Constraint-Based Entity Matching

Warren Shen **Xin Li** **AnHai Doan**
University of Illinois, Urbana, USA
{whshen, xli1, anhai}@cs.uiuc.edu

Abstract

Entity matching is the problem of deciding if two given mentions in the data, such as "Helen Hunt" and "H. M. Hunt", refer to the same real-world entity. Numerous solutions have been developed, but they have not considered in depth the problem of exploiting integrity constraints that frequently exist in the domains. Examples of such constraints include "a mention with age two cannot match a mention with salary 200K" and "if two paper citations match, then their authors are likely to match in the same order". In this paper we describe a probabilistic solution to entity matching that exploits such constraints to improve matching accuracy. At the heart of the solution is a generative model that takes into account the constraints during the generation process, and provides well-defined interpretations of the constraints. We describe a novel combination of EM and relaxation labeling algorithms that efficiently learns the model, thereby matching mentions in an unsupervised way, without the need for annotated training data. Experiments on several real-world domains show that our solution can exploit constraints to significantly improve matching accuracy, by 3-12% F-1, and that the solution scales up to large data sets.

Introduction

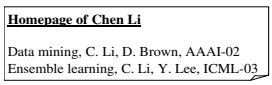
Entity matching decides if two given *mentions* in the data, such as "Helen Hunt" and "H. M. Hunt", refer to the same real-world entity. This problem plays a key role in information integration, natural language understanding, information processing on the World-Wide Web, and on the emerging Semantic Web. As such, it has received significant attention in the AI, database, data mining, and Web communities. Variants of the problem are known as identity uncertainty, tuple matching, deduplication, record linkage, and mention matching, among others.

Numerous entity matching solutions have been developed. Early solutions employ manually specified rules (Hernandez & Stolfo 1995), while subsequent works focus on learning the rules (Tejada, Knoblock, & Minton 2002; Bilenko & Mooney 2003), matching strings efficiently (Cohen 1998; Cohen, Ravikumar, & Fienberg 2003), clustering a large number of tuples (McCallum, Nigam, & Ungar 2000; Cohen & Richman 2002), personal information management (Dong *et al.* 2005), exploiting links (Bhattacharya & Getoor 2004), matching mentions in text (Li, Morie, & Roth 2004), modeling matching with generative models (Pasula *et al.* 2003; Ravikumar & Cohen 2004; Li, Morie, & Roth 2004) and with conditional random fields (Parag & Domingos 2004; Wellner *et al.* 2004).

While significant progress has been made, virtually all of the above works have focused on exploiting *syntactic similarities* (e.g., those between two names or two addresses) to match mentions. Real-world applications however often have many *semantic integrity constraints*, such as "mentions in the PC listing of a conference refer to different researchers" and "no researcher has published more than five papers in AAAI in a year". The constraints can be either learned from some external data or specified by a domain user (Doan *et al.* 2003). Exploiting such constraints can significantly improve matching accuracy, and indeed some recent works have examined this issue (Doan *et al.* 2003; Dong *et al.* 2005). However, they have only exploited simple pairwise hard constraints that prevent two given mentions from matching, in ad-hoc and often non-scalable ways.

In this paper we describe CME, an entity matching solution that exploits a broad variety of semantic constraints in a principled and scalable manner. We begin by defining an entity matching problem that generalizes most current problem settings that match entities in an unsupervised way, without expensive training data. Next, we describe a broad variety of semantic constraints that can be exploited in entity matching. In contrast to prior works (e.g., (Dong *et al.* 2005)), the constraints here can be both hard (i.e., must always be satisfied) and soft (i.e., are likely to be satisfied). They can also be aggregate constraints involving groups of mentions. We show that all these types of constraints can be expressed explicitly in a probabilistic framework.

