names of $\mathtt{m}_1$ and $\mathtt{m}_2$, $\textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2)$ is a binary observed atom that takes its value from the logical comparison $\mathtt{m}_1.\text{gender} = \mathtt{m}_2.\text{gender}$ and $\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$ is a continuous value to be inferred, which encodes the probability that the mentions $\mathtt{m}_1$ and $\mathtt{m}_2$ are the same person. If this rule was instantiated with the assignments $\mathtt{m}_1$=`John Smith`, $\mathtt{m}_2$=`J Smith`

the resulting hinge-loss potential function would have the form:

$$\begin{aligned}\max(0,\ &\textsc{SimName}(\texttt{JohnSmith}, \texttt{JSmith})\\ &+ \textsc{eqGender}(\texttt{JohnSmith}, \texttt{JSmith})\\ &- \textsc{Same}(\texttt{JohnSmith}, \texttt{JSmith}) - 1)\ .\end{aligned}$$

4.2 PSL model

We define our model using rules similar to those in (3), allowing us to infer the SAME relation between mentions. Each rule encodes graph-structured dependency relationships drawn from the familial network (e.g., if two mentions are co-referent, then their mothers should also be co-referent) or conventional attribute-based similarities (e.g., if two mentions have similar first and last name, then they are possibly co-referent). We present a set of representative rules for our model, but note that additional features (e.g., locational similarity, conditions from a medical history, or new relationships) can easily be incorporated into our model with additional rules.

4.2.1 Scoping the rules

Familial datasets may consist of several mentions and reports. However, our goal is to match mentions from the same family that occur in distinct reports. Obviously, mentions that belong to different families could not be co-referent, so we should only compare mentions that belong to the same family. In order to restrict rules to such mentions, it is necessary to perform scoping on our logical rules. We define two predicates: BELONGSTOFAMILY (abbreviated $\text{BF}(\mathtt{m}_x, \mathtt{F})$) and FROMREPORT (abbreviated $\text{FR}(\mathtt{m}_i, \mathtt{R}_i)$). BF allows us to identify mentions from a particular family's reports, i.e., $\{m_x^i \in \mathbf{M}^i \text{ s.t. } \mathbf{R}^i \in \mathbf{F}\}$, while FR filters individuals from a particular participant report, i.e., $\{m_x^i \in \mathbf{M}^i\}$. In our matching, we wish to compare mentions from the same family but appearing in different participant reports. To this end, we introduce the following clause to our rules:

$$\text{BF}(\mathtt{m}_1, \mathtt{F}) \wedge \text{BF}(\mathtt{m}_2, \mathtt{F}) \wedge \text{FR}(\mathtt{m}_1, \mathtt{R}_i) \wedge \text{FR}(\mathtt{m}_2, \mathtt{R}_j) \wedge \mathtt{R}_i \neq \mathtt{R}_j$$

In the rest of our discussion below, we assume that this scoping clause is included, but we omit replicating it in favor of brevity.

4.2.2 Name similarity rules

One of the most important mention attributes is mention names. Much of the prior research on entity resolution has focused on engineering similarity functions that can accurately capture patterns in name similarity. Two such popular similarity functions are the Levenshtein [26] and Jaro–Winkler [34]. The first is known to work well for common typographical errors, while the second is specifically designed to work well with names. We leverage mention names by introducing rules that capture the intuition that when two mentions have similar names, then they are more likely to represent the same person. For example, when using the Jaro–Winkler function to compute the name similarities, we use the following rule:

$$\textsc{SimName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\ .$$

This rule reinforces an important aspect of PSL: atoms take truth values in the [0, 1] interval, capturing the degree of certainty of the inference. In the above rule, high name similarity

results in greater confidence that two mentions are the same. However, we also wish to penalize pairs of mentions with dissimilar names from matching, for which we introduce the rule using the logical not ($\neg$):

$$\neg\textsc{SimName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \ .$$

The above rules use a generic SIMNAME similarity function. In fact, our model introduces several name similarities for first, last, and middle names as follows:

$$\begin{aligned}
&\textsc{SimFirstName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\\
&\textsc{SimMaidenName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\\
&\textsc{SimLastName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\\
&\neg\textsc{SimFirstName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\\
&\neg\textsc{SimMaidenName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\\
&\neg\textsc{SimLastName}_{JW}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \ .
\end{aligned}$$

In the above rules, we use the Jaro–Winkler similarity function. In our basic model, we additionally introduce the same rules that compute similarities using the Levenshtein distance as well. Finally, we experiment with adding other popular similarity functions, i.e., Monge Elkan, Soundex, Jaro [34], and their combinations and discuss how different string similarity metrics affect performance in the experimental section.

4.2.3 Personal information similarity rules

In addition to the name attributes of a mention, there are often additional attributes provided in reports that are useful for matching. For example, age is an important feature for entity resolution in family trees since it can help us discern between individuals having the same (or very similar) name but belonging to different generations. We introduce the following rules for age:

$$\begin{aligned}
\textsc{SimAge}(\mathtt{m}_1, \mathtt{m}_2) &\Rightarrow \ \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\\
\neg\textsc{SimAge}(\mathtt{m}_1, \mathtt{m}_2) &\Rightarrow \ \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \ .
\end{aligned} \tag{4}$$

The predicate $\textsc{SimAge}(\mathtt{m}_1, \mathtt{m}_2)$ takes values in the interval [0, 1] and is computed as the ratio of the smallest over the largest value, i.e.:

$$\textsc{SimAge}(\mathtt{m}_1, \mathtt{m}_2) = \frac{\min\{\mathtt{m}_1.\text{age}, \mathtt{m}_2.\text{age}\}}{\max\{\mathtt{m}_1.\text{age}, \mathtt{m}_2.\text{age}\}} \ .$$

The above rules will work well when the age is known. However, in the familial networks setting that we are operating on it is often the case where personal information and usually the age is not known. For these cases, we can specifically ask from our model to take into account only cases where the personal information is known and ignore it when this is not available. To this end, we replace the rules in (4) with the following:

$$\begin{aligned}
&\textsc{KnownAge}(\mathtt{m}_1) \wedge \textsc{KnownAge}(\mathtt{m}_2) \wedge \textsc{SimAge}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)\\
&\textsc{KnownAge}(\mathtt{m}_1) \wedge \textsc{KnownAge}(\mathtt{m}_2) \wedge \neg\textsc{SimAge}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)
\end{aligned}$$

In other words, using the scoping predicates $\textsc{KnownAge}(\mathtt{m}_1)$ and $\textsc{KnownAge}(\mathtt{m}_2)$ we can handle missing values in the PSL model, which is an important characteristic.

While attributes like age have influence in matching, other attributes cannot be reliably considered as evidence to matching, but they are far more important in disallowing matches between the mentions. For example, simply having the same gender is not a good indicator that two mentions are co-referent. However, having a different gender is a strong evidence that two mentions are not co-referent. To this end, we also introduce rules that prevent mentions from matching when certain attributes differ:

$$\neg\text{EQGENDER}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$$
$$\neg\text{EQLIVING}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\text{SAME}(\mathtt{m}_1, \mathtt{m}_2) .$$

We note that the predicates $\text{EQGENDER}(\mathtt{m}_1, \mathtt{m}_2)$ and $\text{EQLIVING}(\mathtt{m}_1, \mathtt{m}_2)$ are binary-valued atoms.

4.2.4 Relational similarity rules

Although attribute similarities provide useful features for entity resolution, in problem settings such as familial networks, relational features are necessary for matching. Relational features can be introduced in a multitude of ways. One possibility is to incorporate purely structural features, such as the number and types of relationships for each mention. For example, given a mention with two sisters and three sons and a mention with three sisters and three sons, we could design a similarity function for these relations. However, practically this approach lacks discriminative power because there are often mentions that have similar relational structures (e.g., having a mother) that refer to different entities. To overcome the lack of discriminative power, we augment structural similarity with a matching process. For relationship types that are surjective, such as mother or father, the matching process is straightforward. We introduce a rule:

$$\text{SIMMOTHER}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \text{SAME}(\mathtt{m}_1, \mathtt{m}_2) ,$$

SIMMOTHER may have many possible definitions, ranging from an exact string match to a recursive similarity computation. In this subsection, we define SIMMOTHER as equal to the maximum of the Levenshtein and Jaro–Winkler similarities of the first names, and discuss a more sophisticated treatment in the next subsection. However, when a relationship type is multi-valued, such as sister or son, a more sophisticated matching of the target individuals is required. Given a relation type t and possibly co-referent mentions m_1^i, m_2^j, we find all entities $M_x = \{m_x^i : r_t(m_1^i, m_x^i) \in \mathbf{Q}_1^i\}$ and $M_y = \{m_y^j : r_t(m_2^j, m_y^j) \in \mathbf{Q}_2^j\}$. Now we must define a similarity for the sets M_x and M_y, which in turn will provide a similarity for m_1^i and m_2^j. The similarity function we use is:

$$\text{SIM}_t(\mathtt{m}_1, \mathtt{m}_2) = \frac{1}{|M_x|} \sum_{\mathtt{m}_x \in M_x} \max_{\mathtt{m}_y \in M_y} \text{SIMNAME}(\mathtt{m}_x, \mathtt{m}_y) .$$

For each $\mathtt{m}_x$ (an individual with relation t to $\mathtt{m}_1$), this computation greedily chooses the best $\mathtt{m}_y$ (an individual with relation t to $\mathtt{m}_2$). In our computation, we assume (without loss of generality, assuming symmetry of the similarity function) that $|M_x| < |M_y|$. While many possible similarity functions can be used for SIMNAME, we take the maximum of the Levenshtein and Jaro–Winkler similarities of the first names in our model.

Our main goal in introducing these relational similarities is to incorporate relational evidence that is compatible with simpler, baseline models. While more sophisticated than simple structural matches, these relational similarities are much less powerful than the transitive relational similarities supported by PSL, which we introduce in the next section.

4.2.5 Transitive relational (similarity) rules

The rules that we have investigated so far can capture personal and relational similarities, but they cannot identify similar persons in a collective way. To make this point clear, consider the following observation: when we have high confidence that two persons are the same, we also have a stronger evidence that their associated relatives, e.g., father, are also the same. We encode this intuition with rules of the following type:

$$\textsc{Rel}(\texttt{Father}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\texttt{Father}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b) \ .$$

The rule above works well with surjective relationships, since each person can have only one (biological) father. When the cardinality is larger, e.g., sister, our model must avoid inferring that all sisters of two respective mentions are the same. In these cases, we use additional evidence, i.e., name similarity, to select the appropriate sisters to match, as follows:

$$\textsc{Rel}(\texttt{Sister}, \mathtt{m}_1, \mathtt{m}_a) \wedge \ \textsc{Rel}(\texttt{Sister}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b) \ .$$

Just as in the previous section, we compute SIMNAME by using the maximum of the Jaro–Winkler and Levenshtein similarities for first names. For relationships that are one to one, we can also introduce negative rules which express the intuition that two different persons should be connected to different persons given a specific relationship. For example, for a relationship such as spouse, we can use a rule such as:

$$\textsc{Rel}(\texttt{Spouse}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\texttt{Spouse}, \mathtt{m}_2, \mathtt{m}_b) \wedge \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\textsc{Same}(\mathtt{m}_a, \mathtt{m}_b).$$

However, introducing similar rules for one-to-many relationships is inadvisable. To understand why, consider the case where two siblings do not match, yet they have the same mother, whose match confidence should remain unaffected.

