Entity Resolution in Familial Networks

Pigi Kouki
UC Santa Cruz
pkouki@soe.ucsc.edu

Christopher Marcum
National Institutes of Health
chris.marcum@nih.gov

Laura Koehly
National Institutes of Health
koehlyl@mail.nih.gov

Lise Getoor
UC Santa Cruz
getoor@soe.ucsc.edu

ABSTRACT

Entity resolution is an important graph mining problem. Entity resolution is particularly interesting and challenging when there is rich relational structure. In this paper, we study the problem of performing entity resolution in familial networks. In our setting, we are given partial views of a familial network as described from the point of view of different people in the network and our goal is to reconstruct the underlying familial network from these perspective partial views. The data and relations provided may be inaccurate, missing or incomplete. In our approach, we start by augmenting the known set of familial relations with additional ones that are either inversed or derived from the original set of relations by linkage heuristics. Additionally, we propose a set of measures that capture the similarity of persons in the familial network based on both personal and relational information. We present a supervised learning approach where we view entity resolution in familial networks as a classification problem. Our experiments on real-world data from multiple-informant pedigrees show that our approach works well and that we can improve performance by considering separate similarity scores for each relation type.

1. INTRODUCTION

Many database applications, ranging from online shopping to health records, are faced with the challenge of determining whether two or more data instances refer to the same real-world entity. For example, in a hospital database "Jon Smith" and "Jonathan Smith" may refer to the same person and thus a doctor would like to have a unified view of their health record. The problem of identifying, matching, and merging references that correspond to the same entities is called **entity resolution**, which is also called de-duplication, reference reconciliation, record linkage, and data matching in the literature.

Entity resolution presents several challenges. In some cases, the data may contain polysemy: two references that appear to be identical, prima facie, are actually two separate entities. In other cases, the challenge is synonyms: two references that look different, but actually refer to the same underlying entity. Finally, the data itself may be inaccurate, missing, or incomplete, which sui generis poise complications to reconciling a set of entities.

Typical approaches to performing entity resolution use the set of attributes characterizing the references (e.g. name, occupation, age) to compute different notions of similarity (e.g. approximate string matching for names) among the references [11]. These similarities are then used in a variety of ways to determine whether two references refer to the same entity. In some cases, we may have additional information available for the references that may be useful while performing entity resolution. One such example is relational information that describes how references are related to each other. To this end, approaches that incorporate relational similarities [2, 4, 7, 10] have been shown to generally perform better than approaches that use only attribute-based similarities. Most of the previous works consider approaches that deal with accurate data and they typically consider one type of relation. However, in several real-life applications we need to operate with information that may be inaccurate or missing and also with several types of relations.

Our work is motivated by the problem of entity resolution in familial networks. Familial networks consist of the members of a family together with their relations. Such networks are prevalent in family health history applications (e.g., the medical history of the ancestors of a person), genealogy applications (e.g., *ancestry.com*), and also in areal administrative records (i.e., civic residential registrations). In such applications, several different people provide a portion of the familial network as seen from their point of view and our goal is to infer the whole familial network. For example, in a service such as *ancestry.com* users provide their immediate family tree and the application will attempt to determine the users' family trees as far back as possible. To do this, we need to determine which persons from the different trees are the same, i.e. we need to perform entity resolution within and between family trees.

In addition to the typical challenges of entity resolution described above (i.e. polysemy and synonymy) the nature of the data in such networks presents additional challenges. First, because the users provide their data from memory, the data may be inaccurate, missing or incomplete. For example, a given person providing data may remember his aunt being 45 years old while another person may recall her as being 47. Additionally, people may not provide accurate information for relatives that are further in the tree (e.g.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

may misspell the name of a cousin or may forget to include one altogether). Finally, not all relations are provided beforehand. For example, a son may provide his mother's and father's information and indicate the parent relation but not the spouse relation between his parents. Similarly, when the mother provides the data, she may describe a child relation and a spouse relation. The fact that there are several ways of describing the same relations within a familial network is an additional challenge.

In our work, we start by proposing an algorithm for inferring the missing relationships in order to create a richer representation that will help with our entity resolution task. Then, we view the entity resolution problem as a supervised classification problem and we explore the performance of Naïve Bayes and logistic regression. More specifically, for all potential pairs of references we want to determine whether they refer to the same physical entity. For our approach, we propose both features that capture attribute similarities, and also propose features that capture relational similarity. We investigate different notions of relational similarity, depending on type of relations. Finally, because of the possibility that our classification scheme may indicate that a given reference matches more than one other references, we also solve the so-called one-to-one matching problem; we implement a matching restrictions algorithm that ensures that one person will be matched to at most one entity. Our experimental evaluation shows that relational features do improve the performance of the entity resolution task in familial networks.

Our contributions are: 1) an algorithm that infers missing familial relationships based on existing familial relationships; 2) an exploration of different types of similarity measures including name similarities, numeric attribute similarities, and familial relationship similarities; 3) a greedy algorithm that satisfies the one-to-one matching restriction; and 4) an extensive evaluation study on a real dataset of familial networks coming from the medical domain.

