

Semantic Integration in Text

From Ambiguous Names to Identifiable Entities

Xin Li, Paul Morie, and Dan Roth

■ Semantic integration focuses on discovering, representing, and manipulating correspondences between entities in disparate data sources. The topic has been widely studied in the context of structured data, with problems being considered including ontology and schema matching, matching relational tuples, and reconciling inconsistent data values. In recent years, however, semantic integration over text has also received increasing attention. This article studies a key challenge in semantic integration over text: identifying whether different mentions of real-world entities, such as “JFK” and “John Kennedy,” within and across natural language text documents, actually represent the same concept.

We present a machine-learning study of this problem. The first approach is a discriminative approach—a pairwise local classifier is trained in a supervised way to determine whether two given mentions represent the same real-world entity. This is followed, potentially, by a global clustering algorithm that uses the classifier as its similarity metric. Our second approach is a global generative model, at the heart of which is a view on how documents are generated and how names (of different entity types) are “sprinkled” into them. In its most general form, our model assumes (1) a joint distribution over entities (for example, a document that mentions “President Kennedy” is more likely to mention “Oswald” or “White House” than “Roger Clemens”), and (2) an “author” model that assumes that at least one mention of an entity in a document is easily identifiable and then generates other mentions via (3) an “appearance” model that governs how mentions are transformed from the “representative” mention. We show that both approaches perform very accurately, in the range of 90–95 percent. F_1 measure for different entity types, much better than previous approaches to some aspects of this problem. Finally, we discuss

how our solution for mention matching in text can be potentially applied to matching relational tuples, as well as to linking entities across databases and text.

Sharing data across disparate sources provides a foundation for intelligent and efficient access to heterogeneous information. Such sharing, however, raises many difficult semantic-integration challenges. The AI and the database communities have been actively working on these challenges, focusing largely on *structured data sources* (Rahm and Bernstein 2001; Ouksel and Seth 1999). For example, significant research has been done on the problems of ontology and schema matching and on matching relational tuples (see the other articles in this issue of *AI Magazine*, as well as a forthcoming *SIGMOD Record* special issue on semantic integration, edited by Doan, Noy, and Halevy; Madhavan, Bernstein, and Rahm [2001]; Doan, Domingos, and Halevy [2001]; Dhamankar et al. [2004]; Hernandez and Stolfo [1995]; Tejada, Knoblock, and Minton [2002]; Andritsos, Miller, and Tsaparas [2004]; Doan et al. [2003]).

In the past several years, however, semantic-integration issues over natural language text (such as news articles) and semistructured text and web data (such as e-mail, hypertext markup language [HTML] pages, and seminar announcements) have attracted increasing attention. Key challenges here include finding relevant text fragments, identifying whether different fragments refer to the same real-world

entity, and, if so, merging them to obtain more information about the entity.

Longstanding applications that require semantic integration over text include information extraction (Roth and Yih 2001) and question-answering systems (Voorhees 2002; Roth et al. 2002; Moldovan et al. 2002) from natural language texts. Recent applications include, for example, Citeseer, the well-known research paper index. To construct the citation graph, given any two paper citations in text format, Citeseer must decide whether they refer to the same real-world paper. Another key challenge in constructing a Citeseer-style application is to decide whether two mentions of author names in text (for example, “Michael Jordan” and “M. Jordan”) refer to the same real-world author. Another emerging application area is that of personal information management (PIM) (Jones and Maier 2004; Dong et al. 2004). In PIM, disparate data sources are merged to construct a network of interrelated entities (for example, persons, projects, papers, and so on) so that the data owner can easily browse and retrieve information about any particular entity. Other applications include acquiring social network information and performing other name-related disambiguation tasks (Pasula et al. 2002; Cohen, Ravikumar, and Fienberg 2003; Bilenko and Mooney 2003; Mann and Yarowsky 2003; Gooi and Allan 2004; Culotta, Bekkerman, and McCallum 2004).

As mentioned previously, the key challenge in semantic integration over text is to automatically identify concepts in text and link textual segments representing “mentions” of concepts to real-world objects. This problem is difficult because most names of people, locations, organizations, events, and others entities have multiple writings that are being used freely within and across documents. Consider, for example, a user that attempts to acquire a concise answer, “on May 29, 1917,” to the question “When was President Kennedy born?” by accessing a large collection of textual articles. The sentence, and even the document that contains the answer, may not contain the name “President Kennedy”; it may refer to this entity as “Kennedy,” “JFK,” or “John Fitzgerald Kennedy.” Other documents may state that “John F. Kennedy Jr. was born on November 25, 1960,” but this fact refers to our target entity’s son. Other mentions, such as “Senator Kennedy” or “Mrs. Kennedy” are even “closer” to the writing of the target entity, but clearly refer to different entities. Even the statement “John Kennedy, born 5-29-1941” turns out to refer to a different entity, as one can tell by observing that the document discusses Kennedy’s batting statistics. A

similar problem exists for other entity types, such as locations and organizations.

In this article, we present a solution to the mention-matching problem in natural language texts, which we called the *robust reading* problem (Li, Morie, and Roth 2004a), and then discuss its potential application to record matching and linking entities across text and databases. Two conceptually different machine-learning approaches are presented (Li, Morie, and Roth 2004b) and compared to existing approaches. We conclude that an unsupervised learning approach can be applied very successfully to this problem, provided that it is used along with strong but realistic assumptions on the usage of names in documents.

