Entity Resolution with Markov Logic

Parag Singla Pedro Domingos
Department of Computer Science and Engineering
University of Washington Seattle, WA 98195-2350, U.S.A.
parag,pedrod@cs.washington.edu

Abstract

Entity resolution is the problem of determining which records in a database refer to the same entities, and is a crucial and expensive step in the data mining process. Interest in it has grown rapidly in recent years, and many approaches have been proposed. However, they tend to address only isolated aspects of the problem, and are often ad hoc. *This paper proposes a well-founded, integrated solution to the entity resolution problem based on Markov logic. Markov logic combines first-order logic and probabilistic graphical models by attaching weights to first-order formulas, and viewing them as templates for features of Markov networks. We show how a number of previous approaches can be formulated and seamlessly combined in Markov logic, and how the resulting learning and inference problems can be solved efficiently. Experiments on two citation databases show the utility of this approach, and evaluate the contribution of the different components.*

1 Introduction

Data cleaning and preparation is the first stage in the data mining process, and in most cases it is by far the most expensive. Data from relevant sources must be collected, integrated, scrubbed and pre-processed in a variety of ways before accurate models can be mined from it. When data from multiple databases is merged into a single database, many duplicate records often result. These are records that, while not syntactically identical, represent the same real-world entity. Correctly merging these records and the information they represent is an essential step in producing data of sufficient quality for mining. This problem is known by the name of entity resolution, record linkage, object identification, de-duplication, merge/purge, data association, identity uncertainty, reference reconciliation, and others. In recent years it has received growing attention in the data mining community, with a related workshop at KDD-2003 [27] and a related task as part of the 2003 KDD Cup [16].

The entity resolution problem was first identified by Newcombe et al. [31], and given a statistical formulation by Fellegi and Sunter [14]. Most current approaches are variants of the Fellegi-Sunter model, in which entity resolution is viewed as a classification problem: given a vector of similarity scores between the attributes of two entities, classify it as "Match" or "Non-match." A separate match decision is made for each candidate pair, followed by transitive closure to eliminate inconsistencies. Typically, a logistic regression model is used [1]. One line of research has focused on scaling entity resolution to large databases by avoiding the quadratic number of comparisons between all pairs of entities (e.g., [20, 30, 26, 7]). Another has focused on the use of active learning techniques to minimize the need for labeled data (e.g., [44, 38, 4]). Several authors have devised, compared and learned similarity measures for use in entity resolution (e.g., [6, 45, 3]). A number of alternate formulations have also been proposed (e.g., [5]). Entity resolution has been applied in a wide variety of domains (e.g., [33, 10]) and to different types of data, including text (e.g., [25]) and images (e.g., [21]). Winkler [46] surveys research in traditional record linkage.

Most recently, several authors have pointed out that match decisions should not be made independently for each candidate pair. While the Fellegi-Sunter model treats all pairs of candidate matches as i.i.d. (independent and identically distributed), this is clearly not the case, since each entity appears in multiple candidate matches. While this interdependency complicates learning and inference, it also offers the opportunity to improve entity resolution, by taking into account information that was previously ignored. For example, Singla and Domingos [42], Dong et al. [12] and Culotta and McCallum [9] allow the resolution of entities of one type to be helped by resolution of entities of related types (e.g., if two papers are the same, their authors are the same, which in turn is evidence that other pairs of papers by the same authors should be matched, etc.). McCallum and Wellner [28] incorporate the transitive closure step into the statistical model. Pasula et al. [34] incorporate parsing of entities from citation lists into a citation matching model. Bhattacharya and Getoor [2] use coauthorship relations to help match authors in citation databases. Milch et al [29] propose a language for reasoning about entity resolution. Shen et al. [40] exploit various types of constraints to improve matching accuracy. Davis et al. [10] use inductive logic programming techniques to discover relational rules for entity resolution, which they then combine using a naive Bayes classifier.

In this paper we propose a simple and mathematically sound formulation of the entity resolution problem that incorporates these non-i.i.d. approaches, and can be viewed as a generalization of the Fellegi-Sunter model. It takes advantage of the recent progress in statistical relational learning (a.k.a. multi-relational data mining), which provides rich representations and efficient inference and learning algorithms for non-i.i.d. data [15, 13]. In particular, we use Markov logic, which combines first-order logic and Markov random fields [36], with weighted satisfiability testing for efficient inference and a voted perceptron algorithm for discriminative learning [41]. Our formulation makes it practical to combine many different components into a comprehensive solution to the entity resolution problem. We illustrate this in this paper by combining a few salient ones, and applying the resulting system to a large citation database.

We begin by briefly reviewing the necessary background on Markov networks, first-order logic and Markov logic. We then describe our proposed approach to entity resolution, report on our experiments, and outline directions for future work.

2 Markov Networks

A *Markov network* (also known as *Markov random field*) is a model for the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ [35]. It is composed of an undirected graph G and a set of potential functions ϕ_k. The graph has a node for each variable, and the model has a potential function for each clique in the graph. A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by

$$P(X=x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \quad (1)$$

where $x_{\{k\}}$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z, known as the *partition function*, is given by $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$. Markov networks are often conveniently represented as *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state, leading to

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right) \quad (2)$$

A feature may be any real-valued function of the state. This paper will focus on binary features, $f_j(x) \in \{0,1\}$. In the most direct translation from the potential-function form (Equation 1), there is one feature corresponding to each possible state $x_{\{k\}}$ of each clique, with its weight being $\log \phi_k(x_{\{k\}})$. This representation is exponential in the size of the cliques. However, we are free to specify a much smaller number of features (e.g., logical functions of the state of the clique), allowing for a more compact representation than the potential-function form, particularly when large cliques are present. Markov logic takes advantage of this.