We then describe and motivate a two-layer architecture for entity matching. At the heart of the first layer is a generative model on how data sets that satisfy constraints are generated. This model builds on the generative model recently proposed in (Li, Morie, & Roth 2004), but significantly extends it to take into account the constraints, and to handle a wide variety of attribute formats (see the section on generative model). We then develop a novel combination of the EM algorithm and the relaxation labeling algorithm to perform matching. Briefly, the matching process is carried out in multiple iterations. In each iteration the EM algorithm estimates the parameters of the generative model and a matching assignment, then employs relaxation labeling to exploit the constraints, to significantly improve the accuracy of the above estimates. Relaxation labeling has been successfully



Title	Authors	Conf.	Year
Entity matching	Chen Y. Li, David Brown	IJCAI	2001
Wrapper induction	Chang Li, Jane Smith	KDD	2002

Figure 1: A sample data set in the publication domain.

employed in computer vision, hypertext classification, and ontology matching (e.g., (Chakrabarti, Dom, & Indyk 1998; Doan *et al.* 2002)), but to our knowledge, not yet to entity matching. The key advantage of relaxation labeling is that it scales up to very large data sets and that it can accommodate a wide range of domain constraints.

The first layer in effect "clusters" mentions into groups (such that all matching mentions belong to the same group) and exploits constraints at the *group* level. Once this is done, the second layer exploits additional constraints at the level of *individual* matching mention pair. This two-layer architecture is in sharp contrast to prior works, which employ only the first or the second layer. In this paper, we show that exploiting constraints within a single layer architecture can significantly reduce the matching accuracy.

Finally, we note that matching mentions is often just a first step in a broader integration process. Once automatic matching is done, users often want to examine the results, provide feedback, and try out various "what-if" scenarios. As a side benefit, our constraint exploitation framework enables a relatively simple but effective method for such user interaction. We model user feedback as "temporary" domain constraints, then rerun relaxation labeling taking into account these constraints. Because relaxation labeling is guaranteed to be fast, we can in effect provide an interactive setting for the users to explore the matches. As far as we know, such fast interactive settings have not been considered in prior works on entity matching. In the rest of the paper we describe our entity matching solution. The full version of this paper can be found at http://anhai.cs.uiuc.edu/home.

Problem Definition

Data Sets, Entities, Mentions, & Attributes: Let D be a data set consisting of n documents $d_1, \ldots, d_n$, where each d_i is a text article or a relational database record. Let M_D be a set of mentions $m_1, \ldots, m_k$ of real-world entities in data set D. The entity matching problem is to find pairs of mentions that match, that is, refer to the same entities.

For example, Figure 1 shows a simplified data set that consists of three documents: a single text article and two relational records. Examples of mentions include "Chang Li" and "D. Brown" for persons, "AAAI" and "KDD" for conferences, and "Ensemble learning" for papers. Given this data set, our goal is to find out that "C. Li" matches "Chen Y. Li", "D. Brown" matches "David Brown", and so on.

As the first step in exploiting constraints, we focus here on the case of matching mentions of a *single* type of entity (e.g., persons). Our solution however generalizes to the more general setting of matching multiple types of entities simultaneously. We assume that each mention is associated with a set of *attributes*. Mentions of persons for example can be associated with attributes such as title (e.g., "Mr.", "Prof."), first name, last name, age, salary, address, etc. We then match mentions by utilizing the values of their attributes.

Type	Example
Aggregate	c1 = No researcher has published more than five AAAI papers in a year.
Subsumption	c2 = If a citation X from DBLP matches a citation Y in a homepage, then each author mentioned in Y matches some author mentioned in X.
Neighborhood	c3 = If authors X and Y share similar names and some co-authors, they are likely to match
Incompatible	c4 = No researcher exists who has published in both HCI and numerical analysis.
Layout	c5 = If two mentions in the same document share similar names, they are likely to match.
Key/Uniqueness	c6 = Mentions in the PC listing (e.g. of a conference) refer to different researchers.
Ordering	c7 = If two citations match, then their authors will be matched in order
Individual	c8 = The researcher with the name "Mayssam Saria" has fewer than five mentions in DBLP (e.g., being a new graduate student with fewer than five papers).

Figure 2: A broad variety of constraints can be defined and exploited in entity matching.

It is important to emphasize that we do not deal with the problem of extracting mentions and their attributes from text or relational records. This problem has received much attention in the AI, database, KDD, and WWW communities, within the context of named entity recognition, information extraction, and text segmentation (e.g., (Agichtein & Ganti 2004; Borkar, Deshmukh, & Sarawagi 2001; Freitag 1998)). Hence, in this paper we focus on the problem of matching mentions, given their attributes. Attribute values are string or numeric, and can be missing for certain mentions.