Our first model is a discriminative approach that models the problem as that of deciding whether any two names mentioned in a collection of documents represent the same entity. This straightforward modeling of the problem results in a classification problem—as has been done by several other authors (Cohen, Ravikumar, and Fienberg 2003; Bilenko and Mooney 2003), allowing us to compare our results with these. This is a standard pairwise classification task, and a classifier for it can be trained in a supervised manner; our main contribution in this part is to show how relational (string and token level) features and structural features, representing transformations between names, can improve the performance of this classifier. Several attempts have been made in the literature to improve the results of a pairwise classifier of this sort by performing some global clustering, with the pairwise classifier as a similarity metric. The results of these attempts were not conclusive, and we provide some explanation for it. First, we show that, in general, a clustering algorithm used in this situation may in fact hurt the results achieved by the pairwise classifier. Then, we argue that using a locally trained pairwise classifier as a similarity metric might be the wrong choice for this problem. Our experiments concur with this. However, as we show, splitting data in some coherent way—for example, into groups of documents originated at about the same time period—prevents some of these problems and aids clustering significantly.

This observation motivates our second model, which better exploits structural and global assumptions. The global probabilistic model for mention matching is detailed in the paper by Li, Morie, and Roth (2004b). Here we briefly illustrate one of its instantiations and concentrate on its basic assumptions, the experimental study and a comparison to the discriminative model.

At the heart of our unsupervised approach is

a view of how documents are generated and how names (of different types) are “sprinkled” into them. In its most general form, our model assumes (1) a joint distribution over entities, so that a document that mentions “President Kennedy” is more likely to mention “Oswald” or “White House” than “Roger Clemens,” and (2) an “author” model that makes sure that at least one mention of a name in a document is easily identifiable (after all, that is the author’s goal) and then generates other mentions via (3) an “appearance” model that governs how mentions are transformed from the “representative” mention. Under these assumptions, the goal is to learn a model from a large corpus and use it to support mention matching and tracking. Given a collection of documents, learning proceeds in an *unsupervised* way; that is, the system is not told during training whether two mentions represent the same entity.

Both learning models assume the ability to recognize names and their type using a named entity recognizer as a preprocessor.

Our experimental results are somewhat surprising; we show that the unsupervised approach can solve the problem accurately, giving accuracies (F_1) above 90 percent, and better than our discriminative classifier (obviously, with a lot more data).

After discussing some related work, we first present the experimental methodology in our evaluation in order to present a more concrete instantiation of the problem at hand. Next, we describe the design of our pairwise name classifier, compare the design to other classifiers in the literature, and discuss clustering on top of a pairwise classifier. We then present the generative model and compare the discriminative and generative approaches along several dimensions. Finally, we discuss how our solution can potentially be applied to the record-matching problem, as well as the problem of linking entities across databases and text.

Previous Work

We discuss research related to our work from several perspectives: schema and record matching, coreference resolution, and mention matching in text.

Schema and Record Matching

The mention-matching problem is related to tuple matching and schema matching as studied in the database domain (for example, Rahm and Bernstein 2001; Hernandez and Stolfo 1995; Tejada, Knoblock, and Minton 2002; Andritsos, Miller, and Tsaparas 2004; and Doan et al. 2003). Indeed, at an abstract level, the prob-

lem of semantic integration over structured information sources, as defined in the database domain, can be phrased as follows: well-defined concepts (in the case of schema matching) and entities (in the case of tuple matching) are manifested when put into databases in multiple, ambiguous occurrences. The matching approaches attempt to identify concepts or entities from these occurrences and allow for the integration of information based on this semantic connection. This abstract definition also captures the problem of mention matching, in the context of semantic integration over text.

However, mention matching in text differs from schema and record matching in at least two important aspects. First, the information that can be used to discriminate between two names in text is not well defined. Even when one can identify (in principle) the information, it is still hard to extract it accurately from a document and place it into a well-structured tuple.

Second, textual documents contain, in principle, more information that might influence the decision of whether to merge two mentions. For example, the notion of individual documents might be very significant here. While similar mentions that occur within and across different documents may refer to different entities, and very different (surface level) mentions could refer to the same entity, these variations are typically more restricted within one document. And, learning those variations within a document may contribute to better decisions across documents. Moreover, there is more contextual information for a given mention than in a typical database tuple; this information might include the syntactic structure of the sentence as well as entities that cooccur in the same document.

Within the context of record matching, machine-learning approaches (Cohen and Richman 2002; Bilenko and Mooney 2003) usually consider a pair of records and extract from the pair features that capture their similarity. The classifier is thus a parameterized similarity function that is trained given a set of annotated examples. That is, the pairs are labeled as matching or nonmatching tags, and training serves to choose the parameters that optimize some loss function. Learning-based similarity metrics vary in their selection of features, hypotheses, and learning algorithms. These approaches can be viewed as addressing some aspects of the “appearance” model—a lower-level processing step in our approach.

Coreference Resolution

The mention-matching problem is also related to the general coreference-resolution problem

in natural language processing (Soon, Ng, and Lim 2001; Ng and Cardie 2002; Kehler 2002), which attempts to determine whether two forms of reference in a text, typically a name (more generally, a noun phrase) and a pronoun, actually refer to the same thing. This is typically done among references that appear within a relatively short distance in one document.

The problem we address here has a different goal and a much wider scope. We aim at the identification and disambiguation of real-world concepts from its multiple and ambiguous mentions both within and across documents. Our problem has broader relevance to the problem of intelligent information access and semantic integration across resources. Typically, machine-learning approaches to the coreference problem first convert the local information into a set of features and then make use of a supervised-learning approach to determine whether a given pronoun and a given noun phrase corefer. Approaches differ in the algorithm used and the features extracted. In contrast, one of our approaches is unsupervised, which does not require knowing, for training, the entity referred by any mention. The approach is based on a probabilistic model that integrates the generation process of all mentions from their original entities with the notion of individual documents. Given the generative model, model estimation and mention matching can be solved simultaneously through an iterative optimizing process.