The rest of our paper is organized as follows. We start by formally defining the problem in Section 2. In Section 3 we present our approach, which we experimentally evaluate in Section 4. We discuss related work in Section 5 and we conclude and present our future work in Section 6.

2. PROBLEM DEFINITION

In our entity resolution setting, we are given a set of ego-centric views of a family tree. The family tree is not directly provided to us. Each of the ego-centric views is from the perspective of a *participant* who has provided familial relationships between the participant and mentions of other family members as well as personal information (e.g. name, age, gender) for the participant and the rest of the mentions. Relationship types are divided into two categories: first and second degree. First degree relations include: mother, father, spouse, brother, sister, son, daughter, while second degree relations include: grandmother, grandfather, aunt, uncle, grandson, granddaughter, niece, and nephew. Our task is to align all the ego-centric views in order to construct the family tree, which maps the participant, and each of the mentions, to a member of the family tree. This allows us to reconstruct the family tree from the collection of ego-centric views. We refer to this task as *entity resolution in familial networks*. We formally define the problem as follows:

Problem Definition. We assume there is an underlying family $\mathbf{F}$ which contains an unobserved set of actors (persons), i.e. $\mathbf{F} = \{A_1, A_2, \ldots, A_m\}$. We are given a set of k ego-centric views of the family tree $\mathcal{T} = \{\mathbf{T}^1(\cdot), \mathbf{T}^2(\cdot), \ldots, \mathbf{T}^k(\cdot)\}$ which we call *participant ego-centric trees*. Each such ego-centric tree is defined as $\mathbf{T}^i(p^i) = \{r_{t_x}(p^i, m_1^i), \ldots, r_{t_y}(p^i, m_{li}^i)\}$, where $\mathcal{M}^i = \{p^i, m_1^i, \ldots, m_{li}^i\}$ is the set of mentions in the tree that correspond to the actors, p^i is the mention that provided its ego-centric tree and is called a *participant* , $t_j \in \tau$ denotes the type of relation (e.g. son, daughter, father, aunt) and m_j^i denotes the mention with whom the participant p^i shares the relation type t_j. A participant p^i can have an arbitrary number of relations of the same type (e.g. two daughters, three brothers, zero sisters). The participant p^i also specifies a set of attributes for himself and all his related mentions. The attributes are *first name*, *maiden name*, *last name*, *title*, *gender*, *age*, and *living status*. For the related mentions some attribute values may be missing or incorrect. Our goal is to examine all the mentions (participants and non-participants) across all the ego-centric trees and match them 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. □

3. OUR APPROACH

We view this problem as a *supervised classification* problem and use machine learning to solve it. More specifically, we consider all pairs of mentions where each part of the pair is coming from a different ego-centric tree in the family. If the pair of mentions represents the same entity we consider the pair belonging to the class MATCH, otherwise to the class NO MATCH. As we will show later, we will train the classifier on a collection of matched family trees, and then use the classier to make predictions for a new, unresolved family.

In such an approach, we need to make a few important decisions. First, we can take advantage of the relation information in order to perform effective entity resolution. As we discussed, the relationship information is only available for the *participants* and not for all the mentions in the family. So, we can use the observed relationships for the participants in order to infer the unobserved relationships for the rest of the mentions. Having a richer representation will help us match the mentions more effectively. Second, we need to design the features that we will use when learning our model. These features are meant to capture various aspects of similarity between the mention pairs. In traditional entity resolution tasks, features based on personal information are typically used. In our work we additionally consider features based on familial relationships. After defining the features, we need to determine which classification method performs well for our task.

Finally, because we are classifying pairs of mentions, we may generate solutions that are inconsistent. For example, the classifier may output that mention $m_1^1 \in \mathbf{T}_1$ is the same with both mentions $m_1^2 \in \mathbf{T}_2$ and $m_2^2 \in \mathbf{T}_2$. In this case, we need to fix the output so that it is consistent with the requirement that each mention from a tree can be matched to at most one other mention in any other tree.

In what follows, we present our approach to populating relationships, generating features for comparing two mentions, and discussing the matching restriction problem.

3.1 Relationship Population

Figure 1: (a)For a family $\mathbf{F}$, participant ego-centric tree $\mathbf{T}^1$(Anabel Perez) and (b) the derived mention ego-centric trees $\mathbf{T}^1_{dev}$(Mateo Garcia) and $\mathbf{T}^1_{dev}$(Lucia Martinez). Bold black borders indicate the center of the tree (participant or mention).