4.2.6 Bijection and transitivity rules

Our entity resolution task has several natural constraints across reports. The first is bijection, namely that a mention m_x^i can match at most one mention, m_y^j from another report. According to the bijection rule, if mention m_a from report R_1 is matched to mention m_b from report R_2, then m_1 cannot be matched to any other mention from report R_2:

$$\textsc{FR}(\mathtt{m}_a, \mathtt{R}_1) \wedge \textsc{FR}(\mathtt{m}_b, \mathtt{R}_2) \wedge \textsc{FR}(\mathtt{m}_c, \mathtt{R}_2) \wedge \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \neg\textsc{Same}(\mathtt{m}_a, \mathtt{m}_c) \ .$$

Note that this bijection is *soft* and does not guarantee a single, exclusive match for $\mathtt{m}_a$, but rather attenuates the confidence in each possible match modulated by the evidence for the respective matches. A second natural constraint is transitivity, which requires that if m_a^i and m_y^j are the same, and mentions m_y^j and m_c^k are the same, then mentions m_a^i and m_c^k should also be the same. We capture this constraint as follows:

$$\textsc{FR}(\mathtt{m}_a, \mathtt{R}_1) \wedge \textsc{FR}(\mathtt{m}_b, \mathtt{R}_2) \wedge \textsc{FR}(\mathtt{m}_c, \mathtt{R}_3) \wedge \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_b, \mathtt{m}_c) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_c) \ .$$

4.2.7 Prior rule

Entity resolution is typically an imbalanced classification problem, meaning that most of the mention pairs are not co-referent. We can model our general belief that two mentions are likely not co-referent, using the prior rule:

$$\neg \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \ .$$

4.2.8 Rules to leverage existing classification algorithms

Every state-of-the-art classification algorithm has strengths and weaknesses which may depend on data-specific factors such as the degree of noise in the dataset. In this work, our goal is to provide a flexible framework that can be used to generate accurate entity resolution decisions for any data setting. To this end, we can also incorporate the predictions from different methods into our unified model. Using PSL as a meta-model has been successfully applied in recent work [29]. In our specific scenario of entity resolution, for example, the predictions from three popular classifiers (logistic regression (LR), support vector machines (SVMs), and logistic model trees (LMTs)) can be incorporated in the model via the following rules:

$$\begin{aligned}
&\textsc{SameLR}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \\
&\neg\textsc{SameLR}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \\
&\textsc{SameSVMs}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \\
&\neg\textsc{SameSVMs}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \\
&\textsc{SameLMTs}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \\
&\neg\textsc{SameLMTs}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \ \neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \ .
\end{aligned}$$

Note that for each classifier we introduce two rules: 1) the direct rule which states that if a given classifier predicts that the mentions are co-referent, then it is likely that they are indeed co-referent and 2) the reverse rule which states that if the classifier predicts that the mentions are not co-referent, then it is likely that they are not co-referent. Additionally, using PSL we can introduce more complex rules that combine the predictions from the other algorithms. For example, if all three classifiers agree that a pair of mentions is co-referent, then this is strong evidence that this pair of mentions is indeed co-referent. Similarly, if all three classifiers agree that the pair of mentions is not co-referent, then this is strong evidence that they are not co-referent. We can model these ideas through the following rules:

$$\begin{aligned}
&\textsc{SameLR}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SameSVMs}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SameLMTs}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \\
&\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \\
&\neg\textsc{SameLR}(\mathtt{m}_1, \mathtt{m}_2) \wedge \neg\textsc{SameSVMs}(\mathtt{m}_1, \mathtt{m}_2) \wedge \neg\textsc{SameLMTs}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \\
&\neg\textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \ .
\end{aligned}$$

4.2.9 Flexible modeling

We reiterate that in this section we have only provided *representative* rules used in our PSL model for entity resolution. A full compendium of all rules used in our experiments is presented in the appendix. Moreover, a key feature of our model is the flexibility and the ease with which it can be extended to incorporate new features. For example, adding

additional attributes, such as profession or location, is easy to accomplish following the patterns of Sect. 4.2.3. Incorporating additional relationships, such as cousins or friends, is simply accomplished using the patterns in Sects. 4.2.4 and 4.2.5. Our goal has been to present a variety of patterns that are adaptable across different datasets and use cases.

4.3 Learning the PSL model

Given the above model, we use observational evidence (similarity functions and relationships) and variables (potential matches) to define a set of ground rules. Each ground rule is translated into a hinge-loss potential function of the form (2) defining a Markov random field, as in (1) (Sect. 4.1). Then, given the observed values $\mathbf{X}$, our goal is to find the most probable assignment to the unobserved variables $\mathbf{Y}$ by performing joint inference over interdependent variables.

As we discussed in 4.1, each of the first-order rules introduced in the previous section is associated with a nonnegative weight w_j in Eq. 1. These weights determine the relative importance of each rule, corresponding to the extent to which the corresponding hinge function ϕ_j alters the probability of the data under Eq. 1. A higher weight w_j corresponds to a greater importance of information source j in the entity resolution task. We learn rule weights using Bach et al.'s [2] approximate maximum likelihood weight learning algorithm, using a held-out training set. The algorithm approximates a gradient step in the conditional likelihood,

$$\frac{\partial log\, P(\mathbf{Y}|\mathbf{X})}{\partial w_j} = \mathbb{E}_w[\phi_j(\mathbf{Y}, \mathbf{X})] - \phi_j(\mathbf{Y}, \mathbf{X}) \,, \tag{5}$$

by replacing the intractable expectation with the MAP solution based on w, which can be rapidly solved using ADMM. Finally, since the output of the PSL model is a soft truth value for each pair of mentions, to evaluate our matching we choose a threshold to make a binary match decision. We choose the optimal threshold on a held-out development set to maximize the F-measure score and use this threshold when classifying data in the test set.

4.4 Satisfying matching restrictions

One of the key constraints in our model is a bijection constraint that requires that each mention can match at most one mention in another report. Since the bijection rule in PSL is *soft*, in some cases, we may get multiple matching mentions for a report. To enforce this restriction, we introduce a greedy 1:1 matching step. We use a simple algorithm that first sorts output matchings by the truth value of the SAME(m_x^i, m_y^j) predicate. Next, we iterate over this sorted list of mention pairs, choosing the highest ranked pair for an entity, (m_x^i, m_y^j). We then remove all other potential pairs, $\forall_{m_a^i, a \neq x}(m_a^i, m_y^j)$ and $\forall_{m_b^j, b \neq y}(m_x^i, m_b^j)$, from the matching. The full description can be found in Algorithm 1. This approach is simple to implement, efficient, and can potentially improve model performance, as we will discuss in our experiments.

input : A set of mention pairs classified as MATCH together with the likelihood of the MATCH
output: A set of mention pairs satisfying the one-to-one matching restrictions

repeat
  pick unmarked pair $\{a_i, a_j\}$ with highest MATCH likelihood;
  output pair $\{a_i, a_j\}$ as MATCH;
  mark pair $\{a_i, a_j\}$;
  output all other pairs containing either a_i or a_j as NO MATCH;
  mark all other pairs containing either a_i or a_j;
until *all pairs are marked*;

Algorithm 1: Satisfying matching restrictions.

5 Experimental validation

5.1 Datasets and baselines

For our experimental evaluation, we use two datasets: a clinical dataset provided by the National Institutes of Health (NIH) [15] and a public dataset crawled from the structured knowledge repository, Wikidata.[1] We provide summary statistics for both datasets in Table 1.

The NIH dataset was collected by interviewing 497 patients from 162 families and recording family medical histories. For each family, 3 or 4 patients were interviewed, and each interview yielded a corresponding ego-centric view of the family tree. Patients provided first- and second-degree relations, such as parents and grandparents. In total, the classification task requires determining co-reference for about 300, 000 pairs of mentions. The provided dataset was manually annotated by at least two coders, with differences reconciled by blind consensus. Only 1.6% of the potential pairs are co-referent, resulting in a severely imbalanced classification, which is common in entity resolution scenarios.

The Wikidata dataset was generated by crawling part of the Wikidata[2] knowledge base. More specifically, we generated a seed set of 419 well-known politicians or celebrities, e.g., "Barack Obama."[3] For each person in the seed set, we retrieved attributes from Wikidata including their full name (and common variants), age, gender, and living status. Wikidata provides familial data only for first-degree relationships, i.e., siblings, parents, children, and spouses. Using the available relationships, we also crawled Wikidata to acquire attributes and relationships for each listed relative. This process resulted in 419 families. For each family, we have a different number of family trees (ranging from 2 to 18) with 1844 family trees in total, and 175,000 pairs of potentially co-referent mentions (8.7% of which are co-referent). Mentions in Wikidata are associated with unique identifiers, which we use as ground truth. In the next section, we describe how we add noise to this dataset to evaluate our method.

We compare our approach to state-of-the-art classifiers that are capable of providing the probability that a given pair of mentions is co-referent. Probability values are essential since they are the input to the greedy 1–1 matching restrictions algorithm. We compare our approach to the following classifiers: logistic regression (LR), logistic model trees (LMTs), and support vector machines (SVMs). For LR, we use a multinomial logistic regression model with a ridge estimator [6] using the implementation and improvements of Weka [14] with the default settings. For LMTs, we use Weka's implementation [23] with the default

[1] Code and data available at: https://github.com/pkouki/icdm2017.

[2] https://www.wikidata.org/.

[3] https://www.wikidata.org/wiki/Q76.

Table 1 Datasets description

Dataset	NIH	Wikidata
No. of families	162	419
No. of family trees	497	1844
No. of mentions	12,111	8553
No. of 1st degree relationships	46,983	49,620
No. of 2nd degree relationships	67,540	0
No. of pairs for comparison	300,547	174,601
% of co-referent pairs	1.6	8.69

settings. For SVMs, we use Weka's LibSVM library [7], along with the functionality to estimate probabilities. To select the best SVM model, we follow the process described by Hsu et al. [19]: we first find the kernel that performs best, which in our case was the radial basis function (RBF). We then perform a grid search to find the best values for C and γ parameters. The starting point for the grid search was the default values given by Weka, i.e., C = 1 and $\gamma = 1/$(number of attributes), and we continue the search with exponentially increasing/decreasing sequences of C and γ. However, unlike our model, none of these off-the-shelf classifiers can incorporate transitivity or bijection.