Domain Constraints: To match mentions, we can often exploit a broad range of domain constraints. Figure 2 shows sample constraints in the publication domain. The constraints can be either learned from the data, or specified by a domain expert or user (Doan *et al.* 2003). This is done only *once*, at the start of the matching process. They can then be re-used across matching problems in the same domain. As Figure 2 shows, the constraints capture the user's knowledge about relationships among document layouts, order of mentions, aggregate properties of real-world entities, and semantic properties of database columns (e.g., a column being a key means each of its mentions must refer to a different entity), among others. A constraint such as c_1 = "no researcher has published more than five AAAI papers in a year" is *hard* in that it must be satisfied. In contrast, a constraint such as c_5 = "if two mentions in the same document share similar names, they are likely to match" is *soft*, in that it is not always satisfied.

The constraints shown in Figure 2 are specific to the publication domain. However, similar constraints can be specified in many other domains as well. For example, in the movie domain we can also specify an aggregate constraint such as "no director has produced more than seven movies in a single year".

We model each constraint by specifying its effects on the *probability* of a mention referring to a real-world entity. For example, constraint c_5 can be modeled as $P(m = e|\ \exists\ m'\ \ st.\ \ [m\ \&\ m'\ are\ \ in\ \ the\ \ same\ \ document] \wedge [name(m) \approx name(m')] \wedge [m' = e]) = 0.8$, where m and m' are mentions in M_D, and $m = e$ means mention m refers to real-world entity e. As another example, the constraint "no person has age 2 and salary 200K" can be modeled as $P(m = e|\ \ [salary(m) = 200K]\ \wedge\ [\ \exists\ \ m'\ \ st.\ \ (m' = e) \wedge (age(m') = 2)\ ]) = 0$.

In general, each constraint c_i is modeled as $P(m = e|f_i(O_m, m, e) = true) = p_i$, where O_m is an assignment of all mentions other than m to entities, and f_i is a binary

function that describes a characteristic of this assignment.

Thus, a key distinguishing aspect of our work is that both hard and soft constraints can be modeled in a uniform and intuitive way, with well-defined probabilistic meaning (for their "weights"). For a hard constraint, the probability of m referring to e is 0 or 1, while for a soft constraint, it is a value between 0 and 1. As we show below, this value can be set by the user, based on his or her domain experience, or learned in an unsupervised way from the same data set to be matched.

The Mention Matching Problem: We now can define the mention matching problem as follows: given a data set D with M_D mentions, find matching mention pairs, utilizing the attributes $a_1, \ldots, a_k$ of the mentions, and taking into account a set of constraints $c_1, \ldots, c_l$. This is a very general problem setting that subsumes those in record linkage and mention matching in text. Furthermore, while this problem setting is unsupervised, in that it does not assume the availability of expensive annotated training data, the solution we offer below can be generalized to supervised settings as well.

The CME Solution

We now describe mention matching with CME. This section describes the first layer which employs a generative model and exploits constraints in a "global manner" to match mentions. The next section describes the second layer which exploits constraints in a pairwise manner and user interaction with CME.

The Generative Model

At the heart of the first layer is a model that generates data sets to satisfy domain constraints as follows. To create a data set D, a data creator generates document d_1, then d_2, and so on. To generate d_1, the creator randomly chooses a number num_e, then selects a set of num_e entities E_1 from the set of all possible real-world entities E, according to a probability $P(E_1)$. Next, for each entity $e \in E_1$, he or she randomly chooses a number num_m, then generates num_m mentions. Each mention m is independently generated from e according to a transformation probability $P(m|e)$. The mentions are then randomly "sprinkled" into document d_1. Once the data set $D = \{d_1, \ldots, d_n\}$ has been generated, the creator checks if it satisfies the supplied constraints $C = \{c_1, \ldots, c_l\}$. If yes, then the data set is retained, otherwise it is discarded, and the generation process is repeated until a data set satisfying the constraints is created.