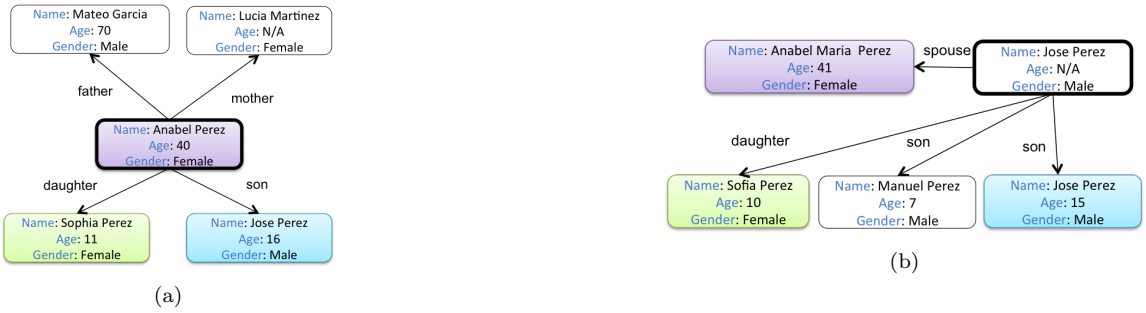

Figure 2: Two participant ego-centric trees for family $\mathbf{F}$: $\mathbf{T}^1$(*Anabel Perez*) and $\mathbf{T}^2$(*JosePerez*). Mentions in same colors represent same actors. White means that the mentions were not matched across the trees.

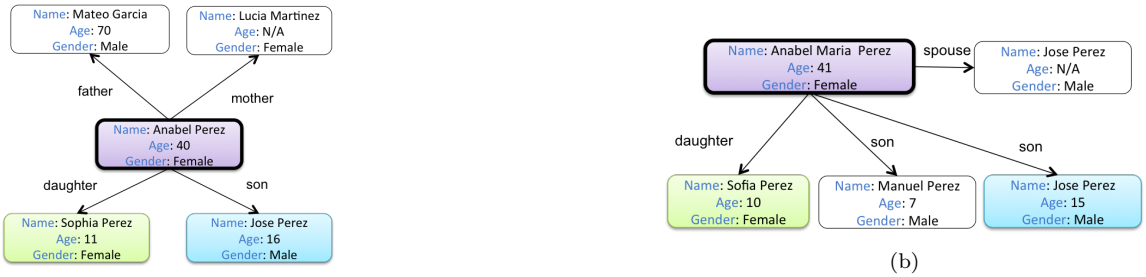

Figure 3: (a) Participant ego-centric tree $\mathbf{T}^1$(*Anabel Perez*) and (b) Mention ego-centric tree $\mathbf{T}^2_{dev}$(Anabel Maria Perez) for family $\mathbf{F}$.

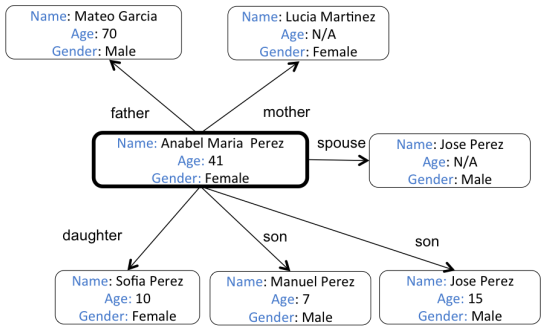

Figure 4: Aggregated family tree for family $\mathbf{F}$.

To infer the missing relationships, for each mention $m_x^i \in \bigcup_{j=1}^{k} \mathcal{M}^j$ we generate its ego-centric tree $\mathbf{T}_{dev}^i(m_x^i)$ based on the information given in $\mathbf{T}^i(p^i)$. We call the derived trees *mention ego-centric*. More specifically, to construct the relation set $\mathbf{T}_{dev}^i(m_x^i)$ for mention m_x^i we perform two kinds of operations:

1. **Relationship Inversion**. Given the relationship $r_{t_1}(p^i, m_x^i)$ in $\mathbf{T}^i(p^i)$ we introduce the inverted relationship $r_{t_1'}(m_x^i, p^i)$ in $\mathbf{T}_{dev}^i(m_x^i)$. For example, if we are given the relationship $r_{father}(p^i, m_x^i)$ in $\mathbf{T}^i(p^i)$, then we create the inverse relation $r_{son}(m_p^i, p^i)$ in $\mathbf{T}_{dev}^i(m_x^i)$ (or *daughter* depending on the gender of p^i).

2. **Relationship Derivation**. We add relationships in $\mathbf{T}_{dev}^i(m_x^i)$ that we derive based on the existence of other relationships in $\mathbf{T}^i(p^i)$ that do not involve m_x^i. For example, given the relations $\{r_{father}(p^i, m_x^i), r_{mother}(p^i, m_y^i)\}$ in $\mathbf{T}^i(p^i)$, then we derive the relations $r_{spouse}(m_x^i, m_y^i)$ in $\mathbf{T}_{dev}^i(m_x^i)$ and $r_{spouse}(m_y^i, m_x^i)$ in $\mathbf{T}_{dev}^i(m_y^i)$.