Maximum *a posteriori* (MAP) inference in Markov networks involves finding the most likely state of a set of query (output) variables given the state of a set of evidence (input) variables, and is NP-hard [37]. Conditional inference involves computing the distribution of the query variables given the evidence, and is #P-complete [37]. The most widely used approximate solution to this problem is Markov chain Monte Carlo (MCMC) [18], and in particular Gibbs sampling, which proceeds by sampling each non-evidence variable in turn given its Markov blanket (i.e., its neighbors in the graph), and counting the fraction of samples that each variable is in each state.

Maximum-likelihood or MAP estimates of Markov network weights cannot be computed in closed form, but, because the log-likelihood is a concave function of the weights, they can be found efficiently using standard gradient-based or quasi-Newton optimization methods [32]. Another alternative is iterative scaling [11]. Features can also be learned from data, for example by greedily constructing conjunctions of atomic features [11].

3 First-Order Logic

A *first-order knowledge base (KB)* is a set of sentences or formulas in first-order logic [17]. Formulas are constructed using four types of symbols: constants, variables, functions, and predicates. Constant symbols represent objects in the domain of interest (e.g., people: `Anna`, `Bob`, `Chris`, etc.). Variable symbols range over the objects in the domain. Function symbols (e.g., `MotherOf`) represent mappings from tuples of objects to objects. Predicate symbols represent relations among objects in the domain (e.g., `Friends`) or attributes of objects (e.g., `Smokes`). A *term* is any expression representing an object in the domain. It can be a constant, a variable, or a function applied to a tuple of terms. For example, `Anna`, `x`, and

GreatestCommonDivisor(x, y) are terms. An *atomic formula* or *atom* is a predicate symbol applied to a tuple of terms (e.g., Friends(x, MotherOf(Anna))). A *ground term* is a term containing no variables. A *ground atom* or *ground predicate* is an atomic formula all of whose arguments are ground terms. Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers. A *positive literal* is an atomic formula; a *negative literal* is a negated atomic formula. A KB in *clausal form* is a conjunction of *clauses*, a clause being a disjunction of literals. Every KB can be converted to clausal form. A *possible world* or *Herbrand interpretation* assigns a truth value to each possible ground atom. In finite domains, first-order KBs can be *propositionalized* by replacing each universally (existentially) quantified formula with a conjunction (disjunction) of all its groundings.

A central (and NP-complete) problem in logic is that of determining if a KB (usually in clausal form) is *satisfiable*, i.e., if there is an assignment of truth values to ground atoms that makes the KB true. One approach to this problem is stochastic local search, exemplified by the WalkSAT solver [39]. Beginning with a random truth assignment, WalkSAT repeatedly flips the truth value of either (a) an atom that maximizes the number of satisfied clauses, or (b) a random atom in an unsatisfied clause. WalkSAT is highly efficient, and is able to solve hard instances of satisfiability with hundreds of thousands of variables in minutes. Many first-order problems (e.g., planning, software verification) can be solved efficiently by propositionalizing them and applying a satisfiability solver. The *weighted satisfiability* problem is a variant of satisfiability where each clause has an associated weight, and the goal is to maximize the sum of the weights of satisfied clauses. MaxWalkSAT is a direct extension of WalkSAT to this problem [22].

4 Markov Logic

A first-order KB can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in Markov logic is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal.

Definition 1 *[36] A Markov logic network (MLN) L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ (Equations 1 and 2) as follows:*

1. *$M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.*

2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.*

Thus there is an edge between two nodes of $M_{L,C}$ iff the corresponding ground predicates appear together in at least one grounding of one formula in L. An MLN can be viewed as a *template* for constructing Markov networks. From Definition 1 and Equations 1 and 2, the probability distribution over possible worlds x specified by the ground Markov network $M_{L,C}$ is given by

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right) \tag{3}$$

where F is the number formulas in the MLN and $n_i(x)$ is the number of true groundings of F_i in x. As formula weights increase, an MLN increasingly resembles a purely logical KB, becoming equivalent to one in the limit of all infinite weights.

In this paper we will focus on MLNs whose formulas are function-free clauses and assume domain closure, ensuring that the Markov networks generated are finite [36]. In this case, the groundings of a formula are formed simply by replacing its variables with constants in all possible ways. For example, if $C = \{\texttt{Anna}, \texttt{Bob}\}$, the formula ∀x Smokes(x) ⇒ Cancer(x) in the MLN L yields the features Smokes(Anna) ⇒ Cancer(Anna) and Smokes(Bob) ⇒ Cancer(Bob) in the ground Markov network $M_{L,C}$ (or ¬Smokes(Anna) ∨ Cancer(Anna) and ¬Smokes(Bob) ∨ Cancer(Bob) in clausal form). See Richardson and Domingos (2006, Table 2) for details.

MAP inference in Markov logic can be carried out efficiently using a weighted satisfiability solver like MaxWalkSAT [22]. This is because the exponent in Equation 3 is the sum of the weights of the satisfied ground clauses, and thus $P(X=x)$ can be maximized by maximizing this sum. To condition on evidence, we first replace the truth values of the evidence atoms into Equation 3 and simplify (false literals disappear, and true literals cause the clauses they appear in to disappear). Conditional probabilities can be computed by Gibbs sampling over the minimal ground network needed to answer the query; see Richardson and Domingos [36] for details.

Given a set of formulas, their weights can be learned either generatively (maximizing the joint likelihood of all predicates) or discriminatively (maximizing the conditional likelihood of the query predicates given the evidence ones).