Mention Matching in Text

Several works have addressed some aspects of the mention-matching problem with text data and studied it in a cross-document setting (Mann and Yarowsky 2003; Bagga and Baldwin 1998; McCallum and Wellner 2003; Gooi and Allan 2004). Mann and Yarowsky (2003) consider one aspect of the problem, that of distinguishing occurrences of identical names in different documents, and for only one type of entity—people. That is, they consider the question of whether occurrences of “Jim Clark” in different documents refer to the same person. Their method makes use of “people-specific” information and may not be applied easily to other types of entities and other aspects of the mention-matching problem. Bagga and Baldwin (1998) build a cross-document system based on an existing coreference resolution tool, Camp. It extracts all the sentences containing an entity as a representation of the entity and then applies a vector space model to compute the similarity between two such representations. Clustering is used subsequently to group entities in different documents into

global coreference chains. McCallum and Wellner (2003) use a conditional model to address the problem of coreference across documents. This work takes a more global view in that it defines a conditional probability distribution over partitions of mentions, given all observed mentions. The derived pairwise classification function that determines whether two names match is learned in a supervised manner, based on a maximum entropy model. However, this model does not incorporate contextual information and cannot resolve the ambiguity at the level we expect it to.

The work by Pasula et al. (2002) considers the problem of identity uncertainty in the context of citation matching and suggests a relational probabilistic model, which is related to our relational appearance model. This work also exhibits a need to perform tuple matching in a semistructured database with various textual fields.

Experimental Methodology

In our experimental study we evaluated different models on the problem of mention matching for three entity types—people (Peop), locations (Loc) and organizations (Org). The document segments shown in figure 1 exemplify the preprocessed data given as input to the evaluation. The learning approaches were evaluated on their ability to determine whether items in a pair of entities (within or across documents) actually correspond to the same real-world entity.

We collected 8,000 names from randomly sampled 1998–2000 *New York Times* articles in the TREC corpus (Voorhees 2002). These include about 4,000 personal names,¹ 2,000 locations, and 2,000 organizations. The documents were annotated by a named entity tagger.² The annotation was verified and manually corrected if needed, and each name mention was labeled with its corresponding entity by two annotators. Tests were done by averaging over five pairs of sets, each containing 600 names, that were randomly chosen from the 8,000 names. For the discriminative approach, given a training set of 600 names (each of the five test sets corresponds to a different training set), we generated positive training examples using all coreferencing pairs of names and negative examples by randomly selecting pairs of names that do not refer to the same entity. Since most pairs of names do not corefer, to avoid excessive negative examples in training sets, we use a ratio of 10:1 between negative examples and positive examples. The probabilistic model is trained using a larger corpus.

The results in all the experiments in this ar-

Document One

The Justice Department has officially ended its inquiry into the assassinations of President John F. Kennedy and Martin Luther King Jr., finding “no persuasive evidence” to support conspiracy theories, according to department documents. The House Assassinations Committee concluded in 1978 that Kennedy was “probably” assassinated as the result of a conspiracy involving a second gunman, a finding that broke from the Warren Commission’s belief that Lee Harvey Oswald acted alone in Dallas on Nov. 22, 1963.

Document Two

David Kennedy was born in Leicester, England in 1959. . . . Kennedy coedited The New Poetry (Bloodaxe Books 1993), and is the author of New Relations: The Refashioning Of British Poetry 1980-1994 (Seren 1996).

Figure 1. Segments from Two Documents Preprocessed by Our Named Entity Tagger as Input to the Mention-Matching Classifiers.

Different types of entities are annotated with different shades of gray. As shown, similar mentions within and across documents may sometimes correspond to the same entities and sometimes to different entities.

ticle are evaluated using the same test sets, except when comparing the clustering schemes. For a comparative evaluation, the outcomes of each approach on a test set of names are converted to a classification over all possible pairs of names (including nonmatching pairs). Since most pairs are trivial negative examples, and the classification accuracy can always reach nearly 100 percent, the evaluation is done as follows. Only examples in the set M_p , those that are predicated to belong to the same entity (positive predictions), are used in the evaluation and are compared with the set M_a of examples annotated as positive. The performance of an approach is then evaluated by precision and recall, defined respectively as:

$$P = \frac{|M_p \cap M_a|}{|M_p|} \quad R = \frac{|M_p \cap M_a|}{|M_a|}$$

and summarized by

$$F_1 = \frac{2P \cdot R}{P + R}.$$

Only F_1 values are shown and compared in this article.

A Discriminative Model

A natural way to formalize the mention matching is as a pairwise classification problem: Find a function $f: N \times N \rightarrow \{0, 1\}$ that classifies two strings (representing entity mentions) in the name space N , as to whether they represent the same entity (1) or not (0). Most prior work in this line adopted fixed-string similarity metrics and created the desired function by simply thresholding the similarity between two names. Cohen, Ravikumar, and Fienberg (2003) compared experimentally a variety of string similarity metrics on the task of matching entity names and found that, overall, the best-performing method is a hybrid scheme (SoftTFIDF) combining a TFIDF weighting scheme of tokens with the JaroWinkler string-distance scheme. Although this is a fixed scheme, the threshold used by the SoftTFIDF classifier is trained. Bilenko and Mooney (2003) proposed a learning framework (Marlin) for improving entity matching using trainable measures of textual similarity. They compared a learned edit distance measure and a learned vector space-based measure that employs a

Honorific Equal	Active if both tokens are honorifics and identical. An honorific is a title such as “Mr.,” “Mrs.,” “President,” “Senator,” or “Professor.”
Honorific Equivalence	Active if both tokens are honorifics, not identical, but equivalent (“Professor,” “Prof.”).
Honorific Mismatch	Active for different honorifics.
Equality	Active if both tokens are identical.
Case-Insensitive Equal	Active if the tokens are case-insensitive equal.
Nickname	Active if tokens have a “nickname” relation (“Thomas” and “Tom”).
Prefix Equality	Active if the prefixes of both tokens are equal. The prefix is defined as a substring starting at the beginning of a token and going until the first vowel (or the second if the first letter is a vowel).
Substring	Active if one of the tokens is a substring of the other.
Abbreviation	Active if one of the tokens is an abbreviation of the other; for example, “Corporation” and “Corp.”
Prefix Edit Distance	Active if the prefixes of both tokens have an edit distance of 1.
Edit Distance	Active if the tokens have an edit distance of 1.
Initial	Active if one of the tokens is an initial of another; that is, “Paul” and “P.”
Symbol Map	Active if one token is a symbolic representative of the other (“and” and “&”).