As an example, let us assume that for a family $\mathbf{F}$ a participant describes her ego-centric tree, i.e. in Figure 1(a) participant *Anabel Perez* describes In this case:

$$\mathbf{T}^1(\textit{Anabel Perez}) = \{r_{father}(\textit{Anabel Perez, Mateo Garcia}), \\ r_{mother}(\textit{Anabel Perez, Lucia Martinez})\}$$

In Figure 1(b) we follow the above process and derive the mention ego-centric trees

$$\mathbf{T}_{dev}^1(\textit{Mateo Garcia}) = \{r_{daughter}(\textit{Mateo Garcia, Anabel Perez}), \\ r_{spouse}(\textit{Mateo Garcia, Lucia Martinez})\}$$

and

$$\mathbf{T}_{dev}^1(\textit{Lucia Martinez}) = \{r_{daughter}(\textit{Anabel Perez}), \\ r_{spouse}(\textit{Mateo Garcia})\}$$

for mentions *Mateo Garcia* and *Lucia Martinez* respectively.

Once we create all the ego-centric trees, we compare all pairs of ego-centric trees that do not contain the same participant. More specifically, for each mention m_x^i, we compare its ego-centric tree $\mathbf{T}^i(m_x^i)$ with all other ego-centric trees $\mathbf{T}^v(m_y^v)$ where $m_y^v \in \bigcup_{j=1, j\neq i}^{k} \mathcal{M}^j$, in order to determine whether the two mentions represent the same actor. For each pair of mentions we decide if it represents the same entity or not. For example, in Figure 2, given two ego-centric trees $\mathbf{T}^1(\textit{Anabel Perez})$ and $\mathbf{T}^2(\textit{Jose Perez})$ we pairwise compare all the mentions from $\mathbf{T}^1(\textit{Anabel Perez})$ with all the mentions from $\mathbf{T}^2(\textit{Jose Perez})$. The goal is to identify that mentions in colored (i.e. non-white) boxes in Figure 2 are the same actor and construct the aggregated family tree as shown in Figure 4.

3.2 Feature Generation and Classification

In order to train a classification model we need to generate features that the classifier will use to decide between the MATCH and NO MATCH class for a given pair of mentions. To this end, we generate several features that capture different notions of similarity between the mentions. Our intuition is that two mentions that refer to the same entity will have high similarity values in these features.

Given a set of participant ego-centric trees similar to those of Figure 2, we employ a set of similarity metrics in order to compare pairs of mentions, presented below. When comparing two mentions, we use their ego-centric trees (participant or mention). For example, if we want to compare *Anabel Perez* from tree (a) with *Anabel Maria Perez* from tree (b) then we use the trees that are ego-centric towards *Anabel Perez* and *Anabel Maria Perez* as depicted in Figure 3. In the following, when using the term "tree" we refer to a participant or mention ego-centric tree.

3.2.1 Name Similarity

Obviously, the most important property of the mentions is their names. In order to match names, there are a variety of string metrics that can be used, all capturing different aspects of the kinds of mistakes that can be made. In our evaluation, we use two popular metrics, namely the Levenshtein [8] and Jaro-Winkler [6, 12]. The former is known to work well for common typographical errors, while the latter is specifically designed to work well with names. For example, in Figure 2 *Sophia Perez* and *Sofia Perez* from the two trees will have high name similarity under these two metrics, while *Sophia Perez* and *Jorge Perez* will have lower similarity. There are several ways we can use these metrics. In our work, given two mentions that each has a first, maiden, and last name we compute three different features: one feature for the first name similarity, one for the maiden name similarity, and one for the last name similarity. We compute the values of these features for both similarity metrics (we present all features used in Table 2). We also experimented with using one similarity score that uses a single combined string of the first, maiden, and last name, but we found that this approach did not work as well.

3.2.2 Personal Information Similarity

In addition to names, we also have available other personal information for the mentions that we can use towards our entity resolution task. For example, age is an important feature for entity resolution in family trees, since often times there will be the same names, but in different generations, and our goal is to distinguish between them. For personal information such as gender and living status, we use a simple binary match feature which is 1 if the values match and 0 otherwise. For personal information that is numeric such as age, we compute the ratio of the smallest value over the largest value. For example, for two mentions $m_1^1 \in \mathbf{T}^1$ and $m_1^2 \in \mathbf{T}^2$ for the feature age we compute:

$$s_{age}(m_1^1, m_1^2) = \frac{\min\{m_1^1.age, m_1^2.age\}}{\max\{m_1^1.age, m_1^2.age\}}$$

Again, there are several options to compare numeric attributes. We also experimented with the difference in ages and the ratio of the absolute difference of two ages over the maximum value of age in the dataset, however these did not work as well.

3.2.3 Relationship Similarity

Using name and personal information similarities is the first step towards entity resolution. However, oftentimes these two pieces of information are not enough. For example, often times people in the same tree can have the same

name (sometimes jr./sr. are used to disambiguate, but often these are not available). In our example, *Jose Perez* from tree (a) has the same name as two mentions from tree (b). Additionally, the same actors may have different last names across different participant ego-centric trees. This is often the case for married women where in one tree the married last name is provided while in another the family last name is provided. Finally, in some cases personal information might be missing or incorrect.

In order to deal with these challenges, we also use the provided relational information. There are many ways that we can combine and use such information. In our case we define a feature that takes into account 1) the number of matching relationships between two mentions (e.g. two mentions have both two sisters and three sons) and 2) how close the names of the persons within the same relationships are (e.g. two mentions have a mother that in both cases has name *Anabel Maria Perez*). We introduce several versions of this feature by varying the type of relations we are using (e.g. use of all relations, use only relations of first degree).