In this paper we use discriminative learning, as proposed by Singla and Domingos [41]. The training data is a relational database (i.e., a set of positive ground literals with the closed world assumption, by which all atoms not in the database are assumed false).[1] Let x be the vector of truth values of the evidence atoms, and y the truth values of the query atoms. For simplicity, we assume that all non-evidence atoms are query atoms, which is appropriate for the application in this paper. We learn weights by gradient descent (or, more precisely, ascent) on the conditional log-likelihood of y given x. From Equation 3, the partial derivative of the conditional log-likelihood with respect to the weight of the ith clause is

$$\frac{\partial}{\partial w_i} \log P_w(y|x) = n_i(x,y) - E_w[n_i(x,y)] \tag{4}$$

where $n_i(x,y)$ is the number of true groundings of the ith clause in the data, and $E_w[n_i(x,y)]$ is the clause's expected number of true groundings, averaged over all possible worlds, weighted by their probabilities according to the current weights. Thus a clause's weight should increase when its actual count is greater than its predicted one, and decrease when it is lower. Although computing the expected counts $E_w[n_i(x,y)]$ is intractable, they can be approximated by the counts $n_i(x,y_w^*)$ in the MAP state $y_w^*(x)$ (i.e., the most likely state of y given x). This is a good approximation if most of the probability mass of $P_w(y|x)$ is concentrated around $y_w^*(x)$, and is the essence of the *voted perceptron* algorithm [8], which initializes all weights to zero, performs T steps of gradient descent, and returns the weights averaged over all iterations ($w_i = \sum_{t=1}^{T} w_{i,t}/T$). While it was originally developed for the special case of hidden Markov models, and used the Viterbi algorithm to find the MAP state, Singla and Domingos [41] generalized it to MLNs by replacing Viterbi with MaxWalkSAT.

It is also possible to learn the structure of MLNs using inductive logic programming techniques [23]. Learning can start from an empty network, or from an initial knowledge base.

Markov logic affords us the expressiveness of first-order logic while avoiding its brittleness, and makes it easy to incorporate partial, imperfect, and even contradictory knowledge into the data mining process. MaxWalkSAT inference and voted perceptron learning are efficient enough to be practical for domains of realistic size. These algorithms are publicly available in the Alchemy system, which we use in our experiments [24].

[1] If the closed world assumption is not made, the truth values of some atoms are unknown, and weights can be learned using a form of the EM algorithm.

5 Entity Resolution

5.1 Equality in Markov Logic

Most systems for inference in first-order logic make the *unique names assumption*: different constants refer to different objects in the domain. This assumption can be removed by introducing the *equality predicate* ($\texttt{Equals}(x,y)$ or $x = y$ for short) and its axioms [17]:

Reflexivity: $\forall x \;\; x = x.$

Symmetry: $\forall x,y \;\; x = y \Rightarrow y = x.$

Transitivity: $\forall x,y,z \;\; x = y \wedge y = z \Rightarrow x = z.$

Predicate equivalence: For each binary predicate R: $\forall x_1,x_2,y_1,y_2 \; x_1 = x_2 \wedge y_1 = y_2 \Rightarrow (R(x_1,y_1) \Leftrightarrow R(x_2,y_2))$. Similar axioms are required for predicates of other arities and for functions, but binary predicates suffice for this paper.

Adding the formulas above with infinite weight to an MLN allows it to handle non-unique names. We can also add the reverse of the last one with finite weight:

Reverse predicate equivalence: For each binary predicate R: $\forall x_1,x_2,y_1,y_2 \;\; R(x_1,y_1) \wedge R(x_2,y_2) \Rightarrow (x_1 = x_2 \Leftrightarrow y_1 = y_2)$, and similarly for other arities and functions.

The meaning of this formula is most easily understood by noting that it is equivalent to the two clauses:

$$\begin{aligned}
&\forall x_1,x_2,y_1,y_2 \; R(x_1,y_1) \wedge R(x_2,y_2) \wedge x_1 = x_2 \\
&\quad \Rightarrow y_1 = y_2 \\
&\forall x_1,x_2,y_1,y_2 \; R(x_1,y_1) \wedge R(x_2,y_2) \wedge y_1 = y_2 \\
&\quad \Rightarrow x_1 = x_2
\end{aligned}$$

As statements in first-order logic, these clauses are false, because different groundings of the same predicate do not generally represent the same tuples of objects. However, when added to an MLN with a finite weight, they capture an important statistical regularity: *if two objects are in the same relation to the same object, this is evidence that they may be the same object.* Some relations provide stronger evidence than others, and this is captured by assigning (or learning) different weights to the formula for different R. For example, when de-duplicating citations, two papers having the same title is stronger evidence that they are the same paper than them having the same author, which in turn is stronger evidence than them having the same venue. However, even the latter is quite useful: two papers appearing in the same venue are much more likely to be the same than two papers about which nothing is known.

Remarkably, *the equality axioms and reverse predicate equivalence are all that is needed to perform entity resolution in Markov logic*. As we will see, despite its simplicity, this formulation incorporates the essential features of some of the most sophisticated entity resolution approaches to date, including the "collective inference" approaches of McCallum and Wellner [28] and Singla and Domingos [42]. A number of other approaches can be incorporated by adding the corresponding formulas. (We emphasize that our goal is not to reproduce every detail of these approaches, but rather to capture their essential features in a simple, consistent form.) Further, entity resolution and data mining can be seamlessly combined, and aid each other, by performing structure learning on top of the entity resolution MLN.