Table 1. Thirteen Types of Token-Based Features.

classifier (SVM) against fixed-distance measures (but not the one mentioned above) and showed some improvements in performance.

We propose a learning approach, LMR,³ that focuses on representing a given pair of names using a collection of relational (string and token level) and structural features. Over these we learn a linear classifier for each entity type using the sparse network of winnows (SNoW) (Carlson et al. 1999) learning architecture. A feature extractor⁴ automatically extracts features in a data-driven way for each pair of names. Our decision is thus of the form:

$$\begin{aligned}
 & f(n_1, n_2) \\
 &= \arg \max_{c \in \{0,1\}} f^c(n_1, n_2) \\
 &= \arg \max_{c \in \{0,1\}} \sum_{i=1}^k w_{i,c} f_i
 \end{aligned} \tag{1}$$

where $w_{i,c}$ is the weight of feature f_i ($1 \leq i \leq k$) in the function $f^c(n_1, n_2)$.

Feature Extraction

An example is generated by extracting features from a pair of names. Two types of features are used: *relational features*, representing mappings between tokens in the two names, and *structural features*, representing the structural transformations of tokens in one name into tokens of the other.

Each name is modeled as a partition in a bipartite graph, with each token in that name as a vertex (see figure 2), and there is a solid directed edge between two tokens (from the vertex in the smaller partition to the vertex in the larger one), which activates a token-based feature for the two names. At most one token-based relational feature is extracted for each

edge in the graph, by traversing a prioritized list of feature types until a feature is activated; if no active features are found, the feature extractor goes to the next pair of tokens. This scheme guarantees that only the most important (expressive) feature is activated for each pair of tokens. An additional constraint is that each token in the smaller partition can activate only one feature. Table 1 defines 13 types of token-based features, shown in the priority order as described previously.

Relational features are not sufficient, since a nonmatching pair of names could activate exactly the same set of features as a matching pair. Consider, for example, two names that are all the same except that one has an additional token. Our structural features were designed to distinguish between these cases. These features encode information on the relative order of tokens between the two names, by recording the location of the participating tokens in the partition. For example, for the pairs (“John Kennedy,” “John Kennedy”) and (“John Kennedy,” “John Kennedy Davis”), the active relational features are identical; but, the first pair activates the structural features “(1, 2)” and “(1, 2),” while the second pair activates “(1, 3)” and “(1, 2, 0).”

Experimental Comparison

Figure 3 presents the average F_1 for three different pairwise classifiers on the five test sets described in the experimental methodology section. The LMR classifier outperforms the SoftTFIDF classifier and the Marlin classifier when trained and tested on the same data sets.

Figure 4 shows the contribution of different feature types to the performance of the LMR classifier. The *baseline* classifier in this experiment makes use only of string-edit-distance features and “Equality” features. The *token-based* classifier uses all relational token-based features, while the *structural* classifier uses, in addition, the structural features. Adding relational and structural feature types is very significant, and more so to the *people* entity type due to a larger number of overlapping tokens between entities.

Does Clustering Help?

There is a long-held intuition that the performance of a pairwise classifier can be improved if it is used as a similarity metric and a global clustering is performed on top of it. Several works (Cohen, Ravikumar, and Fienberg 2003; Cohen and Richman 2002; McCallum and Wellner 2003) have thus applied clustering in similar tasks, using their pairwise classifiers as the metric. However, we show here that this

may not be the case; we provide theoretical arguments as well as experimental evidence that show that global clustering applied on the pairwise classifier might in fact degrade its performance. Specifically, we show that while optimal clustering always helps to reduce the error of a pairwise classifier when there are two clusters (corresponding to two entities), in general, for $k > 2$ classes, this is not the case. We sketch these arguments below.

A typical clustering algorithm views data points $n \in N$ to be clustered as feature vectors $n = (f_1, f_2, \dots, f_d)$ in a d -dimensional feature space. A data point n is in one of k classes $C = \{C_1, C_2, \dots, C_k\}$. From a generative perspective, the observed data points $D = \{n_1, n_2, \dots, n_t\}$ are sampled independently, and identically distributed from a joint probability distribution P defined over tuples $(n, c) \in N \times C$. (P is a mixture of k models.) A distance metric $\text{dist}(n_1, n_2)$ is used to quantify the distance between two points. Clustering algorithms differ in how they approximate the hidden generative models given D .

In the following definitions, we assume that $P = P_{N \times C}$ is a distribution over $N \times C$ and that $(n_1, c_1), (n_2, c_2) \in N \times C$ are sampled independently, and identically distributed according to it, with n_1, n_2 observed and c_1, c_2 hidden.