More specifically, we begin with the intuition that if two mentions in two different trees represent the same actor then they should have a similar set of relations. For example, in Figure 2 *Anabel Perez* from tree (a) is the same person as *Anabel Maria Perez* from tree (b). One way to measure the similarity of these two mentions in terms of relationships is by capturing how many relationships of the same type they share. However, just counting the number of matching relationships is not sufficient. This is because there are cases where two mentions from different trees have a high number of matching relations, but the persons in these same relationships are completely different. Since our goal is to match entities, we also consider information for the mentions that participate in the relationships. More specifically, given two mentions, we consider as a feature the sum of the name similarities of their matching relations. When there are multiple relationships of the same type, we consider the match with the highest name similarity. For example, when comparing *Jose Perez* from tree (a) to the sons of *Anabel Perez* from tree (b), we will consider that, for the son relation, he is a better match to *Jose Perez* from tree (b) than *Manuel Perez* because of the name similarity.

We formally define this similarity for two mentions $m_1^1 \in \mathbf{T}^1$ and $m_1^2 \in \mathbf{T}^2$ as follows:

$$s_{\tau_\rho,w}(m_1^1, m_1^2) = \frac{\sum\limits_{\substack{t \in \tau_\rho \\ rel_t(m_1^1)\neq\emptyset \\ rel_t(m_1^2)\neq\emptyset}} \max\limits_{\substack{\forall m_i^1 \in rel_t(m_1^1) \\ \forall m_j^2 \in rel_t(m_1^2)}} sim_w(m_i^1.name, m_j^2.name)}{N} \quad (1)$$

where:

- the denominator N is for normalization and counts the number of common relationships.
- $rel_t(m_1^1)$ is the set of m_1^1's relations of type t. For example, in tree (b) of Figure 2 for the participant p^2 = *Anabel Maria Perez* we have that $rel_{son}(p^2) = \{$*Jose Perez, Manuel Perez*$\}$. We assume that $|rel_t(m_1^1)| \leq |rel_t(m_1^2)|$, where $|rel_t(\cdot)|$ is the cardinality of the set, e.g. $|rel_{son}(p^2)| = 2$.
- τ_ρ indicates which relationships we consider when computing the $s_{\tau_\rho,w}$ ($\tau_\rho \subseteq \tau$). For each different value of τ_ρ we compute a feature s_{τ_ρ}. In our work we experimented with different sets of relationships during the computation of Equation 1. More specifically, we considered the relation sets shown in Table 1.

  For each value of the τ_ρ we give to the feature $s_{\tau_\rho,w}$ a different name, e.g. when $\tau_\rho = \{mother, father\}$ we refer to the feature as $s_{parents}$. We present our experimental results when using different values of τ_ρ in the evaluation section.

τ_ρ **value**	**Description**
$\tau_\rho = \tau$	Use all relations
$\tau_\rho = \{t \in \tau : t$ is 1^{st} degree$\}$	Use only first degree relations.
$\tau_\rho = \{t \in \tau : t$ is 2^{nd} degree$\}$	Use only second degree relations.
$\tau_\rho \subset \tau$	Consider a proper subset of τ. We consider the following: parents, grandparents, siblings, grandchildren, uncles and aunts, nieces and nephews.
$\tau_\rho = \{t \in \tau : \lvert t\rvert = 1\}$	Use a single relation from the set. Single relations are: spouse, mother, father, grandmother, grandfather, son, daughter, grandson, granddaughter, sister, brother, uncle, aunt, nephew, niece.

Table 1: Sets of relations considered when using Equation 1.

- $sim_w(m_i^1.name, m_j^2.name)$ is the name similarity between m_i^1 and m_j^2. Again, we use the two variants for name similarity, i.e. both the Levenshtein (s_l) and Jaro-Winkler (s_{jw}) similarities and we can consider different ways of using or combining them. In our approach, for each of the similarities, given the attributes first name, maiden name, and last name, we define the sim_l or sim_{jw} as their summation. For example, for sim_l: $sim_l(m_i^1.name, m_j^2.name) = sim_l(m_i^1.firstName, m_j^2.firstName) + sim_l(m_i^1.maidenName, m_j^2.maidenName) + sim_l(m_i^1.lastName, m_j^2.lastName)$

  Finally, we compute both s_{τ_ρ,s_l} and $s_{\tau_\rho,s_{jw}}$ and we take the maximum of the two. The final relationship similarity is computed as:

  $$s_{\tau_\rho}(m_1^1, m_1^2) = max\{s_{\tau_\rho,s_l}(m_1^1, m_1^2), s_{\tau_\rho,s_{jw}}(m_1^1, m_1^2)\}$$

  We present the full set of features in Table 2.

3.2.4 Classification

For each pair of mentions that we compare, we generate a vector of feature values based on the similarities that we described above. We normalize the values of the features to be in the range between 0 and 1.