Constraint checking works as follows. The creator considers each mention m_i in D in a sequential order. Suppose that constraints $c_{i1}, \ldots, c_{it}$ *can* apply to m_i. Consider a constraint c_{ij}. Suppose it has a probability p_{ij} (i.e., of a mention m referring to entity e, as described in the problem definition section). Then the creator decides with probability p_{ij} if c_{ij} should apply to mention m_i. If yes, then he or she checks if c_{ij} is violated with respect to m_i. If yes, then the dataset D is considered violating the constraints, and is discarded. D is considered not violating the constraints only if all of its mentions do not violate any constraint.

Example 1 *Figure 3 shows how a simplified data set of two documents is generated. First, two entities e_1 and e_2 are selected. They both happen to be persons with name "Chen Li". Next, two mentions "Chen Li" and "Chen" of e_1 as well as a single mention "C. Li" of e_2 are generated and sprinkled into the first document. The second document is generated in a similar manner. Next, constraints are checked. Suppose the only constraint is c_5 shown in Figure 2, with associated probability 0.8. The creator now flips a coin with probability 0.8 to decide if c_5 should apply to the first mention "Chen Li" in the first document. Suppose it does. Then this mention violates the constraint, because the mention "C. Li" in the same document shares a similar name, but refers to a different entity (e_2 instead of e_1). The two documents are then discarded, and the generation process repeats.*

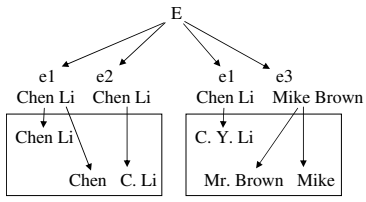


Figure 3: A sample generation process.

Mention Matching: From the above discussion, we can list the parameters θ of the generative model as follows:

- a set of entities E_D and a distribution $P(E_D)$ over it;
- probability distributions over num_e and num_m; we assume these to be uniform over a small plausible range;
- mention-generation probabilities $P(m|e)$; and
- probabilities $p_1, \ldots, p_l$ of the constraints $c_1, \ldots, c_l$.

Let F be an assignment of mentions in D to entities, we now can view matching mentions of D as the problem of computing an optimal F^* and θ^* such that $P(D, F^*|\theta^*)$ is maximized over all possible F and θ. Clearly, if we know F^*, then we can match mentions, since any two mentions referring to the same entity match.

Learning the Generative Model

In practice, finding F^* and θ^* is impractical, due to an exponential number of possible assignments and model parameters. Hence, we employ a variant of the EM algorithm to iteratively estimate the above two, as follows:

(1) *(Initialization):* Let $t = 0$. Find an initial assignment F_0, which assigns each mention in D to a real-world entity.
(2) *(Maximization):* Compute model parameters $\theta_{t+1} = argmax_\theta\ P(D, F_t|\theta)$.
(3) *(Expectation):* Compute mention assignments $F_{t+1} = argmax_F\ P(D, F|\theta_{t+1})$.
(4) *(Convergence:)* If $[P(D, F_{t+1}|\theta_{t+1}) - P(D, F_t|\theta_t)] \leq \epsilon$, for a pre-specified ϵ, then stop and return F_{t+1}. Otherwise let $t = t + 1$, and go back to Step 2.

We now describe these steps in detail.

Initialization: The initial assignment F_0 assigns each mention $m \in M_D$ to a distinct entity e.

Maximization: We compute θ_{t+1} as follows. The set of entities E_D will be all entities that F_t maps the mentions in M_D into. For example, if all mentions in M_D are assigned to five entities, then E_D consists of those five. For any set of entities $E = \{e_1, \ldots, e_g\}$, we compute $P(E) = P(e_1)P(e_2)\ldots P(e_g)$, because each entity is chosen independently of the others. Each $P(e_i)$ is computed as the fraction of mentions in M_D that refer to e_i. Each probability p_i of a constraint c_i is approximated as the fraction of mentions in M_D that satisfy c_i.