Definition 1

The problem of mention matching is that of finding a function $f: N \times N \rightarrow \{0, 1\}$ that satisfies:

$$f(n_1, n_2) = 1 \text{ iff } c_1 = c_2 \quad (2)$$

Definition 2

Let $\text{dist}: N \times N \rightarrow \mathbb{R}$ be a distance metric, and $T \geq 0$ be a constant threshold. The pairwise classification function f_p in this setting is defined by:

$$f_p(n_1, n_2) = 1 \text{ iff } \text{dist}(n_1, n_2) \leq T \quad (3)$$

The clustering-based decision f_c is defined by:

$$f_c(n_1, n_2) = 1 \text{ iff } \arg\max_i \Pr\{n_1 | C_i\} = \arg\max_i \Pr\{n_2 | C_i\} \quad (4)$$

Definition 3

Define $I(n_1, n_2)$ to be 1 when $c_1 = c_2$ and 0 otherwise. The error rate of the function $f: N \times N \rightarrow \{0, 1\}$ is defined as:

$$\text{err}(f) = E(\Pr\{f(n_1, n_2) \neq I(n_1, n_2)\}) \quad (5)$$

where the expectation is taken over independent samples, according to $P_{N \times C}$, of pairs of points $\{(n_1, c_1), (n_2, c_2)\}$.

Theorem 1 (Proof Omitted)

Assume data is generated according to a uniform mixture of k Gaussians $G = \{g_1, g_2, \dots, g_k\}$ with the same covariance matrix. Namely, a data point is generated by first choosing one of k models with probability $p_k = 1/k$, and then sam-

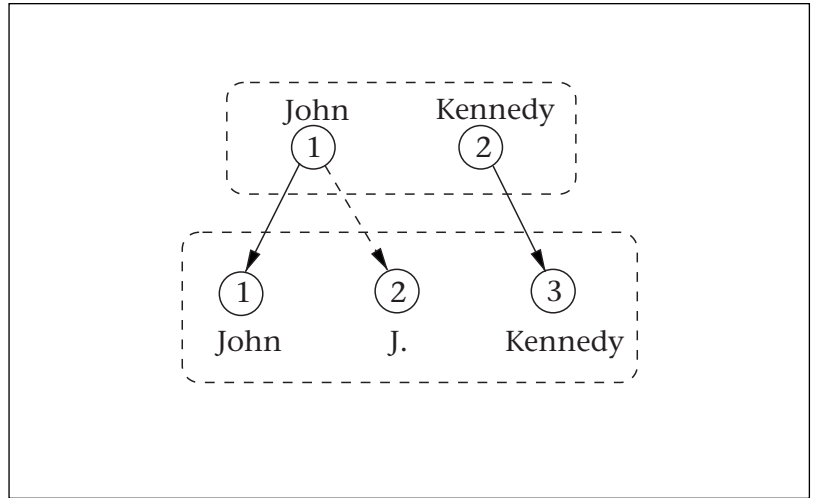


Figure 2. There are Two Possible Features (Equality and Initial) for Token 1 in the Smaller Partition, but Only the Higher Priority Equality Feature Is Activated.

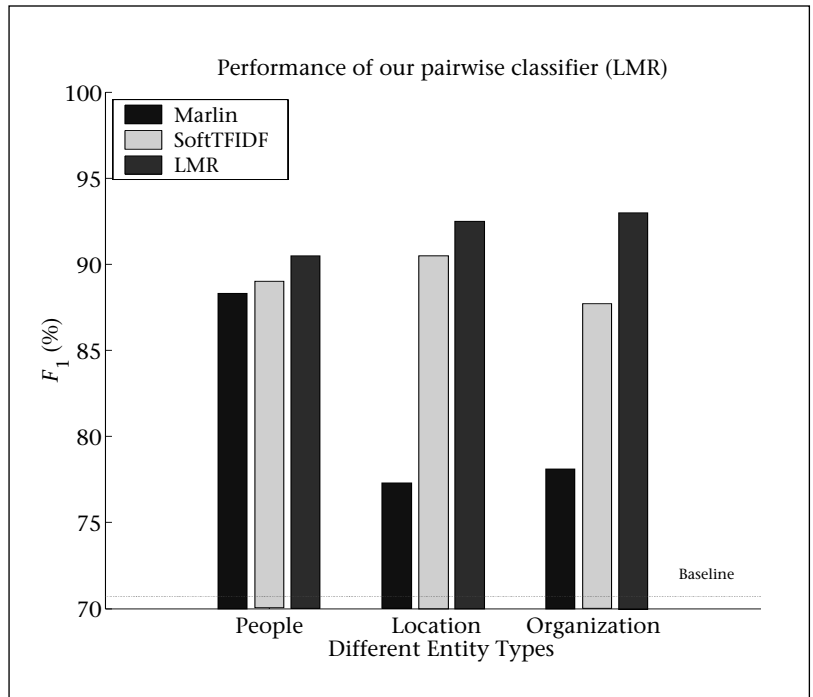


Figure 3. Performance of Different Pairwise Classifiers.

Results are evaluated using the F_1 value and are averaged over five test sets of 600 names each, for each entity type. Our LMR classifier is compared with Marlin (Bilenko and Mooney 2003) and SoftTFIDF (Cohen, Ravikumar, and Fienberg 2003). The learned classifiers are trained using corresponding training sets with 600 names. The baseline performance in the experiment is 70.7 percent given by a classifier that predicts only identical names as positive examples, and it is averaged over the three entity types.