Once we have computed the feature vectors of all pairs of mentions in our training dataset, we can use the data to learn a classification model. Once we train our classifier on the training data, we can then use this classifier to predict arbitrary and unseen pairs of mentions. In our work we experimented with different classifiers. We report the results on two of them in our experimental section.

3.3 Matching Restrictions

Type	Feature	Description
Name Similarity	l_f, l_m, l_l	Levenshtein similarity for first, maiden, and last name
	w_f, w_m, w_l	Jaro-Winkler similarity for first, maiden, and last name
Personal Similarity	s_{age}	Ratio of the ages of the two mentions
	b_{title}	Whether the two mentions agree on their titles (e.g. the both have Sr. as a title)
	b_{alive}	Whether the two mentions are both alive
	b_{gender}	Whether the two mentions agree on their gender
Relationship Similarity	s_{τ_ρ}	The maximum of the sum of Levenshtein and Jaro-Winkler distances of the mentions' relationship names.

Table 2: Features used by our classification model.

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

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

Algorithm 1: Enforcing the one-to-one matching restrictions.

In order to satisfy the one-to-one matching restriction, we implement the approach presented in Algorithm 1. The specific algorithm is a greedy approach that enforces one-to-one matching restrictions. The main idea is to sort the pairs that are considered to belong in the MATCH class by their likelihood of belonging to that class according to our classifier. Then, we keep the pairs that have the maximum likelihood of belonging to the MATCH class and we discard all other pairs. We use the specific algorithm because it is simple to implement, efficient, and has good performance in practice.

4. EVALUATION

4.1 Dataset and Evaluation Metrics

For our experimental evaluation we use a dataset provided by the National Institues of Health (NIH). Our dataset consists of 162 families. For each family we are given 3 or 4 family trees. In total, we have to compare around 300 thousand potential pairs of mentions and decide whether they belong to the same actor. The dataset has been annotated by hand. The annotation was performed by at least two coders, with reconciliation of differences.

Out of all the pairs that we have to compare, 1.6% represent the same actors and thus belong to the class MATCH and 98.4% belong to the class NO MATCH. Thus, we are dealing with a classification problem with imbalanced classes. In the following, we report the results after performing 5-fold cross validation. For each fold, we partition our dataset in a training part that will be used to learn our model, and a testing part that will be used to validate our model. The training set consists of mentions that belong to 130 families for which the true class (MATCH or NO MATCH) is known and the test set consists of mentions that belong to the remaining 32 families for which the true class is unknown. We repeat this process 5 different times to get an estimation of our performance with lower variance and to ensure the generality of our approach to new datasets.

For our evaluation metrics, we use the *precision*, *recall*, and *f-measure* per class. At a high level, *precision* is the fraction of correct classifications that we performed. For example, for the class MATCH, if our classifier output 100 pairs belonging to that class but only 90 of them were actually a MATCH according to our labeled data, then our precision is 0.9. On the other hand, *recall* is the fraction of correct classifications that we were able to retrieve from the data. For example, for the class MATCH, if there are 100 pairs belonging to that class in our labeled data but we only identified 10 of these pairs then our recall is 0.1. Apparently, we want our classifier to have both high precision (i.e. not introduce mistakes in the predictions) and high recall (i.e. identify as many MATCH pairs as possible). Finally, the f-measure combines precision and recall in one metric.

In the following, we present the evaluation results of our approach. First, we show the performance of simple name similarity features and then we add personal information and relationship similarities. After that, we explore different ways of using relationship similarity.

4.2 Evaluation of Classification

In our approach we experimented with several classifiers, but due to space constraints we report results only on Naïve Bayes and logistic regression that performed well. The output of these classifiers is also helpful with enforcing the one-to-one restrictions because they output the probability that a given pair belongs to a specific class. As we discussed in Section 3.3, such information is essential for dealing with the matching restrictions problem. For all the classifiers we use Weka's implementation.[1] We ran all classifiers with Weka's default settings.

Table 3 presents the results of our method for the class MATCH only. We denote with bold the model that outperforms all the others for the same setting. The results for the class NO MATCH are very similar without much variation when changing the feature set or the classifier used, so we omit them due to space constraints. In general, for the class NO MATCH, precision varies from 98.5% to 99.9%, recall varies from 98.4% to 99.9%, and f-measure varies from 99.1% to 99.7%. The performance for the class NO MATCH are better than the results for the class MATCH. This is expected since the NO MATCH class is the majority class so the prediction task is much easier for this class.