Finally, we note that we also experimented with off-the-shelf collective classifiers provided by Weka.[4] More specifically, we experimented with Chopper, TwoStageCollective, and YATSI [11]. Among those, YATSI performed the best. YATSI (Yet Another Two-Stage Classifier) is collective in the sense that the predicted label of a test instance will be influenced by the labels of related test instances. We experimented with different configurations of YATSI, such as varying the classification method used, varying the nearest neighbor approach, varying the number of the neighbors to consider, and varying the weighting factor. In our experiments, YATSI was not able to outperform the strongest baseline (which as we will show is LMTs), so, for clarity, we omit these results from our discussion below.

5.2 Experimental setup

We evaluate our entity resolution approach using the metrics of *precision*, *recall*, and *F-measure* for the positive (co-referent) class which are typical for entity resolution problems [8]. For all reported results, we use fivefold cross-validation, with distinct training, development, and test sets. Folds are generated by randomly assigning each of the 162 (NIH) and 419 (Wikidata) families to one of five partitions, yielding folds that contain the participant reports for approximately 32 (NIH) and 83 (Wikidata) familial networks.

The NIH dataset is collected in a real-world setting where information is naturally incomplete and erroneous, and attributes alone are insufficient to resolve the entities. However, the Wikidata resource is heavily curated and assumed to contain no noise. To simulate the noisy conditions of real-world datasets, we introduced additive Gaussian noise to the similarity scores. Noise was added to each similarity metric described in the previous section (e.g., first name Jaro–Winkler, age ratio). For the basic experiments presented in the next Sect. 5.3, results are reported for noise terms drawn from a $N(0, 0.16)$ distribution. In our full experiments (presented in Sect. 5.5), we consider varying levels of noise, finding higher noise correlated with lower performance.

[4] Available at: https://github.com/fracpete/collective-classification-weka-package.

In Sect. 4.2.2, we discussed that PSL can incorporate multiple similarities computed by different string similarity functions. For the basic experiments presented in the next Sect. 5.3, results are reported using the Levenshtein and Jaro–Winkler string similarity functions for PSL and the baselines. In our full experiments (presented in Sect. 5.5), we consider adding other string similarity functions.

In each experiment, for PSL, we use threefolds for training the model weights, onefold for choosing a binary classification threshold, and onefold for evaluating model performance. To train the weights, we use PSL's default values for the two parameters: number of iterations (equal to 25) and step size (equal to 1). For SVMs, we use threefolds for training the SVMs with the different values of C and γ, onefold for choosing the best C and γ combination, and onefold for evaluating model performance. For LR and LMTs, we use threefolds for training the models with the default parameter settings and onefold for evaluating the models. We train, validate, and evaluate using the same splits for all models. We report the average precision, recall, and F-measure together with the standard deviation across folds.

5.3 Performance of PSL and baselines

For our PSL model, we start with a simple feature set using only name similarities (see Sect. 4.2.2), transitivity and bijection soft constraints (see Sect. 4.2.6), and a prior (see Sect. 4.2.7). We progressively enhance the model by adding attribute similarities computed based on personal information, relational similarities, and transitive relationships. For each experiment, we additionally report results when including predictions from the other baselines (described in Sect. 4.2.8). Finally, since our dataset poses the constraint that each person from one report can be matched with at most one person from another report, we consider only solutions that satisfy this constraint. To ensure that the output is a valid solution, we apply the greedy 1:1 matching restriction algorithm (see Sect. 4.4) on the output of the each model.

For each of the experiments, we also ran baseline models that use the same information as the PSL models in the form of features. Unlike our models implemented within PSL, the models from the baseline classifiers do not support collective reasoning, i.e., applying transitivity and bijection is not possible in the baseline models. However, we are able to apply the greedy 1:1 matching restriction algorithm on the output of each of the classifiers for each of the experiments to ensure that we provide a valid solution. More specifically, we ran the following experiments:

Names We ran two PSL models that use as features the first, middle, and last name similarities based on Levenshtein and Jaro–Winkler functions to compute string similarities. In the first model, PSL(N), we use rules only on name similarities, as discussed in Sect. 4.2.2. In the second model, PSL($N + pred$) we enhance PSL(N) by adding rules that incorporate the predictions from the other baseline models as described in Sect. 4.2.8. We also ran LR, LMTs, and SVMs models that use as features the first, middle, and last name similarities based on Levenshtein and Jaro–Winkler measures.

Names + Personal Info We enhance **Names** by adding rules about personal information similarities, as discussed in Sect. 4.2.3. Again, for PSL we ran two models: PSL(P) which does not include predictions from the baselines and PSL($P + pred$) that does include predictions from the baselines. For the baselines, we add corresponding features for age similarity, gender, and living status. This is the most complex feature set that can be supported without using the normalization procedure we introduced in Sect. 3.

Names + Personal + Relational Info (1st degree) For this model and all subsequent models, we perform normalization to enable the use of relational evidence for entity resolution. We

present the performance of four PSL models. In the first model, PSL(R_1), we enhance PSL(P) by adding first-degree relational similarity rules, as discussed in Sect. 4.2.4. First-degree relationships are: mother, father, daughter, son, brother, sister, spouse. In the second model, PSL($R_1 + pred$) we extend PSL(R_1) by adding the predictions from the baselines. In the third model, PSL($R_1 T R_1$), we extend the PSL(R_1) by adding first-degree transitive relational rules, as discussed in Sect. 4.2.5. In the fourth model, PSL($R_1 T R_1 + pred$), we extend the PSL($R_1 T R_1$) by adding the predictions from the baselines. For the baselines, we extend the previous models by adding first-degree relational similarities as features. However, it is not possible to include features similar to the transitive relational rules in PSL, since these models do not support collective reasoning or inference across instances.

Names + Personal + Relational Info (1st + 2nd degree) As above, we evaluate the performance of four PSL models. In the first experiment,

PSL($R_{12} T R_1$), we enhance the model PSL($R_1 T R_1$) by adding second-degree relational similarity rules, as discussed in Sect. 4.2.4. Second-degree relationships are: grandmother, grandfather, granddaughter, grandson, aunt, uncle, niece, nephew. In the second experiment, PSL($R_{12} T R_1 + pred$), we enhance PSL($R_{12} T R_1$) by adding the predictions from the baselines. In the third experiment, PSL($R_{12} T R_{12}$), we enhance PSL($R_{12} T R_1$) by adding second-degree transitive relational similarity rules, as discussed in Sect. 4.2.5. In the fourth experiment, PSL($R_{12} T R_{12} + pred$), we enhance PSL($R_{12} T R_{12}$) by adding the predictions from the baselines. For the baselines, we add the second-degree relational similarities as features. Again, it is not possible to add features that capture the transitive relational similarity rules to the baselines. Since Wikidata dataset does not provide second-degree relations, we do not report experimental results for this case.

5.3.1 Discussion

We present our results in Tables 2 (NIH) and 3 (Wikidata). For each experiment, we denote with bold the best performance in terms of the F-measure. We present the results for both our method and the baselines and only for the positive class (co-referent entities). Due to the imbalanced nature of the task, performance on non-matching entities is similar across all approaches, with precision varying from 99.6 to 99.9%, recall varying from 99.4 to 99.9%, and F-measure varying from 99.5 to 99.7% for the NIH dataset. For the Wikidata, precision varies from 98.7 to 99.8%, recall varies from 98.9 to 99.9%, and F-measure varies from 99.5 to 99.7%. Furthermore, to highlight the most interesting comparisons we introduce Figs. 5, 6, and 7 as a complement for the complete tables. The plots in these figures show the F-measure when varying the classification method (i.e., baselines and PSL) or the amount of information used for the classification (e.g., use only names). Figures in blue are for NIH, while figures in orange are for the Wikidata dataset. Next, we summarize some of our insights from the results of Tables 2 and 3. For the most interesting comparisons, we additionally refer to Figs. 5, 6, and 7.

PSL models universally outperform baselines In each experiment, PSL outperforms all the baselines using the same feature set. PSL produces a statistically significant improvement in F-measure as measured by a paired t test with $\alpha = 0.05$. Of the baselines, LMTs perform best in all experiments and will be used for illustrative comparison. When using name similarities only (**Names** models in Tables 2 and 3) PSL(N) outperforms LMTs by 2.3% and 3.6% (absolute value) for the NIH and the Wikidata dataset accordingly. When adding personal information similarities (**Names + Personal Info**), PSL(P) outperforms LMTs by 1.4% and 2% for the NIH and the Wikidata accordingly. For the experiment **Names + Personal +**

Table 2 Performance of PSL and baseline classifiers with varying types of rules/features for the NIH dataset

Experiment	Method	NIH Precision (SD)	Recall (SD)	F-measure (SD)
Names	LR	0.871 (0.025)	0.686 (0.028)	0.767 (0.022)
	SVMs	0.870 (0.022)	0.683 (0.027)	0.765 (0.020)
	LMTs	0.874 (0.020)	0.717 (0.027)	0.787 (0.022)
	PSL(N)	0.866 (0.021)	0.761 (0.028)	0.810 (0.023)*
	PSL($N + pred$)	0.873 (0.021)	0.764 (0.022)	**0.815** (0.019)
Names + Personal Info	LR	0.968 (0.010)	0.802 (0.035)	0.877 (0.024)
	SVMs	0.973 (0.008)	0.832 (0.025)	0.897 (0.017)
	LMTs	0.961 (0.012)	0.857 (0.020)	0.906 (0.016)
	PSL(P)	0.942 (0.014)	0.900 (0.022)	0.920 (0.015)*
	PSL($P + pred$)	0.949 (0.008)	0.895 (0.018)	**0.921** (0.013)*
Names + personal + relational info (1st degree)	LR	0.970 (0.012)	0.802 (0.034)	0.878 (0.024)
	SVMs	0.983 (0.008)	0.835 (0.026)	0.903 (0.018)
	LMTs	0.961 (0.010)	0.859 (0.020)	0.907 (0.014)
	PSL(R_1)	0.943 (0.012)	0.881 (0.030)	0.910 (0.015)
	PSL($R_1 + pred$)	0.958 (0.009)	0.885 (0.017)	0.920 (0.013)*
	PSL($R_1 T R_1$)	0.964 (0.007)	0.937 (0.015)	0.951 (0.009)*
	PSL($R_1 T R_1 + pred$)	0.966 (0.009)	0.939 (0.011)	**0.952** (0.010)*
Names + personal + relational info (1st + 2nd degree)	LR	0.970 (0.012)	0.807 (0.051)	0.880 (0.032)
	SVMs	0.985 (0.006)	0.856 (0.029)	0.916 (0.019)
	LMTs	0.975 (0.008)	0.872 (0.016)	0.921 (0.011)
	PSL($R_{12} T R_1$)	0.964 (0.008)	0.935 (0.017)	0.949 (0.010)*
	PSL($R_{12} T R_1 + pred$)	0.970 (0.008)	0.943 (0.011)	**0.957** (0.009)*
	PSL($R_{12} T R_{12}$)	0.965 (0.008)	0.937 (0.015)	0.951 (0.009)*
	PSL($R_{12} T R_{12} + pred$)	0.969 (0.009)	0.943 (0.011)	0.956 (0.008)*

Numbers in parenthesis indicate standard deviations. Bold shows the best performance in terms of F-measure for each feature set. We denote by * statistical significance among the PSL model and the baselines at $\alpha = 0.05$ when using paired t test

Relational Info 1st **degree**, the PSL model that uses both relational and transitive relational similarity rules, PSL($R_1 T R_1$), outperforms LMTs by 4.4% for the NIH and 3.1% for the Wikidata. Finally, for the NIH dataset, for the experiment that additionally uses relational similarities of second degree, the best PSL model, PSL($R_{12} T R_{12}$), outperforms LMTs by 3%. When incorporating the predictions from the baseline algorithms (LR, SVMs, and LMTs) we observe that the performance of the PSL models further increases. We graphically present the superiority (in terms of F-measure) of the PSL models when compared to the baselines in all different sets of experiments in Figs. 5 and 6 for the NIH and the Wikidata datasets accordingly.