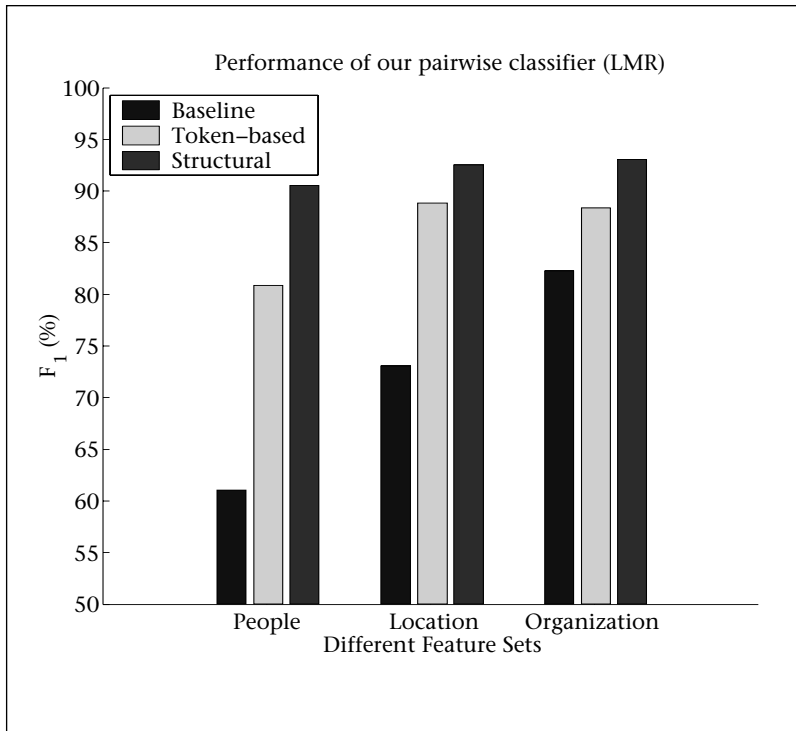


Figure 4. The Contribution of Different Feature Sets.

The LMR classifier is trained with different feature sets using the five training sets. Results are evaluated using the F_1 value and are averaged over the five test sets for each entity type with 600 names in each of them.

pling according to the i th Gaussian chosen. Suppose further that the clustering algorithm yields the correct k Gaussian distributions; then, $\forall$ threshold $T > 0$, if $k = 2$ then

$$\text{err}(f_c) < \text{err}(f_p). \quad (6)$$

However, this doesn't hold in general for $k > 2$.

For $k > 2$, we can show cases where $\text{err}(f_p) < \text{err}(f_c)$. Specifically, this holds when the centers of the Gaussians are close, and $T \rightarrow 0$.

To study this issue experimentally, we designed and compared several clustering schemes for the mention-matching task. These clustering approaches are designed based on the learned pairwise classifier LMR. Given the activation values of the classifier—the values output by the linear functions for the classes, we define a similarity metric (instead of a distance metric) as follows: Let p, n be the activation values for class 1 and class 0, respectively, for two names n_1 and n_2 ; then,

$$\text{sim}(n_1, n_2) = \frac{e^p}{e^p + e^n}.$$

In our direct clustering approach, we cluster names from a collection of documents with regard to the entities they refer to. That is, entities are viewed as the hidden classes that gen-

erate the observed named entities in text. We have experimented with several clustering algorithms and show here the best performing one, a standard agglomerative clustering algorithm based on completelink.⁵ The basic idea of this algorithm is as follows: it first constructs a cluster for each name in the initial step. In the following iterations, these small clusters are merged together step by step until some condition is satisfied (for example, if there are only k clusters left). The two clusters with the maximum average similarity between their elements are merged in each step. The evaluation, presented in figure 5, shows degradation in the results relative to pairwise classification.

Although, as we show, clustering does not help when applied directly, we attempted to see if clustering can be helped by exploiting some structural properties of the domain. We split the set of documents into three groups, each containing documents from the same time period. We then cluster first names belonging to each group and then choose a representative for the names in each cluster and, hierarchically, cluster these representatives across groups into final clusters. The completelink algorithm is applied again in each of the clustering stages. In this case (Hier (Date)—hierarchically clustering according to dates), the results are better than in direct clustering. We also performed a control experiment (Hier (Random)), in which we split the document set randomly into three sets of the same size; the deterioration in the results in this case indicates that the gain was due to exploiting the structure. The data set used here was slightly different from the one used in other experiments. It was created by randomly selecting names from documents of the years 1998–2000, 600 names from each year and for each entity type. The 1,800 names for each entity type were randomly split into equal training and test sets. We trained the LMR pairwise classifier for each entity type using the corresponding labeled training set and clustered the test set with LMR as a similarity metric.

One reason for the lack of gain from clustering is the fact that the pairwise classification function learned here is local—without using any information except for the names themselves—and thus suffers from noise. This is because, in training, each pair of names is annotated with regard to the entities to which they refer rather than their similarity in writing. Specifically, identical names might be labeled as negative examples, since they correspond to different entities, and vice versa. Our conclusion, reinforced by the slight improvement we got when we started to exploit structure in the

hierarchical clustering experiment, is that the mention-matching problem necessitates better exploitation of global and structural aspects of data. Our next model was design to address this issue.

A Generative Model for Mention Matching

Motivated by the above observation, we describe next a generative model for mention matching, designed to exploit the structure of documents and assumptions on how they are generated. The details of this model are described in the paper by Li, Morie, and Roth (2004b). Here we briefly describe one instantiation of the model (Model II there), focusing on discussing the key assumptions and their advantages and on its experimental study and a comparison to the discriminative model.

We define a probability distribution over documents $d = \{E^d, R^d, M^d\}$ by describing how documents are being generated. In its most general form the model has the following three components: (1) A joint probability distribution $P(E^d)$ governs how entities (of different types) are distributed into a document and reflects their cooccurrence dependencies. (When initializing the model described here, we assume that entities are chosen independently of each other.) (2) An “author” model makes sure that at least one mention of an entity in a document—a representative r^d —is easily identifiable and that other mentions are generated from it. The number of entities in a document, $size(E^d)$, and the number of mentions of each entity, $size(M_i^d)$, are assumed in the current evaluation to be distributed uniformly over a small plausible range so that they can be ignored in later inference. (3) The *appearance probability* of a name generated (transformed) from its representative is modeled as a product distribution over relational transformations of attribute values. This model captures the similarity between appearances of two names. In the current evaluation the same appearance model is used to calculate both the probability $P(r|e)$ that generates a representative r given an entity e and the probability $P(m|r)$ that generates a mention m given a representative r . Attribute transformations are relational in the sense that the distribution is over transformation types and independent of the specific names.