5.2 Problem Formulation

We now describe our proposed approach in detail. For simplicity, we assume that the database to be deduplicated contains only binary relations. This entails no loss of generality, because an n-ary relation can always be re-expressed as n binary relations. For example, if a citation database contains ground atoms of the form $\texttt{Paper}(\texttt{title}, \texttt{author}, \texttt{venue})$, they can be replaced by atoms of the form $\texttt{HasTitle}(\texttt{paper}, \texttt{title})$, $\texttt{HasAuthor}(\texttt{paper}, \texttt{author})$ and $\texttt{HasVenue}(\texttt{paper}, \texttt{venue})$. Each real-world entity (e.g., each paper, author or venue) is represented by one or more strings appearing as arguments of ground atoms in the database. For example, different atoms could contain the strings `ICDM-2006`, `Sixth ICDM` and `IEEE ICDM'06`, all of which represent the same conference. We assume that the predicates in the database (representing relations in the real world) are *typed*; for example, the first argument of the predicate $\texttt{HasAuthor}(\texttt{paper}, \texttt{author})$ is of type `Paper`, and the second is of type `Author`. The goal of entity resolution is, for each pair of constants of the same type (x_1, x_2), to determine whether they represent the same entity: is $x_1 = x_2$? Thus the query predicate for inference is equality; the evidence predicates are the (binarized) relations in the database, and other relations that can be deterministically derived from the database (see below). The model we use for entity resolution is in the form of an MLN, with formulas and weights that may be hand-coded and/or learned. The most likely truth assignment to the query atoms given the evidence is computed using MaxWalkSAT. Conditional probabilities of query atoms given the evidence are calculated using Gibbs sampling.

The model includes a unit clause for each query predicate. (A unit clause is a clause containing only one literal.) The weight of a unit clause captures (roughly speaking) the marginal distribution of the corresponding predicate, leaving non-unit clauses to capture the dependencies between predicates. Since most groundings of most predicates are usually false (e.g., most pairs of author strings do not represent the same author), it is clearest to use unit clauses consisting of negative literals, with positive weights. Since predicate arguments are typed, there is a unit clause for equality between entities of each type (e.g., `paper1` = `paper2`, `author1` = `author2`, etc.). From a discriminative point of view, the weight of a unit clause represents the threshold above which evidence must accumulate for a candidate pair of the corresponding type to be declared a match.

5.3 Field Comparison

We assume that each field in a database to be deduplicated is a string composed of one or more words (or, more generally, tokens), and define the predicate $\texttt{HasWord}(\texttt{field}, \texttt{word})$ which is true iff `field` contains `word`. Applied to this predicate, reverse predicate equivalence states that

$$\forall x_1, x_2, y_1, y_2\ \texttt{HasWord}(x_1, y_1) \wedge \texttt{HasWord}(x_2, y_2) \\ \wedge\ y_1 = y_2 \Rightarrow x_1 = x_2$$

or, in other words, fields that have a word in common are more likely to be the same. When inserted into Equation 3, and assuming for the moment no other clauses, this clause produces a logistic regression for the field match predicate $x_1 = x_2$ as a function of the number of words n the two fields have in common: $P(x_1 = x_2 \mid n) = 1/(1 + e^{-wn})$, where w is the weight of the formula. Effectively, then, reverse predicate equivalence applied to `HasWord()` implements a simple similarity measure between fields. This measure can be made adaptive, in the style of Bilenko and Mooney [3], by learning a different weight for each grounding of reverse predicate equivalence with a different word. (Groundings where $y_1 \neq y_2$ are always satisfied, and therefore their weights cancel out in the numerator and denominator of Equation 3, and are irrelevant. Reflexivity ensures the proper treatment of groundings where $y_1 = y_2$.)

It can also be useful to add the "negative" version of reverse predicate equivalence:

$$\forall x_1, x_2, y_1, y_2\ \neg\texttt{HasWord}(x_1, y_1) \wedge \texttt{HasWord}(x_2, y_2) \\ \wedge\ y_1 = y_2 \Rightarrow x_1 \neq x_2$$

$$\forall x_1, x_2, y_1, y_2\ \texttt{HasWord}(x_1, y_1) \wedge \neg\texttt{HasWord}(x_2, y_2) \\ \wedge\ y_1 = y_2 \Rightarrow x_1 \neq x_2$$

$$\forall x_1, x_2, y_1, y_2\ \neg\texttt{HasWord}(x_1, y_1) \wedge \neg\texttt{HasWord}(x_2, y_2) \\ \wedge\ y_1 = y_2 \Rightarrow x_1 = x_2$$

Like other word-based similarity measures, this approach has the disadvantage that it treats misspellings, variant spellings and abbreviations of a word as completely different words. Since these are often a significant issue in entity resolution, it would be desirable to account for them. One way to do this efficiently is to compare word strings by the engrams they contain [19]. This can be done in our framework by defining the predicate $\texttt{HasEngram}(\texttt{word}, \texttt{engram})$, which is true iff engram is a substring of word. (This predicate can be computed on the fly from its arguments, or pre-computed for all relevant word-engram pairs and given as evidence.) Applying reverse predicate equivalence to this predicate results in a logistic regression model for the equality of two words as a function of the number of engrams they have in common. Combined with the logistic regression for field equality, this produces a two-level similarity measure, comparing fields as sets of words, which are in turn compared as strings. Cohen et al. [6] found such hybrid measures to outperform pure word-based and pure string-based ones for entity resolution. The maximum length of engrams to consider is a parameter of the problem. Linear-interpolation smoothing is obtained by defining different predicates for engrams of different length, with the corresponding weights for the corresponding versions of reverse predicate equivalence. (These can also be learned, using weight priors to avoid overfitting.) It is also possible to incorporate string edit distances like Levenshtein and Needleman-Wunsch [3] into an MLN. This involves stating formulas analogous to the recurrence relations used to compute these distances, with (negative) weights corresponding to the costs of insertion, deletion, etc. Pursuing this approach is an item for future work.