To evaluate our approach, we started by using only name similarity and then we added personal information similarity. Finally, as we discussed in Section 3.2.3 in addition to the name and personal information similarity, we also explore relational features. More specifically we explored how weighting the different relationship types can influence our predictions. To this end, we ran the following experiments:

[1] http://www.cs.waikato.ac.nz/ml/weka/

Experiment 1: We use only name similarities as described in section 3.2.1 and call this experiment as **N**.
Experiment 2: We use name and personal information similarities as explained in Sections 3.2.1 and 3.2.2. We call this experiment as **NP**.
Experiment 3: We use one feature for relational similarity where all relationships have the same weight, i.e. $\tau_\rho = \tau$ and we compute one feature s_{all}. We call this experiment **NPR**.
Experiment 4: We use one feature for relational similarity where we consider only first degree relations all having same weight, i.e. $\tau_\rho = \{t \in \tau : t \text{ is } 1^{st} \text{ degree}\}$ and we compute one feature $s_{firstDegree}$. We call this experiment **NPR 1^{st} degree**.
Experiment 5: We use one feature for the relations of first degree and one feature for the relations of second degree, i.e. $\tau_\rho = \{t \in \tau : t \text{ is } 2^{nd} \text{ degree}\}$ and we compute two features $s_{firstDegree}$ and $s_{secondDegree}$. We call this experiment **NPR 1^{st} and 2^{nd} degree**.
Experiment 6: We use eight features, one for each of the grouped relations: parents, grandparents, children, grandchildren, uncles and aunts, nephews and nieces, siblings, spouse, i.e. $\tau_\rho \subset \tau$ and we compute $s_{parents}$, $s_{grandparents}$, $s_{children}$, $s_{grandchildren}$, $s_{auntsUncles}$, $s_{nephewsNieces}$, $s_{siblings}$, s_{spouse}. We call this experiment **NPR group**.
Experiment 7: We use 15 features, one for each of the relations: father, mother, grandfathers, grandmothers, sons, daughters, grandsons, granddaughters, uncles, aunts, nephews, nieces, brothers, sisters, spouse, i.e. $\tau_\rho = \{t \in \tau : |t| = 1\}$ and we compute s_{spouse}, s_{mother}, s_{father}, $s_{grandmother}$, $s_{grandfather}$, $s_{daughter}$, s_{son}, $s_{granddaughter}$, $s_{grandson}$, s_{sister}, $s_{brother}$, s_{aunt}, s_{uncle}, s_{nephew}, s_{niece}. We call this experiment **NPR individual**.
Experiment 8: We use all of the 36 features in experiments 3 to 6 above. We call this experiment **NPR all**.

We present the results in Table 3. Our first observation is that the models using personal information together with the names outperform the models that use only name similarities. Second, the models that use relationship information outperform the models that don't. This confirms our intuition that using both personal and relationship information together with name similarities would improve the performance in our entity resolution task. In general, the improvement seems to increase as we introduce more specific relational features. This is true for logistic regression where the best model is the one using all types of features (i.e. name, personal and all kinds of relationship similarity). For Naïve Bayes, the performance is best for the cases of using the grouped and individual relation features in terms of f-measure. Including all the relational features does not seem to help this classifier. This can be easily explained: Naïve Bayes makes the assumption that all the features are independent with each other given the class. However, the experiment **NPR all** includes features that are correlated with each other. After performing correlation analysis we determined that the individual features from **NPR individual** are correlated with the grouped ones from **NPR group**, e.g. s_{mother} is correlated with $s_{parents}$, $s_{brother}$ is correlated with $s_{sibling}$. On the other hand, logistic regression can better handle correlated features and it outperforms Naïve Bayes.

Finally, we note that for both the Naïve Bayes and the logistic regression classifiers the performance in terms of f-measure of the best models using relational information (**NPR group** and **NPR all** accordingly) is statistically significantly better than the models that do not take into account relational information at $\alpha = 0.05$ when using paired t-test.

4.3 Matching Restrictions

We finally turn to study the performance of our approach when applying the 1-1 matching restrictions that we discussed in Section 3.3. To this end, we picked the best results from each of the classifiers and applied our greedy matching restriction algorithm.

More specifically, we applied our algorithm to the **NPR group** model from Naïve Bayes and the **NPR all** model from logistic regression. We call these **NPR group + Matching Restrictions** and **NPR all + Matching Restrictions** respectively in Table 3.

Overall, we observe that compared to their non-restricted models the restricted ones overall increased precision and reduced recall. This is expected since, by removing pairs, we essentially become stricter in providing a prediction. In this way, we provide fewer predictions for the MATCH class but our predictions are more accurate. In both cases, the overall f-measure increases when we apply our matching restriction algorithm.

5. RELATED WORK

There is a large body of related work in the general area of entity resolution [3]. In our work we target the entity resolution problem when relational data are available. Bhattacharya and Getoor [2] propose a collective classification method based on a greedy clustering technique over the relationship graph. Similarity values are computed as the weighted sum between the attribute value similarity and relational similarity, but for a single relation (co-authorship). In our work, we follow a supervised approch (vs. the unsupervised of Bhattacharya and Getoor [2]) and we consider several relationship types (e.g. father, mother, siblings).

Dong et al. [4] also propose a collective classification approach. The main idea is the use of contextual information (e.g. email lists) together with similarity metrics across attributes (e.g. similarity between email address and name) in order to enrich the references. In our approach, we also perform reference enrichment and, in addition, we also enrich the relations by performing inversion and derivation.

Kalashnikov and Mehrotra [7] propose an approach for the reference disambiguation problem, i.e. for the case when the set of entities is known and the task is to match incoming references to one of the entities. The main idea of their approach is to build a semantic weighted relationship graph among different types of entities and different relations and classify the entities as matching or non-matching in an iterative fashion by using the graph. In our case, the entities are not known beforehand.