Table 3 Performance of PSL and baseline classifiers with varying types of rules/features for the Wikidata dataset

Experiment	Method	Wikidata Precision (SD)	Recall (SD)	F-measure (SD)
Names	LR	0.905 (0.015)	0.6598 (0.022)	0.720 (0.018)
	SVMs	0.941 (0.017)	0.607 (0.034)	0.738 (0.026)
	LMTs	0.926 (0.011)	0.660 (0.034)	0.770 (0.023)
	PSL(N)	0.868 (0.014)	0.754 (0.031)	0.806 (0.016)*
	PSL($N + pred$)	0.876 (0.017)	0.757 (0.031)	**0.811** (0.016)*
Names + personal info	LR	0.953 (0.015)	0.713 (0.032)	0.815 (0.022)
	SVMs	0.970 (0.011)	0.723 (0.034)	0.828 (0.023)
	LMTs	0.960 (0.014)	0.745 (0.037)	0.838 (0.022)
	PSL(P)	0.908 (0.026)	0.816 (0.042)	0.858 (0.016)*
	PSL($P + pred$)	0.928 (0.026)	0.839 (0.040)	**0.880** (0.017)*
Names + personal + relational info (1st degree)	LR	0.962 (0.013)	0.756 (0.028)	0.846 (0.015)
	SVMs	0.975 (0.012)	0.776 (0.035)	0.864 (0.019)
	LMTs	0.967 (0.015)	0.785 (0.037)	0.866 (0.019)
	PSL(R_1)	0.914 (0.017)	0.866 (0.031)	0.889 (0.011)*
	PSL($R_1 + pred$)	0.934 (0.018)	0.900 (0.023)	0.916 (0.011)*
	PSL($R_1 T R_1$)	0.917 (0.018)	0.878 (0.016)	0.897 (0.007)*
	PSL($R_1 T R_1 + pred$)	0.927 (0.018)	0.907 (0.019)	**0.917** (0.011)*

Numbers in parenthesis indicate standard deviations. Bold shows the best performance in terms of F-measure for each feature set. We denote by * statistical significance among the PSL model and the baselines at $\alpha = 0.05$ when using paired t test

Name similarities are not enough When we incorporate personal information similarities (**Names + Personal Info**) on top of the simple **Names** model that uses name similarities only, we get substantial improvements for the PSL model: 11% for the NIH and 5.2% for the Wikidata (absolute values) in F-measure. The improvement is evident in the graphs presented in Fig. 7 when comparing columns N and P for both datasets. The same observation is also true for all baseline models. For the NIH dataset, the SVMs get the most benefit out of the addition of personal information with an increase of 13.2%. For the Wikidata dataset, LR gets the most benefit with an increase of 9.5% for the F-measure.

First-degree relationships help most in low noise scenarios We found that reliable relational evidence improves performance, but noisy relationships can be detrimental. In the NIH dataset, incorporating first-degree relationships using the simple relational similarity function defined in Sect. 4.2.4 decreases performance slightly (1%) for the PSL model (also evident in Fig. 7a when comparing columns P and R_1). For LR, SVMs and LMTs, F-measure increases slightly (0.1%, 0.6%, and 0.1%, respectively). However, for the Wikidata, the addition of simple relational similarities increased F-measure by 3.1% for PSL(R_1). (This is shown in Fig. 7b when comparing columns P and R_1.) The same applies for the baseline models where we observe improvements of 2.8% for LMTs, 3.6% for SVMs, and 3.1% for LR. We believe that the difference in the effect of the simple relational features is due to the different noise in the two datasets. NIH is a real-world dataset with incomplete and unreliable information,

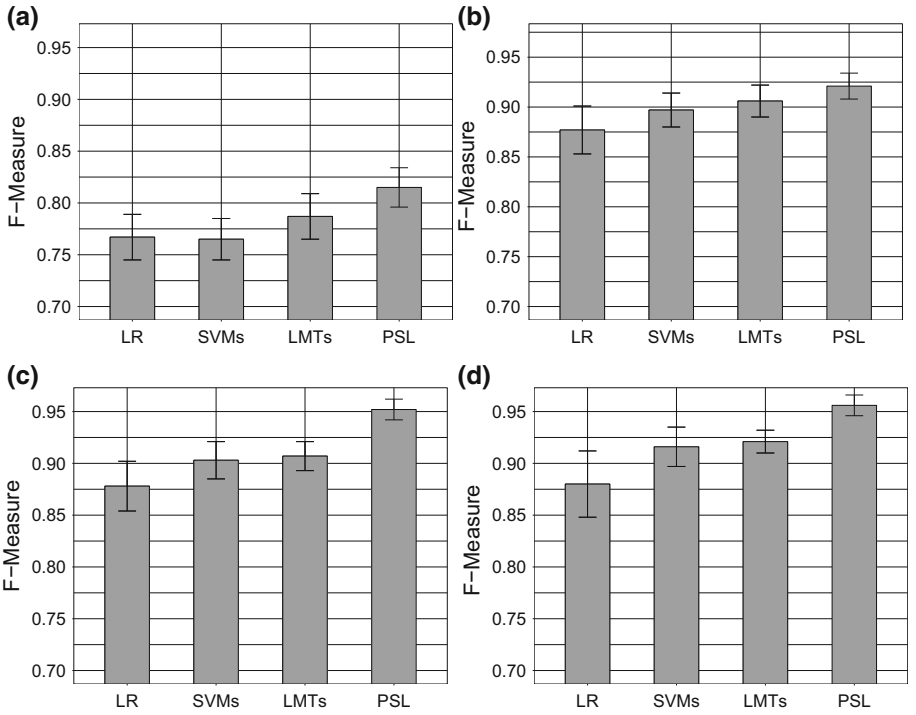

Fig. 5 NIH dataset: graphical representation of the performance (F-measure) of the baselines and the PSL models in different experimental setups. Standard deviations are shown around the top of each bar. For the PSL, we report the results for the models PSL($N + pred$), PSL($P + pred$), PSL($R_1 T R_1 + pred$), and PSL($R_{12} T R_{12} + pred$), respectively. **a** Names, **b** Names + Personal Info, **c** Names + Personal + Relational Info (1st degree) and **d** Names + Personal + Relational Info (1st + 2nd degree)

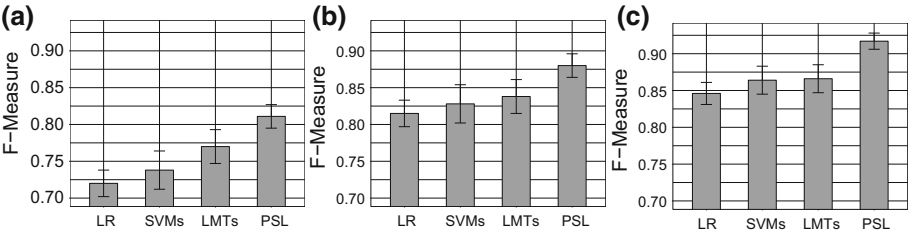

Fig. 6 Wikidata dataset: graphical representation of the performance (F-measure) of the baselines and the PSL models in different experimental setups. Standard deviations are shown around the top of each bar. For the PSL, we report the results for the models PSL($N + pred$), PSL($P + pred$), and PSL($R_1 T R_1 + pred$), respectively. **a** Names, **b** Names + Personal Info and **c** Names + Personal + Relational Info (1st degree)

while Wikidata is considered to contain no noise. As a result, we believe that both the baseline and PSL models are able to cope with the artificially introduced noise, while it is much more difficult to deal with real-world noisy data.

Collective relations yield substantial improvements When we incorporate collective, transitive relational rules to the PSL(R_1) model resulting to the PSL($R_1 T R_1$) model—a key differentiator of our approach—we observe a 4.1% improvement in F-measure for the NIH

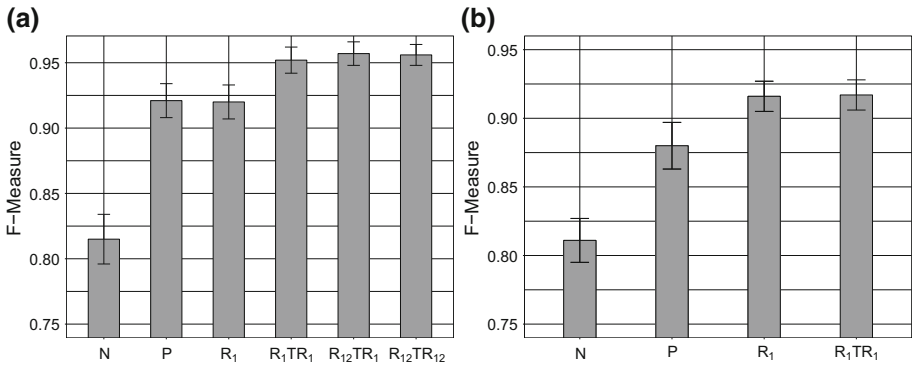

Fig. 7 Graphical representation of the performance of PSL in terms of F-measure with varying types of rules for (**a**) the NIH and (**b**) the Wikidata datasets. Standard deviations are shown around the top of each bar. All reported results are from PSL models that use the predictions from other algorithms. **a** NIH and **b** Wikidata

dataset. This is also evident in Fig. 7a when comparing columns R_1 and R_1TR_1. We note that this is a result of an increase of 5.1% for the recall and 2.1% for the precision. Adding collective rules allows decisions to be propagated between related pairs of mentions, exploiting statistical signals across the familial network to improve recall. The Wikidata also benefits from collective relationships, but the 0.8% improvement in F-measure score is much smaller. (For graphical illustration, there is no obvious improvement when comparing columns R_1 and R_1TR_1 of Fig. 7b.) For this cleaner dataset, we believe that simple relational similarity rules were informative enough to dampen the impact of transitive relational similarity rules. As a result, these rules are not as helpful as in the more noisy NIH dataset.

Second-degree similarities improve performance for the baselines The addition of simple relational similarities from second-degree relationships, such as those available in the NIH dataset, yield improvements in all baseline models. When adding second-degree relationships, we observe a pronounced increase in the F-measure for two baselines (1.6% for LMTs and 1.3% for SVMs), while LR has a small increase of 0.2%. For our approach, PSL($R_{12}TR_1$), slightly decreases the PSL(R_1TR_1) model (0.2% for F-measure), while the addition of second-degree transitive relational features (model PSL($R_{12}TR_{12}$)) improves slightly the performance by 0.2%.