5.4 The Fellegi-Sunter Model

The Fellegi-Sunter model uses naive Bayes to predict whether two records are the same, with field comparisons as the predictors [14]. If the predictors are the field match predicates, Fellegi-Sunter is a special case of reverse predicate equality, with R as the relation between a field and the record it appears in (e.g., $\texttt{HasAuthor}(\texttt{paper}, \texttt{author})$). If the predictors are field similarities, measured by the number of words present in both, either and none of the fields, Fellegi-Sunter is implemented by clauses of the form

$$\forall x_1, x_2, y_1, y_2 \;\; \texttt{HasWord}(x_1, y_1) \wedge \texttt{HasWord}(\texttt{x}_2, \texttt{y}_2) \\ \wedge\, y_1 = y_2 \wedge R(z_1, x_1) \wedge R(z_2, x_2) \Rightarrow z_1 = z_2$$

$$\forall x_1, x_2, y_1, y_2 \;\; \neg\texttt{HasWord}(x_1, y_1) \wedge \texttt{HasWord}(\texttt{x}_2, \texttt{y}_2) \\ \wedge\, y_1 = y_2 \wedge R(z_1, x_1) \wedge R(z_2, x_2) \Rightarrow z_1 \neq z_2$$

$$\forall x_1, x_2, y_1, y_2 \;\; \texttt{HasWord}(x_1, y_1) \wedge \neg\texttt{HasWord}(\texttt{x}_2, \texttt{y}_2) \\ \wedge\, y_1 = y_2 \wedge R(z_1, x_1) \wedge R(z_2, x_2) \Rightarrow z_1 \neq z_2$$

$$\forall x_1, x_2, y_1, y_2 \;\; \neg\texttt{HasWord}(x_1, y_1) \wedge \neg\texttt{HasWord}(\texttt{x}_2, \texttt{y}_2) \\ \wedge\, y_1 = y_2 \wedge R(z_1, x_1) \wedge R(z_2, x_2) \Rightarrow z_1 = z_2$$

$$\forall z_1, z_2 \;\; z_1 \neq z_2$$

for each field-record relation R. The last rule corresponds to the class priors implemented as a unit clause.

5.5 Relational Models

The combination of Fellegi-Sunter with transitivity produces McCallum and Wellner's [28] conditional random field (CRF) model, with field matches or field similarities as the features. (A logistic regression model is a CRF where all the query variables are independent; transitive closure renders them dependent. Discriminatively-trained MLNs can be viewed as relational extensions of CRFs.)

Reverse predicate equivalence applied to the relations in the database yields the CRF model of Singla and Domingos [42], with the field similarity measure described above instead of TF-IDF. It is also very similar to the CRF model of Culotta and McCallum [9]. The model of Dong et al. [12] is also of this type, but more *ad hoc*. All of these models have the property that they allow entities of multiple types to be resolved simultaneously, with inference about one pair of entities triggering inferences about related pairs of entities (e.g., if two papers are the same, their authors are the same; and vice-versa, albeit with lower weight).

We can also use coauthorship relations for entity resolution, in the vein of Bhattacharya and Getoor [2], by defining the $\texttt{Coauthor}(x_1, x_2)$ predicate using the formula

$$\forall x, y_1, y_2 \;\; \texttt{HasAuthor}(x, y_1) \wedge \texttt{HasAuthor}(x, y_2) \\ \Rightarrow \texttt{Coauthor}(y_1, y_2)$$

with infinite weight, and applying reverse predicate equivalence to it. (We can also explicitly state that coauthorship is reflexive and symmetric, but this is not necessary.) Notice that this approach increases the likelihood that two authors are the same even if they are coauthors of a third author on *different* papers. While this is still potentially useful, a presumably stronger regularity is captured by the clause

$$\forall x_1, x_2, y_1, y_2, y_3, y_4 \; \texttt{HasAuthor}(x_1, y_1) \\ \wedge\, \texttt{HasAuthor}(x_2, y_2) \wedge \texttt{HasAuthor}(x_1, y_3) \\ \wedge\, \texttt{HasAuthor}(x_2, y_4) \wedge x_1 = x_2 \wedge y_1 = y_2 \Rightarrow y_3 = y_4.$$

This formula is related to reverse predicate equivalence, but (with suitably high weight) represents the non-linear increase in evidence that can occur when both the papers and the coauthors are the same.

So far, we have seen that an MLN with a small number of hand-coded formulas representing properties of equality is sufficient to perform state-of-the-art entity resolution. However, one of the key features of the entity resolution problem is that a wide variety of knowledge, much of it domain-dependent, can (and needs to) be brought to bear on matching decisions. This knowledge can be incorporated into our approach by expressing it in first-order logic. For example, the constraints listed by Shen et al. [40] can all be incorporated into an MLN in this way. Weighted satisfiability then performs the role of relaxation labeling in Shen at al. It is also possible to incorporate formulas learned independently using ILP techniques, as in Davis et al. [10], or to refine the hand-coded formulas and construct additional ones using MLN structure learning [23].

5.6 Scalability

In practice, even for fairly small databases there will be a large number of equality atoms to infer (e.g., 1000 constants of one type yield a million equality atoms). However, the vast majority of these will usually be false (i.e., non-matches). Scalability is typically achieved by performing inference only over plausible candidate pairs, identified using a cheap similarity measure (e.g., TF-IDF computed using a reverse index from words to fields). This approach can easily be incorporated into our framework simply by adding all the non-plausible matches to the evidence as false atoms. Most clauses involving these atoms will always be satisfied, and thus there is no need to ground them. We select plausible candidates using McCallum et al.'s canopy approach [26], with TF-IDF cosine as the similarity measure, but any other approach could be used. While some of the apparent non-matches might be incorrect, this is a necessary and very reasonable approximation. Notice that reverse predicate equivalence might be able to correct some of these errors, by indirectly inferring that two very different strings in fact represent the same object. We have developed a version of MaxWalkSAT and MCMC that lazily grounds predicates and clauses, effectively allowing predicates that are initially assumed false to be revisited, without incurring the computational cost of completely grounding the network [43]. Incorporating this into our entity resolution system is an item for future work.

6 Experiments

6.1 Datasets

We used two publicly available citation databases in our experiments: Cora and BibServ.