Singla and Domingos [10] also study the entity resolution problem and propose a generalization of the Fellegi-Sunter model [5] which combines first-order logic and Markov random fields to perform collective classification. In the proposed Markov logic networks the predicates take boolean values (i.e. true or false), while in our case we allow for more fine-grained notions of similarity. The aforementioned works propose collective solutions to the entity resolution task, i.e. resolutions are not made independently, but in-

Classifier	Experiment	Precision (SD)	Recall (SD)	F-Measure (SD)
Naïve Bayes	**N**	0.472 (0.045)	0.919 (0.013)	0.622 (0.040)
	NP	0.505 (0.043)	0.926 (0.014)	0.652 (0.037)
	NPR	0.506 (0.042)	0.926 (0.014)	0.653 (0.036)
	NPR 1^{st} degree	0.514 (0.042)	0.926 (0.013)	0.660 (0.035)
	NPR 1^{st} and 2^{nd} degree	0.518 (0.041)	**0.927** (0.013)	0.664 (0.034)
	NPR group	0.533 (0.041)	0.925 (0.012)	0.676 (0.033)
	NPR individual	0.533 (0.035)	0.918 (0.010)	0.674 (0.027)
	NPR all	0.521 (0.037)	0.902 (0.009)	0.660 (0.030)
	NPR group + Matching Restrictions	**0.551** (0.044)	0.924 (0.013)	**0.689** (0.035)
Logistic Regression	**N**	0.748 (0.031)	0.777 (0.042)	0.762 (0.028)
	NP	0.819 (0.027)	0.809 (0.038)	0.813 (0.026)
	NPR	0.809 (0.026)	0.810 (0.037)	0.809 (0.024)
	NPR 1^{st} degree	0.854 (0.018)	0.794 (0.042)	0.822 (0.024)
	NPR 1^{st} and 2^{nd} degree	0.852 (0.016)	0.794 (0.037)	0.822 (0.022)
	NPR group	0.865 (0.014)	0.788 (0.030)	0.824 (0.018)
	NPR individual	0.865 (0.012)	0.788 (0.029)	0.824 (0.017)
	NPR all	0.878 (0.015)	**0.800** (0.028)	0.836 (0.018)
	NPR all + Matching Restrictions	**0.909** (0.014)	0.789 (0.027)	**0.844** (0.016)

Table 3: Performance of two classifiers for the class MATCH with varying types of features. Numbers in parenthesis indicate standard deviations. Bold shows the best performance in each metric for each classifier

stead one resolution decision affects other resolutions [2]. Collective methods have been shown to perform well in certain cases compared to non-collective methods, so we plan to extend our work to include a collective approach in the future.

6. CONCLUSIONS AND FUTURE WORK

In this paper we studied the problem of the entity resolution in familial networks. Our approach starts by augmenting the given set of familial relations with additional ones that are either inversed or derived from the original set of relations. We propose a set of similarity measures that capture the similarity of persons in the family based on both personal and relational information. We presented a supervised learning approach where we view the entity resolution in familial networks as a classification problem. Our experiments on real-world data show that our approach works well and that we can improve performance by considering separate similarity scores for each relation type.

For our future work, we plan to explore the use of structured output learning techniques [9]. These techniques can directly consider the matching constraints during the learning of the classifier instead of post processing the classification results. We plan to focus on a statistical relational learning approach and more specifically Probabilistic Soft Logic (PSL) [1]. PSL is an open source machine learning framework[2] that provides a logic-based declarative language that allows for collective probabilistic inference. Another direction is to consider temporal relations e.g. ex-wife, ex-husband.

Acknowledgements

We would like to thank Jay Pujara for insightful discussions. This work was supported by the National Science Foundation grant IIS1218488 and by the National Human Genome Research Institute Division of Intramural Research at the National Institutes of Health (Z01HG200335 to LMK). 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 or the National Institutes of Health.

[2]http://psl.umiacs.umd.edu

7. REFERENCES

[1] S.H. Bach, M. Broecheler, B. Huang, and L. Getoor. Hinge-loss markov random fields and probabilistic soft logic. *ArXiv:1505.04406 [cs.LG]*, 2015.

[2] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. In *TKDD*, 2007.

[3] P. Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Publishing Company, Incorporated, 2012.

[4] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[5] P. Fellegi and B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 1969.

[6] M. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa. *Journal of the American Statistical Association*, 1989.

[7] D. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *TODS*, 2006.

[8] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computer Survey*, 2001.

[9] S. Nowozin, P. Gehler, J. Jancsary, and C. Lampert. *Advanced Structured Prediction*. The MIT Press, 2014.

[10] P. Singla and P. Domingos. Entity resolution with markov logic. In *ICMD*, 2006.

[11] Winkler W. The state of record linkage and current research problems. In *Tech. rep., Statistical Research Division, U.S. Census Bureau*, 1999.

[12] W. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Section on Survey Research*, 1990.