Predictions from other algorithms always improve performance In all our experiments, we ran different versions of the PSL models that included or omitted the predictions from the baselines, i.e., LR, SVMs, LMTs (discussed in Sect. 4.2.8). We observe that the addition of the predictions of the other algorithms always increases the performance of the PSL models. More specifically, for the NIH dataset, the addition of the predictions from LR, SVMs, and LMTs slightly increases the F-measure of the PSL models. In particular, F-measure increases by 0.5% for the experiment **Names** and 0.1% for the experiment **Names + Personal Info**. Also, the experiment PSL($R_1 + pred$) improves the F-measure of the experiment PSL(R_1) by 1.0%, and the experiment PSL($R_1TR_1 + pred$) slightly improves the F-measure of the experiment PSL(R_1TR_1) by 0.1%. For the case of the experiment PSL($R_1 + pred$), we can see that its performance (F-measure = 0.910) is very close to the performance of the baselines (e.g., the F-measure for the LMTs is 0.907). As a result, adding the baselines helps the PSL model to better distinguish the true positives and true negatives. However, in the case of the model PSL(R_1TR_1) we can see that there is a clear difference between the PSL model and the baselines, so for this experiment adding the predictions

of those cannot improve at a bigger scale the performance of the PSL model. Last, for the experiment **Names + Personal + Relational Info** (1st + 2nd **degree)** we observe that adding the predictions from the other algorithms slightly increases the F-measure by 0.8% for the experiment PSL($R_{12}TR_1 + pred$) and 0.5% for the experiment PSL($R_{12}TR_{12} + pred$). In all cases, we observe that the increase in F-measure is the result of an increase in both the precision and the recall of the model. (The only case that we observe a small decrease in the recall is the experiment **Names + Personal Info**.) For the Wikidata dataset, we observe that the F-measure improves significantly in all experiments when adding the predictions from the baselines. This is a result of the increase of both the precision and the recall. More specifically, we observe the following increases for the F-measure: 0.5% for the experiment **Names**, 2.2% for the experiment **Names + Personal Info**, 2.7% and 2.0% for the two versions of the experiment **Names + Personal + Relational Info** (1st **degree**).

Precision-recall balance depends on the chosen threshold As we discussed in Sect. 4.3 for the PSL model we choose the optimal threshold to maximize the F-measure score. This learned threshold achieves a precision-recall balance that favors recall at the expense of precision. For both datasets, our model's recall is significantly higher than all the baselines in all the experiments. However, since PSL outputs soft truth values, changing the threshold selection criteria in response to the application domain (e.g., prioritizing cleaner matches over coverage) can allow the model to emphasize precision over recall.

Matching restrictions always improves F-measure We note that valid solutions in our entity resolution setting require that an entity matches at most one entity in another ego-centric network. To enforce this restriction, we apply a 1–1 matching algorithm on the raw output of all models (Sect. 4.4). Applying matching restrictions adjusts the precision–recall balance of all models. For both PSL and the baselines across both datasets, when applying the 1–1 matching restriction algorithm, we observe a sizable increase in precision and a marginal drop in recall. This pattern matches our expectations, since the algorithm removes predicted co-references (harming recall) but is expected to primarily remove false-positive pairs (helping precision). Overall, the application of the 1–1 matching restrictions improves the F-measure for all algorithms and all datasets. Since the results before the 1–1 matching do not represent valid solutions and it is not straightforward to compare across algorithms, we do not report them here.

PSL is scalable to the number of instances, based on empirical results One motivation for choosing PSL to implement our entity resolution model was the need to scale to large datasets. To empirically validate the scalability of our approach, we vary the number of instances, consisting of pairs of candidate co-referent entities, and measure the execution time of inference. In Fig. 8, we plot the average execution time relative to the number of candidate entity pairs. Our results indicate that our model scales almost linearly with respect to the number of comparisons. For the NIH dataset, we note one prominent outlier, for a family with limited relational evidence resulting in lower execution time. Conversely, for the Wikidata, we observe two spikes which are caused by families that contain relatively dense relational evidence compared to similar families. We finally note that we expect these scalability results to hold as the datasets get bigger since the execution time depends on the number of comparisons and the number of relations per family.

5.4 Effect of string similarity functions

In Sect. 4.2.2, we discussed that PSL can easily incorporate a variety of string similarity functions. In the basic experiments (Sect. 5.3), all models (PSL and baselines) used the

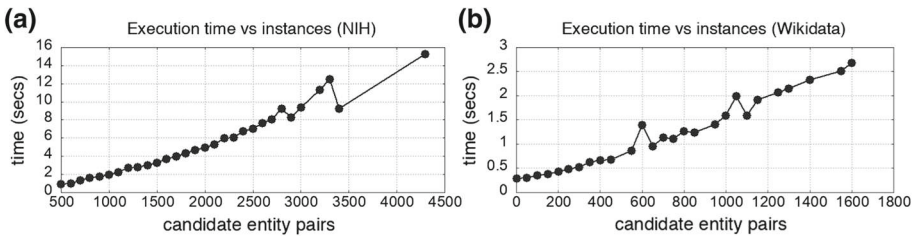

Fig. 8 An analysis of the scalability of our system (**a** is for the NIH and **b** for the Wikidata). As the number of potentially co-referent entity pairs increases, the execution time of our model grows linearly for both datasets

Levensthein and Jaro–Winkler string similarity functions. In this section, we experiment with a wider set of string similarity functions and simple combinations of them in order to study how such different functions can affect performance. More specifically, for all the models (PSL and baselines) we ran the following experiments:

- **Levenshtein** ($\boldsymbol{L}$) We use first, middle, and last name similarities computed using the Levenshtein string similarity function only.
- **Jaro–Winkler** ($\boldsymbol{JW}$) We add Jaro–Winkler similarities.
- **Monge-Elkan** ($\boldsymbol{ME}$) We add Monge-Elkan similarities.
- **Soundex** ($\boldsymbol{S}$) We add Soundex similarities.
- **Jaro** ($\boldsymbol{J}$): We add Jaro similarities.
- **max**($\boldsymbol{L, JW, ME, S, J}$) We combine the string similarity functions by using the maximum value of all the similarity functions.
- **min**($\boldsymbol{L, JW, ME, S, J}$) We combine the string similarity functions by using the minimum value of all the similarity functions.

We note that for the PSL, we run the version PSL(N) and not the version PSL($N + pred$), i.e., we do not use the predictions from the other models in our PSL model. We present the results in Table 4 for the NIH dataset. As we discussed, for the Wikidata dataset we introduced artificial noise to all the similarities, so we focus on the NIH dataset to get a clear picture of the performance of the similarity functions. Here is a summary of the results from Table 4:

The performance of the models changes when the string similarity functions change For PSL, the difference between the model that performs best and the model that performs worst is 3% absolute value, for LMTs 2.5%, for SVMs 4.2%, and for LR 0.9%.

The setting of string similarity functions that performs best is different for each model For PSL, the best model uses Levensthein, Jaro–Winkler, and Monge-Elkan. For LMTs, the best model uses the min(L, JW, ME, S, J), for SVMs the best model uses the Levensthein, and for LR the best model uses Levensthein, Jaro–Winkler, Monge-Elkan, and Soundex.

PSL models outperform baselines In each experiment, PSL outperforms all the baselines using the same string similarity functions. With one exception (for max(L, JW, ME, S, J)) PSL statistically significantly outperforms the baselines that use the same string similarity functions for the F-measure at $\alpha = 0.05$ when using paired t test. For the experiment max(L, JW, ME, S, J), LMTs outperform PSL (by 0.5% absolute value), but this difference is not considered statistically significant. For graphical illustration, Fig. 9 shows the F-measure for the baselines and the PSL model for the setting that each model performed the best. For example, for PSL, we plot the F-measure when using Levensthein, Jaro–Winkler, and Monge-Elkan while for LMTs, we plot the F-measure when using the min(L, JW, ME, S, J).

Table 4 Performance of PSL and baseline classifiers for the experiment that uses only name similarities with varying the string similarity functions used

Method	String functions	NIH Precision (SD)	Recall (SD)	F-measure (SD)
PSL	Levenshtein (L)	0.850 (0.017)	0.757 (0.044)	0.801 (0.029)
	+ Jaro–Winkler (JW)	0.866 (0.021)	0.761 (0.028)	0.810 (0.023)
	+ Monge-Elkan (ME)	0.871 (0.025)	0.766 (0.035)	0.815 (0.028)
	+ Soundex (S)	0.866 (0.019)	0.765 (0.034)	0.812 (0.024)
	+ Jaro (J)	0.868 (0.029)	0.762 (0.035)	0.812 (0.031)
	max(L, JW, ME, S, J)	0.834 (0.025)	0.741 (0.027)	0.785 (0.024)
	min(L, JW, ME, S, J)	0.861 (0.019)	0.752 (0.025)	0.803 (0.021)
LMTs	Levenshtein (L)	0.874 (0.025)	0.699 (0.031)	0.776 (0.026)
	+ Jaro–Winkler (JW)	0.874 (0.002)	0.717 (0.027)	0.787 (0.022)
	+ Monge-Elkan (ME)	0.865 (0.026)	0.714 (0.031)	0.782 (0.027)
	+ Soundex (S)	0.862 (0.026)	0.715 (0.027)	0.782 (0.024)
	+ Jaro (J)	0.854 (0.028)	0.711 (0.028)	0.776 (0.024)
	max(L, JW, ME, S, J)	0.848 (0.028)	0.739 (0.032)	0.789 (0.029)
	min(L, JW, ME, S, J)	0.870 (0.026)	0.681 (0.037)	0.764 (0.030)
SVMs	Levenshtein (L)	0.870 (0.027)	0.716 (0.029)	0.785 (0.025)
	+ Jaro–Winkler (JW)	0.870 (0.022)	0.683 (0.027)	0.765 (0.020)
	+ Monge-Elkan (ME)	0.867 (0.020)	0.675 (0.043)	0.759 (0.033)
	+ Soundex (S)	0.870 (0.030)	0.68 (0.038)	0.763 (0.031)
	+ Jaro (J)	0.870 (0.023)	0.679 (0.027)	0.763 (0.023)
	max(L, JW, ME, S, J)	0.834 (0.035)	0.719 (0.033)	0.772 (0.033)
	min(L, JW, ME, S, J)	0.858 (0.021)	0.656 (0.038)	0.743 (0.030)
LR	Levenshtein (L)	0.871 (0.026)	0.689 (0.031)	0.769 (0.024)
	+ Jaro–Winkler (JW)	0.870 (0.022)	0.683 (0.027)	0.765 (0.020)
	+ Monge-Elkan (ME)	0.870 (0.024)	0.688 (0.026)	0.768 (0.021)
	+ Soundex (S)	0.872 (0.024)	0.694 (0.026)	0.772 (0.021)
	+ Jaro (J)	0.872 (0.023)	0.693 (0.027)	0.772 (0.021)
	max(L, JW, ME, S, J)	0.827 (0.027)	0.715 (0.033)	0.767 (0.030)
	min(L, JW, ME, S, J)	0.871 (0.024)	0.697 (0.029)	0.763 (0.023)