All that is left is to compute $P(m|e)$. Let $m = (a_1 = u_1, \ldots, a_k = u_k)$ and $e = (a_1 = v_1, \ldots, a_k = v_k)$, where the a_i are attributes (we will describe how to compute attribute values for entities shortly). Then by making the simplifying assumption of independence among the attributes, we can compute $P(m|e) = P(u_1|v_1)\ldots P(u_k|v_k)$.

Computing $P(u_i|v_i)$ poses a difficult modeling challenge, since in practice the values u_i and v_i can take on a wide ranging number of formats. We solve this as follows. First, we assume that for each attribute a_i we have defined a distance function q_i that computes the distance between any two of its values. Numerous such distance functions have been described in entity matching literature. Next, we model $P(u_i|v_i)$ as a variant of Gaussian distribution over the distance $q_i(u_i, v_i)$: $P(u_i|v_i) = \frac{1}{\sqrt{\pi/2}\sigma_i} \cdot \exp\left(-q_i(u_i, v_i)^2/\sigma_i^2\right)$. Given the matching mention-entity pairs $\{(m, e)\}_1^n$ as specified by F_t, we compute the maximum likelihood estimation of σ_i as $[\frac{\sum_{(m,e)} q_i(u_i,v_i)^2}{n}]^{1/2}$, where u_i and v_i are the values of attribute a_i of m and e, respectively.

Finally, suppose only the mentions m_1, m_2, m_3 are assigned to an entity e. Then we compute the value for attribute a_i of e as a function over the values for attribute a_i of m_1, m_2, m_3. Currently we take the union of these values, and found this method to work well empirically, although more sophisticated "merging" functions are clearly possible, if the distance functions can utilize them effectively.

Expectation: Next, we compute an optimal assignment F_{t+1} given the estimated model θ_{t+1}. Since the number of possible assignments is exponential, we employ a greedy search in which we assign each mention m to entity e that maximizes $P(e|m)$. We can compute $P(e|m) = P(m|e)P(e)/P(m)$, using $P(m|e)$ and $P(e)$ as estimated in the above maximization step. However, this estimation of the true $P(e|m)$ can be improved further, by exploiting the domain constraints.

To exploit the constraints efficiently, we employ a well-known local optimization procedure called *relaxation labeling* (see the introduction). This procedure iteratively changes the probability of each mention referring an entity, taking into account all constraints involving that mention. In particular, we can write

$$\begin{aligned} P(m = e) &= \sum_{O_m} P(m = e, O_m) \\ &= \sum_{O_m} P(m = e|O_m)P(O_m) \qquad (1) \end{aligned}$$

where the sum is over all possible assignments O_m of entity to all mentions other than m. By making the simplifying assumption that all entity-mention assignments are independent of one another, we can approximate $P(O_m)$ as $\prod_{(m_i=e_i)\in O_m} P(m_i = e_i)$.

Now consider $P(m = e|O_m)$. Recall from the problem definition section that we can model each constraint c_i that applies to m as $P(m = e|f_i(O_m, m, e)) = p_i$. Thus, given the constraints $c_1, \ldots, c_n$ that apply to m, we want to compute $P(m = e|O_m)$ as $P(m = e|f_1, \ldots, f_n)$, the probability of m referring to e, given the n constraints. We use the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$, where x is a linear combination of the features f_k, to estimate the above probability. This function is widely used to combine multiple sources of evidence (Agresti 1990); it does not require these evidence sources to be independent of one another. Thus:

$$P(m = e|f_1, \ldots, f_n) \propto \sigma(\alpha_1 \cdot f_1 + \cdots + \alpha_n \cdot f_n) \qquad (2)$$

where $\propto$ denotes "proportional to". The weight α_k is set so that $P(m = e|f_k) = p_k$, to reflect the natural interpretation that if c_k is true, then m refers to e with probability p_k. The probabilities p_k are learned in the maximization step. Thus, the weights α_k are also learned.

By substituting the various above formulae into Equation 1, we obtain

$$P(m = e) \propto \sum_{O_m} \sigma\left(\sum_{k=1}^{n} \alpha_k f_k(O_m, m, e)\right) \times \prod_{(m_i=e_i)\in O_m} P(m_i = e_i) \qquad (3)$$

The proportionality constant is found by renormalizing the probabilities.