6.1.1 Cora

The hand-labeled Cora dataset is provided by McCallum[2] and has previously been used by Bilenko and Mooney [3] and others. This dataset is a collection of 1295 different citations to computer science research papers from the Cora Computer Science Research Paper Engine. The original dataset contains only unsegmented citation strings. Bilenko and Mooney [3] segmented each citation into fields (author, venue, title, publisher, year, etc.) using an information extraction system. We used this processed version of Cora. We further cleaned it up by correcting some labels. This cleaned version contains references to 132 different research papers. We used only the three most informative fields: first author, title and venue (with venue including conferences, journals, workshops, etc.). We compared the performance of the algorithms for the task of de-duplicating citations, authors and venues. For training and testing purposes, we hand-labeled the field pairs. The labeled data contains references to 50 authors and 103 venues. After forming canopies, the total number of match decisions was 61,177.

6.1.2 BibServ

BibServ.org is a publicly available repository of about half a million pre-segmented citations. It is the result of merging citation databases donated by its users, CiteSeer, and DBLP. We experimented on a subset of 10,000 records extracted randomly from the user-donated subset of BibServ, which contains 21,805 citations. As in Cora, we used the first author, title and venue fields. In order to focus only on hard decisions, we discarded all the canopies of size less than 10. This left us with a total of 15,954 decisions. Since we lacked labeled data for BibServ, we used the parameters learned on Cora (with appropriate modifications) to perform inference on BibServ. Word stemming was used to identify the variations of the same underlying root word both in Cora and BibServ.

6.2 Models

We compared the following models in our experiments.

NB. This is the naive Bayes model as described in Section 5.4. We use a different feature for every word.

MLN(B). This is the basic MLN model closest to naive Bayes. It has the four reverse predicate equivalence rules connecting each word to the corresponding field/record match predicate. With per-word weights, this yields thousands of weights. For speed, these are derived from the naive Bayes parameters. The model also has a unit clause, $\neg\texttt{SameEntity}(e1, e2)$, for each entity type being matched. The weights of these rules are learned discriminatively using the algorithm of Singla and Domingos [41]. When the weights of single predicate rules are set using the class priors in naive Bayes, the two models are equivalent. All the following models are obtained by adding specific rules to the basic MLN model and learning these rules discriminatively.

[2]www.cs.umass.edu/~mccallum/data/cora-refs.tar.gz

MLN(B+C). This is obtained by adding reverse predicate equivalence rules for the various fields to MLN(B). This achieves the flow of information through shared attribute values as proposed by Singla and Domingos [42].

MLN(B+T). This is obtained by adding transitive closure rules to MLN(B). It incorporates transitivity into the model itself as proposed by McCallum and Wellner [28].

MLN(B+C+T). This model has both reverse predicate equivalence and transitive closure rules and i.e. it has the enhancements proposed by both Singla and Domingos [42] and McCallum and Wellner [28].

MLN(B+C+T+S). This model is obtained by adding rules learned using the structure learning algorithm of Kok and Domingos [23] to MLN(B+C+T). An example of a rule added by structure learning is: if two papers have the same title and the same venue, then they are the same paper.

MLN(B+N+C+T). This model has a two-level learning/inference step. The first step involves learning a model to predict if two word mentions are the same based on the n-grams they have in common (we used n=3). This step incorporates word-level transitivity and reverse predicate equivalence rules. The second step involves learning an MLN(B+C+T) model on the words inferred by the first stage.[3] This model implements a hybrid similarity measure as proposed by Cohen et al. [6].

MLN(G+C+T). This model is similar to MLN(B+C+T) except that it does not have per-word reverse predicate equivalence rules. Rather, it has four global rules, each with the same weight for all words, and these weights are learned discriminatively, like the rest.[4]

6.3 Methodology

In the Cora domain, we performed five-fold cross validation. In the BibServ domain, because of the absence of labeled data, the naive Bayes model could not be learned. The weights of the inverse predicate equivalence rules for each word were fixed in proportion to the IDF of the word. This was used as a baseline for all the enhanced models. The weights of other rules were determined from the weights learned on Cora for the corresponding model.

[3]This model was learned on raw words without any stemming, to see if the n-gram step could potentially achieve the effects of stemming.

[4]The rule $\neg$HasWord() $\wedge$ $\neg$HasWord() $\Rightarrow$ SameField() has essentially no predictive power; we set its weight to 0.

For each model, we measured the conditional log-likelihood (CLL) and area under the precision-recall curve (AUC) for the match predicates. The advantage of the CLL is that it directly measures the quality of the probability estimates produced. The advantage of the AUC is that it is insensitive to the large number of true negatives (i.e., ground atoms that are false and predicted to be false). The CLL of a set of predicates is the average over all their groundings of the ground atom's log-probability given the evidence. The precision-recall curve for match predicates is computed by varying the threshold CLL above which a ground atom is predicted to be true. We computed the standard deviations of the AUCs using the method of Richardson and Domingos [36].

6.4 Results

6.4.1 Cora

Table 1 shows the CLL and AUC for the various models on the Cora dataset. For the case of AUC in venues, there is monotonic increase in the performance as we add various collective inference features to the model, with the best-performing model being MLN(B+C+T). This trend shows how adding each feature in turn enhances the performance, giving the largest improvement when all the features are added to the model. For the case of CLL in venues, the performance tends to fluctuate as we add various collective inference features, but the best-performing model is still MLN(B+C+T). For the case of citations, the results are similar, although the improvements as we add features are not as consistent as in case of AUC in venues. For AUC in authors, we have similar results, although the performance gain is much smaller compared to venues and citations. There is more fluctuation in the case of CLL but the collective models are still the best.

MLN(B+C+T+S) helps improve the performance of MLN(B+C+T) on citations and authors (both CLL and AUC) but does not help on venues.