Figure 6 depicts the generation process of a document d . Thus, in order to generate a document d , after picking $size(E^d)$ and $\{size(M_1^d), size(M_2^d), \dots\}$, each entity e_i^d is selected into d independently of others, according to $P(e_i^d)$. Next,

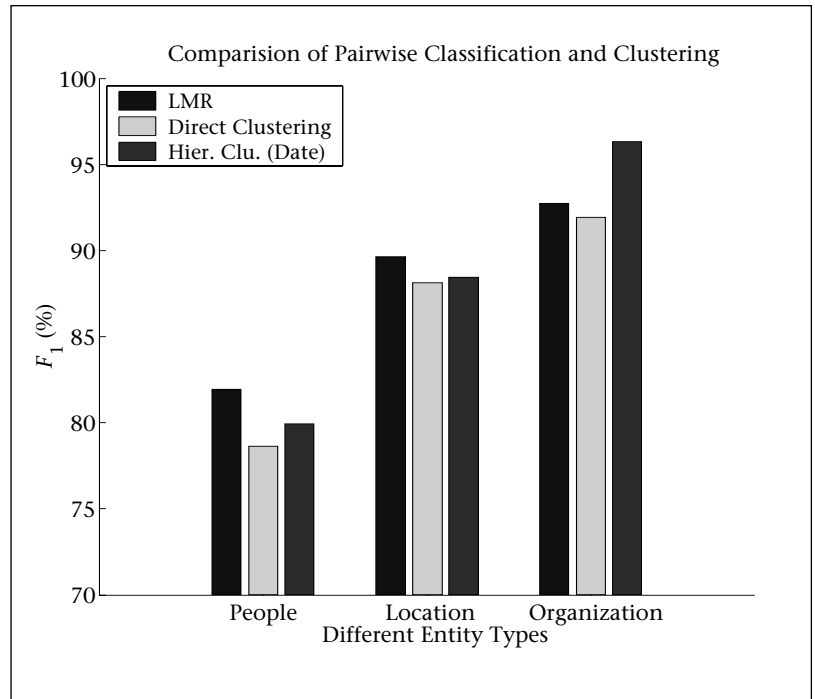


Figure 5. Best Performance of Different Clustering Approaches.

Various parameter settings, including different numbers of clusters were experimented with in direct clustering and the hierarchical clustering. LMR represents our pairwise classifier. It is compared with different clustering schemes, based on it as a similarity metric. Results are evaluated using F_1 values. The test set has 900 names for each entity type.

the representative r_i^d for each entity e_i^d is selected according to $P(r_i^d | e_i^d)$, and for each representative the actual mentions are selected independently according to $P(m_j^d | r_i^d)$. Assuming independence between M^d and E^d given R^d , the probability distribution over documents is therefore that shown in figure 7 after we ignore the size components.

Given a mention m in a document d (M^d is the set of observed mentions in d), the key inference problem is to determine the most likely entity e_m^* that corresponds to it. This is done by computing:

$$E^d = \operatorname{argmax}_{E' \subseteq E} P(E', R^d, M^d | \theta) \quad (7)$$

where θ is the learned model's parameters. This gives the assignment of the most likely entity e_m^* for m . And the probability of the document collection D is

$$P(D) = \prod_{d \in D} P(d).$$

Learning the Models

Confined by the labor of annotating data, we learn the probabilistic models in an unsupervised way given a collection of documents; that is, the system is not told during training whether two mentions represent the same en-