Thus, the expectation step is carried out as follows. First, we compute for each m and e the probability $P(m = e)$ as described in the maximization step. Next, we use Equation 3 to update each $P(m|e)$, in effect "correcting" it using the constraints. This repeats until the $P(m|e)$ converge. Finally, we assign m to e that maximizes $P(e|m) \propto P(m|e)P(e)$.

We have developed several methods to compute $P(m = e)$ in Equation 3 very fast, using dynamic programming. We have also implemented several optimization techniques, which ensure that relaxation labeling takes time only *linear* in the number of mentions. We describe the details of these optimizations in the full paper.

We note further that the above algorithm differs from the traditional EM algorithm in several aspects. First, in the expectation step we perform only a greedy, instead of expected, assignment. Second, the maximization step only approximates the maximum-likelihood θ, instead of computing it exactly, as in traditional EM. We found that these simplifications speed up entity matching and still improve matching accuracy significantly in our experiments.

Exploiting Pairwise Hard Constraints

The generative layer described above assumes that two mentions m_1 and m_2 match if they are assigned to the same entity. Thus, assigning a set of mentions $\{m_1, \ldots, m_k\}$ to

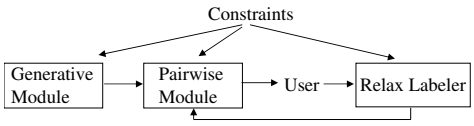


Figure 4: The CME architecture.

the same entity implies that every pair of mentions in that set match. This strong implication can actually cause some constraints to hurt overall matching accuracy.

To illustrate, suppose we have assigned mentions $m_1, \ldots, m_4$ to entity e_1. Suppose that only m_1 really belongs to e_1, and that $m_2 - m_4$ belong to e_2. Now suppose we apply a constraint c_i to this set of mentions, and it can only determine that m_1 does not match m_2. If we reassign m_2 to a different entity and leave the other mentions still assigned to e_1, we will have removed one false positive, pair (m_1, m_2), but introduced two new false negatives (m_2, m_3) and (m_2, m_4). This problem arises because constraint c_i can only make a statement about the pair (m_1, m_2), and not about any other pair. The result is that the increase in precision is offset by a larger drop in recall.

In general, constraints that only apply to a small subset of pairs run into the danger that when applying them, the global implications of the generative layer hurt overall matching accuracy. To address this, we add a second layer that applies hard constraints in a *pairwise* manner. Such a constraint examines a pair of mentions to determine if they could possibly match (e.g., "a person mention with age 2 does not match a person mention with salary 200K").

The second layer conceptually examines all matching pairs returned by the first layer, applies the pairwise hard constraints to filter out false positives, then returns the final matching pairs to the user. We have implemented several optimizations that enable the second layer to avoid examining *all* matching pairs produced by the first layer.

It is important to note that *all* clustering-based matching algorithms suffer from the problem describe above, and thus can benefit from a pairwise constraint second layer. In the experiment section we show that this two-layer architecture improves matching accuracy over the traditional one-layer architecture.

User Interaction: In CME, the user can examine the matching pairs produced by the second layer, then provide some feedback to the system. We model the feedback as "temporary" constraints, and rerun the relaxation labeling algorithm described earlier, taking into account all constraints (including temporary ones). We then perform pairwise constraint checking again, then output the revised matching pairs to the user. The user can then provide feedback again. This process repeats until the user is satisfied. The overall architecture of CME is shown in Figure 4. Notice that such user interaction is possible only if we have a very fast method to exploit user feedback and return the revised matches. Relaxation labeling provides such a method.

Empirical Evaluation

We now present experimental results that demonstrate the utility of exploiting constraints within our framework for mention matching.