MLN(B+N+C+T) improves performance on authors but not on citations and venues. This shows that while stemming can be a good way of identifying word duplicates in most cases, the engram model is quite helpful when dealing with fields such as authors, where initials are used for the complete word and can not be discovered by stemming alone.

MLN(G+C+T) (the model with global word rules) performs well for citations but less so for authors and venues. In general, per-word rule models seem to be particularly helpful when there are few words from which to infer the relationship between two entities.

Table 1. Experimental results on the Cora database.

	Citation		Author		Venue	
System	CLL	AUC	CLL	AUC	CLL	AUC
NB	-0.637 ± 0.010	0.913 ± 0.000	-0.133 ± 0.021	0.986 ± 0.000	-0.747 ± 0.017	0.738 ± 0.002
MLN(B)	-0.643 ± 0.010	0.915 ± 0.000	-0.131 ± 0.022	0.987 ± 0.000	-0.760 ± 0.017	0.736 ± 0.002
MLN(B+C)	-0.809 ± 0.012	0.891 ± 0.000	-0.386 ± 0.064	0.968 ± 0.000	-1.163 ± 0.034	0.741 ± 0.001
MLN(B+T)	-0.369 ± 0.003	0.949 ± 0.000	-0.213 ± 0.036	0.994 ± 0.000	-1.036 ± 0.029	0.745 ± 0.002
MLN(B+C+T)	-0.597 ± 0.007	0.964 ± 0.000	-0.171 ± 0.043	0.984 ± 0.000	-0.704 ± 0.023	0.828 ± 0.002
MLN(B+C+T+S)	-0.503 ± 0.006	0.988 ± 0.000	-0.100 ± 0.033	0.992 ± 0.000	-0.874 ± 0.027	0.807 ± 0.002
MLN(B+N+C+T)	-0.879 ± 0.008	0.952 ± 0.000	-0.096 ± 0.032	0.992 ± 0.000	-0.781 ± 0.023	0.817 ± 0.002
MLN(G+C+T)	-0.394 ± 0.004	0.973 ± 0.000	-0.263 ± 0.053	0.980 ± 0.000	-1.196 ± 0.031	0.743 ± 0.002

6.4.2 BibServ

Since we did not have labeled data for BibServ, we hand-labeled 300 pairs each for citations and venues, randomly selected from the set where the MAP prediction for at least one of the algorithms was different from the others. For authors, we labeled all the potential match decisions, as there were only 240 of them. The results reported below are over these hand-labeled pairs.

Table 2 shows the CLL and AUCs for the various algorithms on the BibServ dataset. As in the case of Cora, for citations and authors, the best-performing models are the ones involving collective inference features. MLN(B+N+C+T) gives the best performance for both CLL and AUC in authors, for the reasons cited before. It is also the best-performing model for AUC in citations, closely followed by other collective models. MLN(B+C) gives the best performance for CLL in citations.

Whereas MLN(G+C+T) performs quite poorly on citations and authors, it in fact gives the best performance on CLL in venues. On AUC in venues, MLN(G+C+T) and MLN(B) are the two best performers. Overall, even though BibServ and Cora have quite different characteristics, applying the parameters learned on Cora to BibServ still gives good results. Collective inference is clearly useful, except for AUC on venues, where the results are inconclusive.

7 Future Work

Directions for future work include:

- Using the EM algorithm applied to MLNs to learn weights for entity resolution without labeled data.
- Allowing context-dependent match decisions (e.g., `J. Smith` may represent two different authors of IJCAI papers, but always the same author of SIGMOD papers).
- Specializing reverse predicate equivalence for one-to-one and one-to-many predicates (e.g., a paper has only one venue, but multiple authors).
- Explicitly representing entities (and not just references to them) by using the predicates $\texttt{Represents}(x_1, y)$ and $\texttt{Represents}(x_2, y)$ instead of $x_1 = x_2$. (This is more complex, but has a number of advantages [29].)
- Integrating further components and aspects of the entity resolution problem.
- Applying our approach to other entity resolution domains besides citation matching.
- Extending our approach to schema and ontology matching, by allowing predicate variables (second-order logic) and applying the equality axioms and reverse predicate equivalence to them.
- Extending our approach to other types of data besides relational databases (e.g., tree-structured data like XML, using $\texttt{Child}(x, y)$ predicates, and text data, by incorporating parsing rules into the MLN).
- Generalizing across database sizes by learning weights that depend on the number of groundings.
- Combining entity resolution and data mining in one learning and inference process, by performing MLN structure learning over the data at any level of resolution.

8 Conclusion

This paper proposes a unifying framework for entity resolution. We show how a small number of axioms in Markov logic capture the essential features of many different approaches to this problem, in particular non-i.i.d. ones, as well as the original Fellegi-Sunter model. Experiments on two citation databases evaluate the contributions of these

Table 2. Experimental results on the BibServ database.

	Citation		Author		Venue	
System	CLL	AUC	CLL	AUC	CLL	AUC
MLN(B)	−0.008±0.003	0.997±0.001	−0.586±0.114	0.910±0.013	−0.806±0.121	0.908±0.011
MLN(B+C)	−0.001±0.000	0.999±0.000	−0.544±0.113	0.887±0.007	−1.166±0.151	0.876±0.012
MLN(B+T)	−0.006±0.003	0.993±0.003	−0.600±0.116	0.909±0.013	−0.827±0.123	0.898±0.010
MLN(B+C+T)	−0.006±0.004	0.998±0.000	−0.473±0.105	0.928±0.009	−1.146±0.149	0.876±0.012
MLN(B+C+T+S)	−0.006±0.004	0.970±0.020	−0.486±0.107	0.926±0.010	−1.133±0.148	0.876±0.012
MLN(B+N+C+T)	−0.018±0.005	1.000±0.000	−0.363±0.091	0.940±0.008	−0.936±0.133	0.897±0.012
MLN(G+C+T)	−0.735±0.101	0.491±0.000	−4.679±0.256	0.432±0.001	−0.716±0.112	0.906±0.012

approaches, and illustrate how Markov logic enables us to easily build a sophisticated entity resolution system.