Numbers in parenthesis indicate standard deviations. For all experiments apart from one (max(L, JW, ME, S, J)) PSL statistically significantly outperforms the baselines that use the same string similarity functions for the f-measure at $\alpha = 0.05$ when using paired t test

5.5 Effect of noise level

As we discussed, to simulate the noisy conditions of real-world datasets, we introduced additive Gaussian noise to all the similarity scores (names, personal information, relational information) of the Wikidata dataset drawn from a $N(0, 0.16)$ distribution. In this section, we experiment with varying the introduced noise. For all experiments, for all models (both PSL and baselines), we additionally ran experiments when introducing noise from the following distributions: $N(0, 0.01)$, $N(0, 0.09)$, $N(0, 0.49)$, $N(0, 0.81)$. We present our results in Table 10 where we plot the average F-measure computed over fivefold cross-validation with respect to the noise added to the similarities. For the experiments of the PSL, we use the

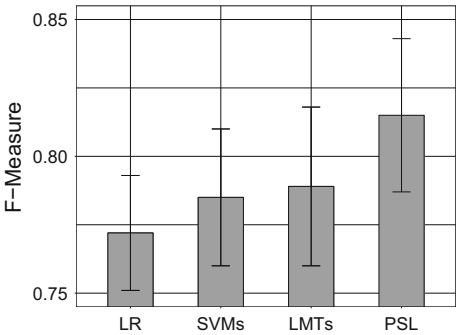

Fig. 9 NIH dataset: graphical representation of the performance (F-measure) of the baselines and the PSL model for the combination of string similarities that each model performs the best. For the PSL, we plot the F-measure when using Levensthein, Jaro–Winkler, and Monge-Elkan. For LMTs, we report results when using the $\min(L, JW, ME, S, J)$, for SVMs we report results when using the Levensthein, and for LR we report results when using Levensthein, Jaro–Winkler, Monge-Elkan, and Soundex. Standard deviations are shown around the top of each bar

following versions: for the experiment **Names**, we use the model PSL(N), for the experiment **Names + Personal Info** the model PSL(P), and for the experiment **Names + Personal + Relational Info** (1st degree) the model PSL(R_1TR_1). In other words, we do not include the predictions from the other baseline models—but we expect them to perform better than the ones we report here since all the experiments that include the predictions outperform the experiments that do not include the predictions for the Wikidata dataset (Table 3). As expected, when the noise increases, then the F-measure decreases and this is true for all models. Another observation is that with very small amount of noise (drawn from $N(0, 0.01)$ or $N(0, 0.09)$ distributions) all the models perform similarly. However, when increasing the noise (drawn from $N(0, 0.16)$, $N(0, 0.49)$, or $N(0, 0.81)$ distributions), then the difference between the models becomes more pronounced. When noise is drawn from these distributions, PSL consistently performs best for all experiments (**Names**, **Names + Personal Info**, **Names + Personal + Relational Info**). This difference is statistically significantly better at $\alpha = 0.05$ when using paired t tests for all experiments. Among the baselines, LMTs perform the best, followed by SVMs, and finally LR.

5.6 Performance with varying number of predicted matches

In this section, our goal is to study the performance of the PSL models and the baseline classifiers with respect to the threshold used for classifying the instances. As we discussed, PSL learns the threshold using a validation set. The baseline classifiers also use some internal threshold to determine whether each pair is co-referent. Since the learned thresholds are different for each model, it would be unfair to plot the F-measure with respect to the threshold to compare the methods. Similarly, precision–recall curves in this setting would not be informative: since the values of the thresholds are not related, it does not make sense to report that a method A is better than method B at a particular threshold. To overcome the above issues and make a fair comparison of the methods we follow the related work [4,16,27] and choose the threshold so that each method produces the same number of predicted matches (i.e., true positives and false positives). To this end, we compute the F-measure when varying the number of predicted matches for each algorithm. For each value of the predicted matches,

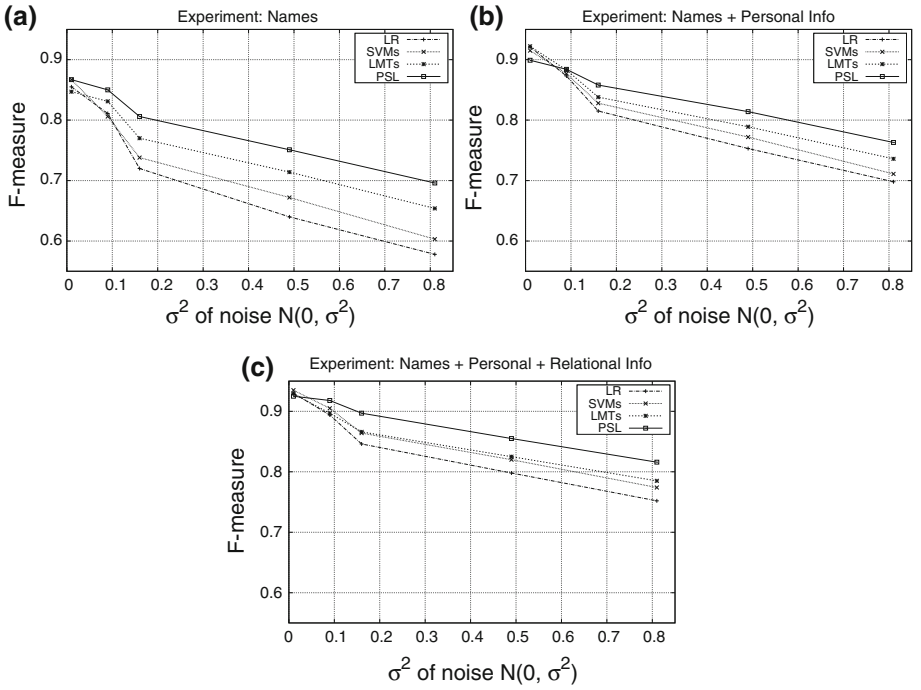

Fig. 10 An analysis of the performance of the models (PSL and baselines) when varying the noise in the similarities for the Wikidata dataset (for the experiments **a Names**, **b Names + Personal Info**, **c Names + Personal + Relational Info** (1st **degree**)). We report average F-measure scores from a fivefold cross-validation. As the noise increases, the F-measure decreases. For the minimum amount of noise all the models perform similarly. However, as the noise increases, the difference in the performance becomes more evident

we compute the precision as the ratio of the true positives over the true positives and false positives in the predicted matches, the recall as the ratio of the true positives over the true positives and false negatives in the predicted matches, and the F-measure as the weighted balance of the precision and recall. We present the results for the NIH dataset in Table 11 and for the Wikidata in Table 12. In all experiments, we report the results of the PSL models that include the predictions of the other classifiers. More specifically, we report the results of the models: PSL($N + pred$), PSL($P + pred$), PSL($R_1 T R_1 + pred$), and PSL($R_{12} T R_{12} + pred$) (only for the NIH dataset).

For the NIH dataset, for the experiment **Names**, we observe that PSL consistently outperforms all the baselines when the number of matches is smaller than 950. However, when the number of matches is larger than 1000, the performance of the PSL is lower than the baselines. For all the other experiments (**Names + Personal Info**, **Names + Personal + Relational Info** (1st degree), and **Names + Personal + Relational Info** (1st + 2nd degree)) all models perform similarly when the number of predicted matches is smaller than 800. When the number of predicted matches is larger than 800, we can see that the PSL models consistently outperform all the baselines. For the Wikidata dataset, for all the experiments we observe that all the models perform similarly for small number of matches (up to 2500). However, when the number of matches increases (i.e., larger than 2500), then we observe a clear win of the PSL models.

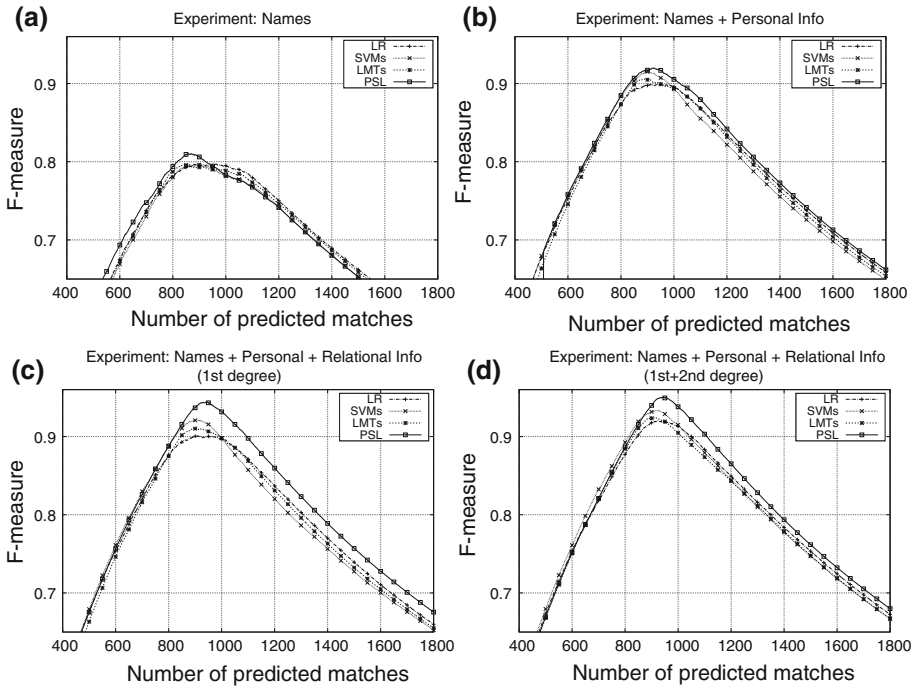

Fig. 11 An analysis of the performance of the models (PSL and baselines) with respect to the number of the predicted matches for the NIH dataset (for the experiments **a Names**, **b Names + Personal Info**, **c Names + Personal + Relational Info** (1st **degree**) and **d Names + Personal + Relational Info** (1st + 2nd **degree**)). We report average F-measure scores from a fivefold cross-validation

6 Related work

There is a large body of prior work in the general area of entity resolution [8]. In this work, we propose a collective approach that makes extensive use of relational data. In the following, we review collective relational entity resolution approaches which according to [31] can be either iterative or purely collective.

For the iterative collective classification case, [5] propose a method based on greedy clustering over the relationships. This work considers only one single relation type, while we consider several types. [10] propose another iterative approach which combines contextual information with similarity metrics across attributes. In our approach, we perform both reference and relation enrichment, by applying inversion and imputation. Finally, [20] propose an approach for the reference disambiguation problem where the entities are already known. In our case, we do not know the entities beforehand.