Researchers:
Researcher mentions collected from home, group, conference, and DBLP homepages
Citations: *<name, coauthors, title, conference, year>*
Contains 4,991 mentions of 2388 distinct researchers, and 21776 correct matching pairs

IMDB:
Mentions of movies and people collected from IMDB records and text documents
People: *<name, gender, birthdate, birthplace, deathdate, deathplace, movies>*
Movies: *<title, year, genre, runtime, language, country, director, color, rating, actors>*
Contains 3889 movie mentions of 1200 distinct movies, and 25725 correct matching pairs

Figure 5: Characteristics of the data sets

F1 (P / R)	Researchers	IMDB
Baseline	.66 (.67/.65)	.69 (.61/.79)
Baseline + Relax	.78 (.78/.78)	.72 (.63/.83)
Baseline + Relax + Pairwise	.79 (.80/.79)	.73 (.64/.83)

Figure 6: Matching Accuracy

Data Sets: Figure 5 describes the two data sets in our experiments. Researchers contains personal homepages, group homepages, and pages from the CS Bibliography DBLP. IMDB contains news articles from *imdb.com* as well as the IMDB homepages of people and movies. We converted each IMDB homepage into a structured record (see Figure 5 for the record schemas). For each data set, we then employed a combination of automatic and manual methods to mark up the mentions (researcher and movies, respectively) and their attributes. Next, following common research practice in entity matching (e.g., (Hernandez & Stolfo 1995; Li, Morie, & Roth 2004)), we randomly perturbed the mentions and their attributes (e.g., adding misspellings and abbreviations) to generate more ambiguity for experimental purposes. Finally, we manually identified all correct matching pairs, to be used for accuracy evaluation.

Constraints: For Researchers, we employed six constraints, including the subsumption, neighborhood, individual, layout, and incompatible constraints listed in Figure 2. We also added an additional constraint that for conferences that were mentioned rarely in our data set, researchers with similar names who published in those conferences were more likely to match. For IMDB, we employed three constraints: incompatible, neighborhood, and individual. We describe the constraints in detail in the full paper.

Performance Measures: We employ the commonly used metrics of precision, recall, and F-1 to measure performance. Let M_p be the set of matching pairs that an algorithm predicts. Let M_a be the set of all correct matching pairs from the data set. Then, we compute precision $P = |M_p \cap M_a|/|M_p|$, recall $R = |M_p \cap M_a|/|M_a|$, and $F\text{-}1 = (2P \cdot R)/(P + R)$.

Matching Accuracy: Figure 6 shows the matching accuracy of CME variants. The first row, "Baseline", refers to entity matching with the generative model (exploiting no constraints). We note that this "baseline" algorithm in effect implements a current state-of-the-art entity matching algorithm (Li, Morie, & Roth 2004). The next row shows the effects of adding relaxation labeling to exploit constraints. The last row shows the effects of adding the pairwise constraint layer (i.e., the complete CME system).

The results show that for both domains, adding relaxation labeling to exploit constraints (in the generative layer) im-

Baseline	.66 (.67/.65)
+ Rare Value	.66 (.67/.66)
+ Subsumption	.67 (.68/.65)
+ Neighborhood	.70 (.68/.72)
+ Individual	.70 (.77/.64)
+ Layout	.71 (.68/.74)

(a)

Baseline	.69 (.61/.79)
+ Incompatible	.70 (.62/.79)
+ Neighborhood	.70 (.62/.81)
+ Individual	.71 (.62/.82)

(b)

Figure 7: The effects of individual constraints (in the generative layer) on (a) Researchers and (b) IMDB datasets.

proves both precision and recall, resulting in a 12% F-1 increase for Researchers. The F-1 accuracy for IMDB increases less, by 3%. In IMDB, the cluster sizes are more evenly distributed. Thus, when global constraints are applied, the changes are less pronounced compared to Researchers. It is interesting to note that the constraints we used manage to avoid a tradeoff between precision and recall, and instead improves them both simultaneously. Adding the pairwise constraint layer on top of the generative layer further improves matching accuracy by 1%.

To evaluate the utility of the pairwise constraint layer, for Researchers data set, we remove this layer and push all of its constraints into the generative layer. This actually *reduces* the matching accuracy. While it improves precision to 77%, recall dropped significantly to 53%, resulting in an F-1 of 63%. This clearly demonstrates the necessity of enforcing certain constraints as local *pairwise* constraints, which avoids the strong implications that result when exploiting them in the generative layer.