Acknowledgements

This research was partly funded by DARPA contract NBCHD030010/02-000225, DARPA grant FA8750-05-2-0283, NSF grant IIS-0534881, and ONR grant N-00014-05-1-0313. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

[1] A. Agresti. *Categorical Data Analysis*. Wiley, New York, NY, 1990.

[2] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *Proc. SIGMOD-04 DMKD Wkshp.*, 2004.

[3] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. KDD-03*, pages 39–48, 2003.

[4] M. Bilenko and R. Mooney. On evaluation and training-set construction for duplicate detection. In *Proc. KDD-03 Wkshp. on Data Cleaning, Record Linkage, and Object Consolidation*, pages 7–12, 2003.

[5] W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *Proc. KDD-00*, pages 255–259, 2000.

[6] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *Proc. KDD-03 Wkshp. on Data Cleaning, Record Linkage, and Object Consolidation*, pages 13–18, 2003.

[7] W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proc. KDD-02*, pages 475–480, 2002.

[8] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP-02*, Philadelphia, PA, 2002.

[9] A. Culotta and A. McCallum. Joint deduplication of multiple record types in relational data. In *Proc. CIKM-05*, pages 257–258, 2005.

[10] J. Davis, I. Dutra, D. Page, , and V. Costa. Establishing identity equivalence in multi-relational domains. In *Proc. ICIA-05*, 2005.

[11] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–392, 1997.

[12] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proc. SIGMOD-05*, pages 85–96, 2005.

[13] S. Džeroski and H. Blockeel, editors. *Proceedings of the KDD-04 Workshop on Multi-Relational Data Mining*. ACM Press, Chicago, IL, 2004.

[14] I. Fellegi and A. Sunter. A theory for record linkage. *J. American Statistical Association*, 64:1183–1210, 1969.

[15] A. Fern, L. Getoor, and B. Milch, editors. *Proceedings of the ICML-2006 Workshop on Open Problems in Statistical Relational Learning*. IMLS, Pittsburgh, PA, USA, 2006.

[16] J. Gehrke, P. Ginsparg, and J. Kleinberg. KDD cup 2003. Online bibliography, SIGKDD, 2003. http://-www.cs.cornell.edu/projects/kddcup.

[17] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1987.

[18] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, UK, 1996.

[19] L. Gravano, P. Ipeirotis, H. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava. Using q-grams in a DBMS for approximate string processing. *IEEE Data Engineering Bulletin*, 24(4):28–34, 2001.

[20] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proc. SIGMOD-95*, pages 127–138, 1995.

[21] T. Huang and S. Russell. Object identification: A Bayesian analysis with application to traffic surveillance. *Artificial Intelligence*, 103:1–17, 1998.

[22] H. Kautz, B. Selman, and Y. Jiang. A general stochastic approach to solving problems with hard and soft constraints. In D. Gu, J. Du, and P. Pardalos, editors, *The Satisfiability Problem: Theory and Applications*, pages 573–586. American Mathematical Society, New York, NY, 1997.

[23] S. Kok and P. Domingos. Learning the structure of Markov logic networks. In *Proc. ICML-05*, pages 441–448, Bonn, Germany, 2005. ACM Press.

[24] S. Kok, P. Singla, M. Richardson, and P. Domingos. The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2005. http://www.-cs.washington.edu/ai/alchemy.

[25] X. Li, P. Morie, and D. Roth. Semantic integration in text: from ambiguous names to identifiable entities. *AI Magazine*, 26(1):45–58, 2005.

[26] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. KDD-00*, pages 169–178, 2000.

[27] A. McCallum, S. Tejada, and D. Quass, editors. *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*. ACM Press, 2003.

[28] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Adv. NIPS 17*, pages 905–912, 2005.

[29] B. Milch, B. Marthi, D. Sontag, S. Russell, and D. L. Ong. BLOG: Probabilistic models with unknown objects. In *Proc. IJCAI-05*, pages 1352–1359, Edinburgh, Scotland, 2005.

[30] A. Monge and C. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proc. SIGMOD-97 DMKD Wkshp.*, 1997.

[31] H. Newcombe, J. Kennedy, S. Axford, and A. James. Automatic linkage of vital records. *Science*, 130:954–959, 1959.

[32] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, 1999.

[33] G. Norén, R. Orre, and A. Bate. A hit-miss model for duplicate detection in the WHO Drug safety Database. In *Proc. KDD-05*, pages 459–468, Chicago, IL, 2005.

[34] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *Adv. NIPS 15*, pages 1401–1408, 2003.

[35] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[36] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[37] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

[38] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proc. KDD-02*, pages 269–278, 2002.

[39] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, pages 521–532. American Mathematical Society, Washington, DC, 1996.

[40] W. Shen, X. Li, and A. Doan. Constraint-based entity matching. In *Proc. AAAI-05*, pages 862–867, Pittsburgh, PA, 2005. AAAI Press.

[41] P. Singla and P. Domingos. Discriminative training of Markov logic networks. In *Proc. AAAI-05*, pages 868–873, Pittsburgh, PA, 2005. AAAI Press.

[42] P. Singla and P. Domingos. Object identification with attribute-mediated dependences. In *Proc. PKDD-05*, pages 297–308, Porto, Portugal, 2005. springer.

[43] P. Singla and P. Domingos. Memory-efficient inference in relational domains. In *Proc. AAAI-06*, pages 488–493, Boston, MA, 2006. AAAI Press.

[44] S. Tejada, C. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.

[45] S. Tejada, C. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. KDD-02*, pages 350–359, 2002.

[46] W. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.