In the case of purely collective approaches, [1] propose the Dedupalog framework for collective entity resolution with both soft and hard constraints. Users define a program with hard and soft rules, and the approach produces a clustering such that no hard constraints are violated and the number of violated soft constraints is minimized. Dedupalog is well suited for datasets having the need to satisfy several matching restrictions. In our case, we have several soft rules with a smaller number of constraints. In another approach, [9] design a conditional random field model incorporating relationship dependencies and propose an algorithm that jointly performs entity resolution over the model. In this work too, the

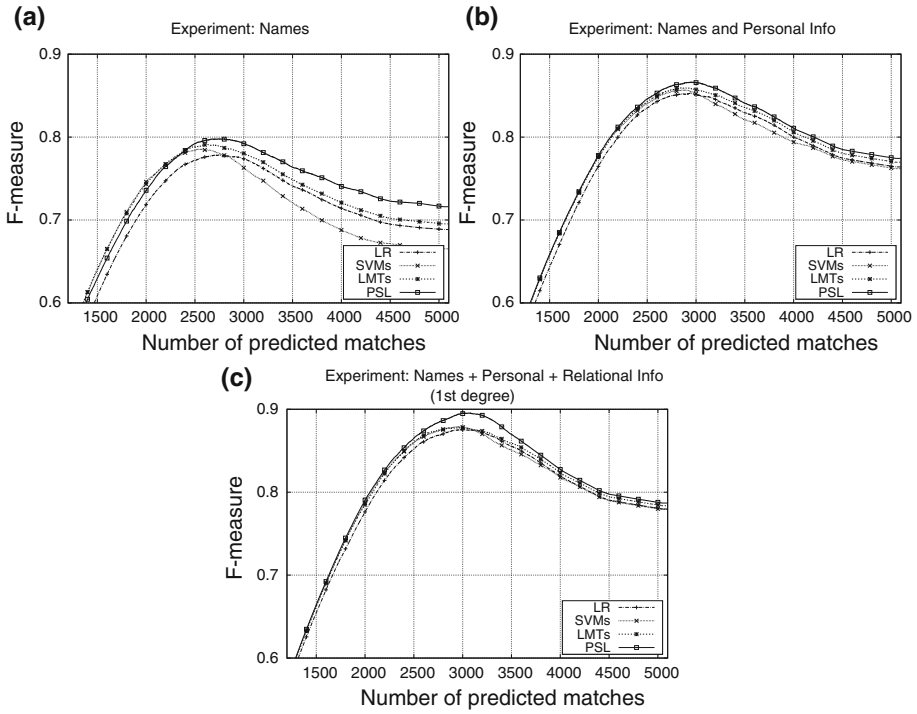

Fig. 12 An analysis of the performance of the models (PSL and baselines) with respect to the number of the predicted matches for the Wikidata dataset (for the experiments **a Names**, **b Names + Personal Info**, **c Names + Personal + Relational Info** (1st **degree**)). We report average F-measure scores from a fivefold cross-validation

number of relationship types considered is small. Finally, [32] propose a generalization of the Fellegi–Sunter model [13] that combines first-order logic and Markov random fields to perform collective classification. The proposed Markov logic networks (MLNs) operate on undirected graphical models using a first-order logic as their template language, similar to PSL. However, the predicates take only boolean values, while in PSL the predicates take soft truth values in the range [0, 1]. Soft truth values are more appropriate in the entity resolution problem setting for two reasons: first, they can better capture notion of similarity (such as name similarity) and second, the predictions can be interpreted as probabilities (in the range [0, 1]) which is convenient when applying the matching restrictions algorithm. (We note again that this algorithm requires as input a ranked list.) Finally, extensive experiments from the related work [2,3] have shown that HL-MRFs can achieve improved performance in much less time compared to MLNs. As HL-MRFs are faster and their output is directly usable from a matching restriction approach that is needed in our scenario, we do not compare our approach to MLNs.

Overall, the purely collective approaches come with a high computational cost for performing probabilistic inference. As a result, they cannot scale to large datasets unless we use techniques that make the EM algorithm scalable [31]. Our approach uses PSL which ensures scalable and exact inference by solving a convex optimization problem in parallel. Speed and scalability is of paramount importance in entity resolution and in particular when we run the prediction task collectively using transitivity and bijection rules.

Regarding the problem of entity resolution in familial networks, we recently proposed a first approach [21]. The problem setting is the same as in the current work, but the approach is non-collective using well-studied classifiers enhanced with features capturing relational similarity. In this work, we propose a more sophisticated collective approach to the familial entity resolution problem.

Additionally, there are some works from the ontology alignment and knowledge graph identification domains that are close to our approach. [33] propose a probabilistic approach for ontology alignment. The tool accepts as input two ontologies and distinguishes the same relations, classes, and instances. As a result, the approach does not take into account transitivity and bijection constraints, which are key features in the familial networks in order both to provide a valid solution and to improve performance. In another approach, [30] use PSL to design a general mechanism for entity resolution in knowledge graphs, a setting with a similarly rich relational structure. Their work considers entity resolution within and between graphs and provides general templates for using attributes and relationships in non-collective and collective rules. However, familial networks have unique characteristics and constraints that differ substantially from knowledge graphs, and in particular, they do not explicitly consider the problem of entity resolution across several subgraphs. Finally, as we mentioned in Introduction, this work is an extended version of our recent work [22].

7 Conclusions and future work

Entity resolution in familial networks poses several challenges, including heterogeneous relationships that introduce collective dependencies between decisions and inaccurate attribute values that undermine classical approaches. In this work, we propose a scalable collective approach based on probabilistic soft logic that leverages attribute similarities, relational information, logical constraints, and predictions from other algorithms. A key differentiator of our approach is the ability to support bijection and different types of transitive relational rules that can model the complex familial relationships. Moreover, our method is capable of using training data to learn the weight of different similarity scores and relational features, an important ingredient of relational entity resolution. In our experimental evaluation, we demonstrated that our framework can effectively combine different signals, resulting in improved performance over state-of-the-art approaches on two datasets. In our experimental evaluation, we also showed that, in most cases, our model outperforms the baselines for a varying set of similarity functions and for varying levels of noise. Additionally, the experimental evaluation showed that the PSL models outperform the baselines when we fix the number of predicted matches.

In this paper, we motivate the importance of our approach with an application for resolving mentions in healthcare records. However, the problem of entity resolution in richly structured domains has many additional applications. For example, many companies[5] provide genealogical discovery services, which require a similar entity resolution process. We also foresee applications in social networks, where the problem of linking user accounts across several social platforms in the presence of a diverse set of relationships (e.g., friends, followers, followees, family cycles, shared groups), ambiguous names, and collective constraints such as bijection and transitivity, provide a similar set of opportunities and challenges.

In future work, we plan to apply our approach to a broader set of problems and discuss general strategies for multi-relational entity resolution. Additionally, we plan to explore

[5] ancestry.com, genealogy.com, familysearch.org, 23andMe.com.

structured output learning techniques [28] inside PSL. Such techniques can directly consider the matching constraints during the learning phase instead of post-processing the classification results. We also plan to explore temporal relations, e.g., ex-wife, and more complex relationships, e.g., adopted child. Finally, in certain cases, we might inadvertently introduce inaccurate relations when following the approach of Sect. 3. To address this, we plan to expand our work to account for uncertainty in the relational normalization step by assuming a probability assigned to each populated relationship instead of the hard values that we currently assign.

Acknowledgements We would like to thank Peter Christen and Jon Berry for insightful comments on this paper. This work was partially supported by the National Science Foundation Grants IIS-1218488, CCF-1740850, and IIS-1703331 and by the National Human Genome Research Institute Division of Intramural Research at the National Institutes of Health (ZIA HG2000397 and ZIA HG200395, Koehly PI). We would also like to thank the Sandia LDRD (Laboratory-Directed Research and Development) program for support. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the National Institutes of Health, or the Sandia Labs.

APPENDIX: PSL model rules

Name similarity rules

$\text{SIMFIRSTNAME}_{JaroWinkler}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMMAIDENNAME}_{JaroWinkler}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMLASTNAME}_{JaroWinkler}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{SIMFIRSTNAME}_{JaroWinkler}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{SIMMAIDENNAME}_{JaroWinkler}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{SIMLASTNAME}_{JaroWinkler}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMFIRSTNAME}_{Levenshtein}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMMAIDENNAME}_{Levenshtein}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMLASTNAME}_{Levenshtein}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{SIMFIRSTNAME}_{Levenshtein}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{SIMMAIDENNAME}_{Levenshtein}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{SIMLASTNAME}_{Levenshtein}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$

Personal information similarity rules

$\text{KNOWNAGE}(\text{m}_1) \wedge \text{KNOWNAGE}(\text{m}_2) \wedge \text{SIMAGE}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{KNOWNAGE}(\text{m}_1) \wedge \text{KNOWNAGE}(\text{m}_2) \wedge \neg\text{SIMAGE}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$
$\neg\text{EQLIVING}(\text{m}_1, \text{m}_2) \Rightarrow \neg\text{SAME}(\text{m}_1, \text{m}_2)$

Relational similarity rules of 1st degree

$\text{SIMMOTHER}(\text{m}_a, \text{m}_b) \wedge \text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMFATHER}(\text{m}_a, \text{m}_b) \wedge \text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMDAUGHTER}(\text{m}_a, \text{m}_b) \wedge \text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMSON}(\text{m}_a, \text{m}_b) \wedge \text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMSISTER}(\text{m}_a, \text{m}_b) \wedge \text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMBROTHER}(\text{m}_a, \text{m}_b) \wedge \text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$
$\text{SIMSPOUSE}(\text{m}_a, \text{m}_b) \wedge \text{EQGENDER}(\text{m}_1, \text{m}_2) \Rightarrow \text{SAME}(\text{m}_1, \text{m}_2)$

Relational similarity rules of 2nd degree

$\textsc{SimGrandMother}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$
$\textsc{SimGrandFather}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$
$\textsc{SimGrandDaughter}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$
$\textsc{SimGrandSon}(\mathtt{m}_a, \mathtt{m}_b) \wedge$
$\textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$
$\textsc{SimAunt}(\mathtt{m}_a, \mathtt{m}_b) \wedge$
$\textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$
$\textsc{SimUncle}(\mathtt{m}_a, \mathtt{m}_b) \wedge$
$\textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$
$\textsc{SimNiece}(\mathtt{m}_a, \mathtt{m}_b) \wedge$
$\textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$
$\textsc{SimNephew}(\mathtt{m}_a, \mathtt{m}_b) \wedge$
$\textsc{eqGender}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2)$

Transitive relational (similarity) rules of 1st degree

$\textsc{Rel}(\mathtt{Mother}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Mother}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Father}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Father}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Spouse}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Spouse}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Spouse}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Spouse}, \mathtt{m}_2, \mathtt{m}_b) \wedge \neg\textsc{Same}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \neg\textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Daughter}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Daughter}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Son}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Son}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Sister}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Sister}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Brother}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Brother}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$