Effects of Individual Constraints: Figure 7 shows the effect of adding individual constraints in the generative layer. The first row shows the accuracy of the baseline algorithm, and each of the other rows show the accuracy of adding one of the constraints in isolation. The results show that most constraints make meaningful contributions to matching accuracy. Note that since more than one constraint may apply to a set of mentions, their effects may overlap. For example, the improvement from applying the incompatible, neighborhood, and individual constraints for IMDB are 1%, 1%, and 2%, respectively. However, the improvement from applying all three together is only 3% (Figure 6). This suggests that some corrections from enforcing one constraint are also brought about by other constraints.

Run Time: We found relaxation labeling to be very fast. For most features, relaxation labeling took less than 12 seconds per iteration on Researchers (7 seconds for IMDB). With some optimizations, the algorithm runs in linear time (in the number of mentions), suggesting that this method can scale to large data sets, and enable an interactive tool for users to dynamically add constraints.

Concluding Remarks

We have described an entity matching solution that can exploit a broad range of domain constraints to significantly improve matching accuracy. The key novelties of the solution include (a) well-defined probabilistic interpretation of the constraints, (b) a significant extension of a previous generative model to handle constraints, (c) a novel combination of EM and relaxation labeling algorithms to exploit constraints efficiently, and (d) a two-layer matching architecture that further improves matching accuracy over existing single-layer ones. Our experiments showed that exploiting constraints improve F-1 accuracy by 3-12% on several real-world data sets. A key direction for our future research is to study how to learn constraints effectively from the current or external data.

References

Agichtein, E., and Ganti, V. 2004. Mining reference tables for automatic text segmentation. In *Proc. of KDD-04*.

Agresti, A. 1990. *Categorical Data Analysis*. NY, Wiley.

Bhattacharya, I., and Getoor, L. 2004. Iterative record linkage for cleaning and integration. In *Proc. of SIGMOD DMKD Workshop*.

Bilenko, M., and Mooney, R. J. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of KDD-03*.

Borkar, V.; Deshmukh, K.; and Sarawagi, S. 2001. Automatic text segmentation for extracting structured records. In *Proc. of SIGMOD-01*.

Chakrabarti, S.; Dom, B.; and Indyk, P. 1998. Enhanced Hypertext Categorization Using Hyperlinks. In *Proc. of SIGMOD-98*.

Cohen, W., and Richman, J. Learning to match and cluster entity names. In *Proc. of SIGKDD-02*.

Cohen, W.; Ravikumar, P.; and Fienberg, S. 2003. A comparison of string metrics for name-matching tasks. In *IIWeb Workshop 2003*.

Cohen, W. 1998. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proc. of SIGMOD-98*.

Doan, A.; Madhavan, J.; Domingos, P.; and Halevy, A. 2002. Learning to map ontologies on the Semantic Web. In *Proc. of WWW-02*.

Doan, A.; Lu, Y.; Lee, Y.; and Han, J. 2003. Profile-based object matching for information integration. *IEEE Intelligent Systems* 18(5):54–59.

Dong, X.; Halevy, A.; Madhavan, J.; and Nemes, S. 2005. Reference reconciliation in complex information spaces. In *Proc. of SIGMOD-05*.

Freitag, D. 1998. Multistrategy learning for information extraction. In *Proc. of ICML-98*.

Hernandez, M., and Stolfo, S. 1995. The merge/purge problem for large databases. In *Proc. of SIGMOD-95*.

Li, X.; Morie, P.; and Roth, D. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *Proc. of AAAI-04*.

McCallum, A.; Nigam, K.; and Ungar, L. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. of SIGKDD-00*.

Parag, and Domingos, P. 2004. Multi-relational record linkage. In *Proc. of the KDD Workshop on Multi-Relational Data Mining*.

Pasula, H.; Marthi, B.; Milch, B.; Russell, S.; and Shpitser, I. 2003. Identity uncertainty and citation matching. In *Proc. of NIPS-03*.

Ravikumar, P., and Cohen, W. 2004. A hierarchical graphical model for record linkage. In *Proc. of UAI-04*.

Tejada, S.; Knoblock, C.; and Minton, S. 2002. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of KDD-02*.

Wellner, B.; McCallum, A.; Peng, F.; and Hay, M. 2004. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proc. of UAI-04*.