Transitive relational (similarity) rules of 2nd degree

$\textsc{Rel}(\mathtt{GrandMother}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{GrandMother}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{GrandFather}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{GrandFather}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{GrandDaughter}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{GrandDaughter}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{GrandSon}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{GrandSon}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Aunt}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Aunt}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Uncle}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Uncle}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Niece}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Niece}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$
$\textsc{Rel}(\mathtt{Nephew}, \mathtt{m}_1, \mathtt{m}_a) \wedge \textsc{Rel}(\mathtt{Nephew}, \mathtt{m}_2, \mathtt{m}_b) \wedge \textsc{Same}(\mathtt{m}_1, \mathtt{m}_2) \wedge \textsc{SimName}(\mathtt{m}_a, \mathtt{m}_b) \wedge \textsc{eqGender}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \textsc{Same}(\mathtt{m}_a, \mathtt{m}_b)$

Bijection and transitivity rules

$\text{FR}(\mathtt{m}_a, \mathtt{R}_1) \wedge \text{FR}(\mathtt{m}_b, \mathtt{R}_2) \wedge \text{FR}(\mathtt{m}_c, \mathtt{R}_2) \wedge \text{SAME}(\mathtt{m}_a, \mathtt{m}_b) \Rightarrow \neg\text{SAME}(\mathtt{m}_a, \mathtt{m}_c)$
$\text{FR}(\mathtt{m}_a, \mathtt{R}_1) \wedge \text{FR}(\mathtt{m}_b, \mathtt{R}_2) \wedge \text{FR}(\mathtt{m}_c, \mathtt{R}_3) \wedge \text{SAME}(\mathtt{m}_a, \mathtt{m}_b) \wedge \text{SAME}(\mathtt{m}_b, \mathtt{m}_c) \Rightarrow \text{SAME}(\mathtt{m}_a, \mathtt{m}_c)$

Rules to leverage existing classification algorithms

$\text{SAMELR}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$
$\neg\text{SAMELR}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$
$\text{SAMESVMS}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$
$\neg\text{SAMESVMS}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$
$\text{SAMELMTS}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$
$\neg\text{SAMELMTS}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$
$\text{SAMELR}(\mathtt{m}_1, \mathtt{m}_2) \wedge \text{SAMESVMS}(\mathtt{m}_1, \mathtt{m}_2) \wedge \text{SAMELMTS}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$
$\neg\text{SAMELR}(\mathtt{m}_1, \mathtt{m}_2) \wedge \neg\text{SAMESVMS}(\mathtt{m}_1, \mathtt{m}_2) \wedge \neg\text{SAMELMTS}(\mathtt{m}_1, \mathtt{m}_2) \Rightarrow \neg\text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$

Prior rule

$\neg\text{SAME}(\mathtt{m}_1, \mathtt{m}_2)$

References

1. Arasu A, Ré C, Suciu D (2009) Large-scale deduplication with constraints using dedupalog. In: IEEE international conference on data engineering (ICDE)
2. Bach S, Broecheler M, Huang B, Getoor L (2017) Hinge-loss markov random fields and probabilistic soft logic. J Mach Learn Res (JMLR) 18(109):1–67
3. Bach S, Huang B, London B, Getoor L (2013) Hinge-loss Markov random fields: convex inference for structured prediction. In: Uncertainty in artificial intelligence (UAI)
4. Belin T, Rubin D (1995) A method for calibrating false-match rates in record linkage. J Am Stat Assoc 90(430):694–707
5. Bhattacharya I, Getoor L (2007) Collective entity resolution in relational data. ACM Trans Knowl Discov Data (TKDD) 1(1). https://doi.org/10.1145/1217299.1217304
6. Cessie S, Houwelingen J (1992) Ridge estimators in logistic regression. Appl Stat 41(1):191–201
7. Chang C, Lin C (2011) Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol (TIST) 2(3):2:27:1–27:27
8. Christen P (2012) Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. Springer, Berlin
9. Culotta A, McCallum A (2005) Joint deduplication of multiple record types in relational data. In: ACM international conference on information and knowledge management (CIKM)
10. Dong X, Halevy A, Madhavan J (2005) Reference reconciliation in complex information spaces. In: ACM special interest group on management of data (SIGMOD)
11. Driessens K, Reutemann P, Pfahringer B, Leschi C (2006) Using weighted nearest neighbor to benefit from unlabeled data. In: Pacific-Asia conference on knowledge discovery and data mining (PAKDD)
12. Efremova J, Ranjbar-Sahraei B, Rahmani H, Oliehoek F, Calders T, Tuyls K, Weiss G (2015) Multi-source entity resolution for genealogical data, population reconstruction
13. Fellegi P, Sunter B (1969) A theory for record linkage. J Am Stat Assoc 64(328):1183–1210
14. Frank E, Hall M, Witten I (2016) The WEKA Workbench. In: Gray J (ed) Practical machine learning tools and techniques. Morgan Kaufmann, Burlington (Online appendix for data mining)
15. Goergen A, Ashida S, Skapinsky K, de Heer H, Wilkinson A, Koehly L (2016) Knowledge is power: improving family health history knowledge of diabetes and heart disease among multigenerational mexican origin families. Public Health Genomics 19(2):93–101
16. Hand D, Christen P (2017) A note on using the f-measure for evaluating record linkage algorithms. Stat Comput 28(3):539–547
17. Hanneman R, Riddle F (2005) Introduction to social network methods. University of California, Riverside
18. Harron K, Wade A, Gilbert R, Muller-Pebody B, Goldstein H (2014) Evaluating bias due to data linkage error in electronic healthcare records. BMC Med Res Methodol 14:36
19. Hsu C, Chang C, Lin C (2003) A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University

20. Kalashnikov D, Mehrotra S (2006) Domain-independent data cleaning via analysis of entity-relationship graph. ACM Trans Database Syst (TODS) 31(2):716–767
21. Kouki P, Marcum C, Koehly L, Getoor L (2016) Entity resolution in familial networks. In: SIGKDD conference on knowledge discovery and data mining (KDD), workshop on mining and learning with graphs
22. Kouki P, Pujara J, Marcum C, Koehly L, Getoor L (2017) Collective entity resolution in familial networks. In: IEEE international conference on data mining (ICDM)
23. Landwehr N, Hall M, Frank E (2005) Logistic model trees. Mach Learn 95(1–2):161–205
24. Li X, Shen C (2008) Linkage of patient records from disparate sources. Stat Methods Med Res 22(1):31–8
25. Lin J, Marcum C, Myers M, Koehly L (2017) Put the family back in family health history: a multiple-informant approach. Am J Prev Med 5(52):640–644
26. Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv 33(1):31–88
27. Newcombe H (1988) Handbook of record linkage: methods for health and statistical studies, administration, and business. Oxford University Press Inc, Oxford
28. Nowozin S, Gehler P, Jancsary J, Lampert C (2014) Advanced structured prediction. The MIT Press, Cambridge
29. Platanios E, Poon H, Mitchell T, Horvitz E (2017) Estimating accuracy from unlabeled data: a probabilistic logic approach. In: Conference on neural information processing systems (NIPS)
30. Pujara J, Getoor L (2016) Generic statistical relational entity resolution in knowledge graphs. In: International joint conference on artificial intelligence (IJCAI), workshop on statistical relational artificial intelligence (StarAI)
31. Rastogi V, Dalvi N, Garofalakis M (2011) Large-scale collective entity matching. In: International conference on very large databases (VLDB)
32. Singla P, Domingos P (2006) Entity resolution with Markov logic. In: IEEE international conference on data mining (ICDM)
33. Suchanek F, Abiteboul S, Senellart P (2011) Paris: probabilistic alignment of relations, instances, and schema. In: Proceedings of the very large data bases endowment (PVLDB), vol 5(3)
34. Winkler W (2006) Overview of record linkage and current research directions. Technical report, US Census Bureau

Pigi Kouki earned her PhD from the Department of Technology and Information Management at the University of California Santa Cruz. Her research interests include entity resolution in relational networks, hybrid recommender systems, and explainable and fair decision support systems. Her work is published in RecSys and in ICDM. She has earned two masters degrees, one at the University of California Santa Cruz and one at the University of Athens, Greece. Her BS is in Computer Science at the University of Athens, Greece.

Jay Pujara is a research assistant professor of Computer Science at the University of Southern California whose principal areas of research are machine learning, artificial intelligence, and data science. He completed a postdoc at UC Santa Cruz, earned his Ph.D. at the University of Maryland, College Park, and received his M.S. and B.S. at Carnegie Mellon University. Prior to his Ph.D., Jay spent 6 years at Yahoo! working on mail spam detection and user trust, and he has also worked at Google, LinkedIn, and Oracle. Jay is the author of over thirty peer-reviewed publications and has received three best paper awards for his work. He is a recognized authority on knowledge graphs and has organized the Automatic Knowledge Base Construction (AKBC) and Statistical Relational AI (StaRAI) workshops, presented tutorials on knowledge graph construction at AAAI and WSDM, and had his work featured in AI Magazine. For more information, visit https://www.jaypujara.org.

Christopher Steven Marcum (UC-Irvine Dept. of Sociology, 2011) is a mathematical sociologist working as a staff scientist and methodologist in the Intramural Research Program of the National Human Genome Research Institute. His research interests include aging and the life course, social interaction, network dynamics, and health. At Genome, his work is primarily focused on the network dynamics of health communication and social behavior within families challenged with heritable disease. In addition, he has a lively program of research in network science methodology, theory, and analysis largely on the topics of relational event and exponential random graph modeling frameworks.

Laura M. Koehly (University of Illinois, Psychology, 1996) is chief and senior investigator in the Social and Behavioral Research Branch, National Human Genome Research Institute, National Institutes of Health. Dr. Koehly heads the Social Network Methods Section, where ongoing research activities focus on understanding how families communicate about and adapt to inherited disease risk. The hallmark of Dr. Koehly's research is the use of multi-informant designs to map the social systems within which family members are embedded, resulting in a large corpus of rich network data. This resource allows her group to advance new methods for social network data that can be used to model complex systems to facilitate the exploration of genomic, social, and environmental contributions to families' responses to disease risk and diagnoses. Ultimately, findings from this programmatic work will inform development of tailored approaches that leverage both genomic information and interpersonal processes to improve health outcomes.

Lise Getoor is a Professor in the Computer Science Department at UC Santa Cruz and founding Director of the UC Santa Cruz Data Science Research Center. Her research areas include machine learning and reasoning under uncertainty, with a focus on graph and network data. She has over 250 publications, including 11 best paper awards. She is a Fellow of the Association for Artificial Intelligence, an elected board member of the International Machine Learning Society, has served on the board of the Computing Research Association (CRA) and AAAI Council, and has served as Machine Learning Journal Action Editor, Associate Editor for the ACM Transactions of Knowledge Discovery from Data, and JAIR Associate Editor. She received her Ph.D. from Stanford University in 2001, her M.S. from UC Berkeley, and her B.S. from UC Santa Barbara, and was a Professor in the Computer Science Department at the University of Maryland, College Park, from 2001 to 2013.