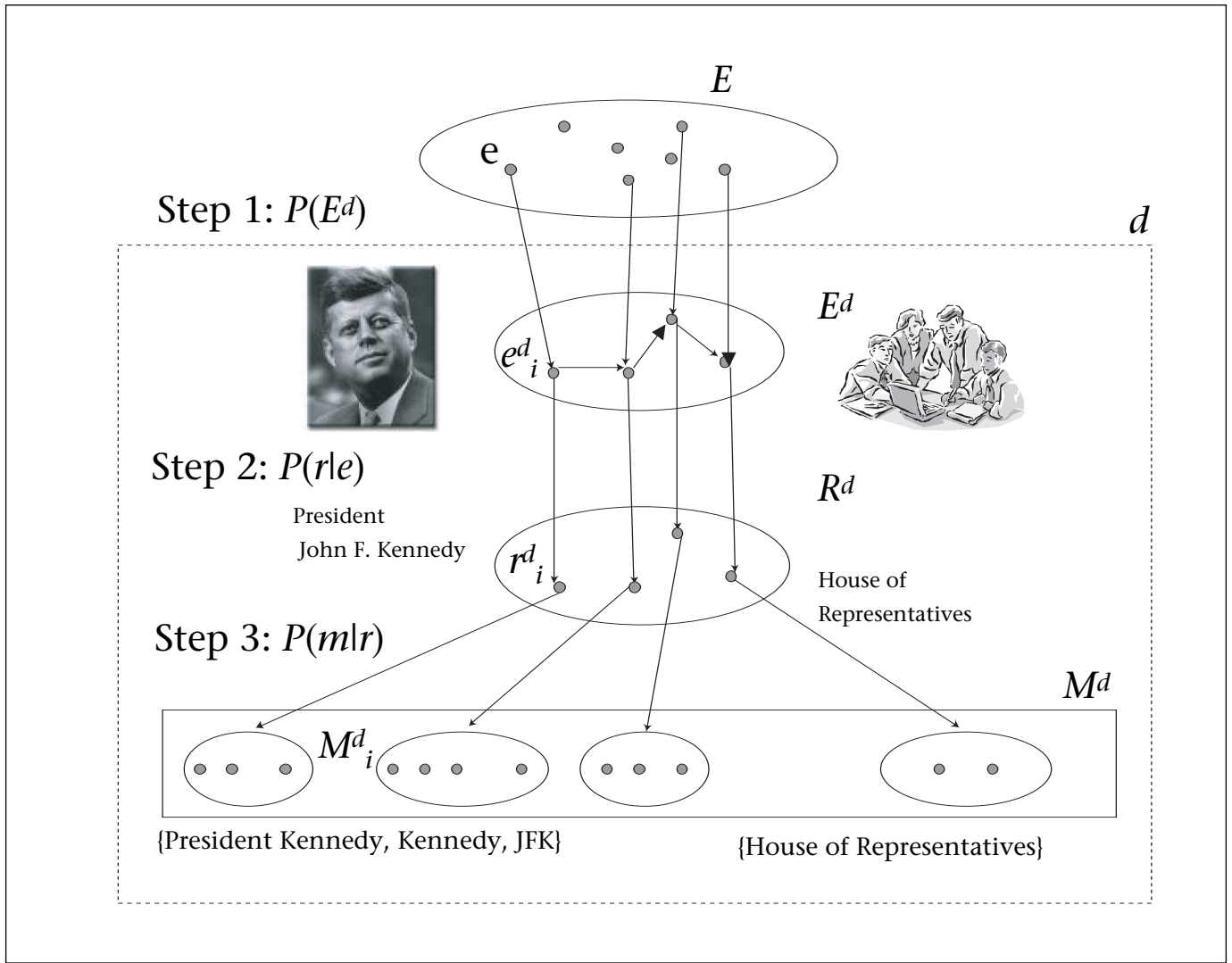


Figure 6. Generating a Document.

Real-world entities are denoted by pictures. Mentions in a document d are generated in three steps according to underlying probabilities.

$$P(d) = P(E^d, R^d, M^d) = P(E^d)P(R^d|E^d)P(M^d|R^d)$$

$$\sim \prod_{i=1}^{|E^d|} \left[P(e_i^d)P(r_i^d|e_i^d) \right] \prod_{(r_j^d, m_j^d)} P(m_j^d|r_j^d)$$

Figure 7. Probability Distribution over Documents.

tity. A greedy search algorithm modified from the standard EM algorithm (we call it the truncated EM algorithm) is adopted here to avoid complex computation.

Given a set of documents D to be studied and the observed mentions M^d in each document, this algorithm iteratively updates the model parameter θ (several underlying probabilistic distributions described before) and the structure (that is, E^d and R^d) of each document d . Different from the standard EM algorithm, in the E-step, it seeks the most likely E^d and R^d for each document rather than the expected assignment.

Truncated EM Algorithm

The basic framework of the truncated EM algorithm is shown in figure 8. It usually takes 310

1. In the initial (I-) step, an initial (E_0^d, R_0^d) is assigned to each document d by an initialization algorithm. After this step, we can assume that the documents are annotated with $D_0 = \{(E_0^d, R_0^d, M_d)\}$.
2. In the M-step, we seek the model parameter θ_{t+1} that maximizes $P(D_t | \theta)$. Given the “labels” supplied in the previous I- or E-step, this amounts to the maximum likelihood estimation (to be described later in this section).
3. In the E-step, we seek (E_{t+1}^d, R_{t+1}^d) for each document d that maximizes $P(D_{t+1} | \theta_{t+1})$ where $D_{t+1} = \{(E_{t+1}^d, R_{t+1}^d, M_d)\}$. It's the same inference problem as in equation 7.
4. Stopping Criterion: If no increase is achieved over $P(D_t | \theta_t)$, the algorithm exits. Otherwise the algorithm will iterate over the M-step and the E-step.

Figure 8. Basic Framework of the Truncated EM Algorithm.

iterations before the algorithms stop in our experiments.

Initialization

The purpose of the initial step is to acquire an initial guess of document structures and the set of entities E in a closed collection of documents D . The hope is to find all entities without loss, so duplicate entities are allowed. For all the models, we use the same algorithm. A local clustering is performed to group mentions inside each document: simple heuristics are applied to calculating the initial similarity between mentions,⁶ and pairs of mentions with similarity above a threshold are then clustered together. The first mention in each group is chosen as the representative, and an entity having the same writing with the representative is created for each cluster. The goal we try to achieve with this simple initialization is high precision, even if the recall is low.⁷ For all the models, the set of entities created in different documents becomes the global entity set E in the following M- and E-steps.

Estimating the Model Parameters

In the learning process, assuming documents have already been annotated $D = \{(e, r, m)\}_1^n$ from the previous I- or E-step, several underlying probability distributions of the relaxed models are estimated by maximum likelihood estimation in each M-step. The model parameters include a set of prior probabilities for entities P_E and the appearance probabilities $P_{W|W}$ of each name in the name space W being transformed from another.

The prior distribution P_E is modeled as a multinomial distribution. Given a set of labeled entity-mention pairs $\{(e, m)\}_1^n$,

$$P(e) = \frac{\text{freq}(e)}{n}$$

where $\text{freq}(e)$ denotes the number of pairs containing entity e .

Appearance probability, the probability of one name being transformed from another, denot-

ed as $P(n_2 | n_1)$ ($n_1, n_2 \in W$), is modeled as a product of the transformation probabilities over attribute values. The transformation probability for each attribute is further modeled as a multinomial distribution over a set of predetermined transformation types: $TT = \{\text{copy}, \text{missing}, \text{typical}, \text{nontypical}\}$.⁸

Suppose $n_1 = (a_1 = v_1, a_2 = v_2, \dots, a_p = v_p)$ and $n_2 = (a_1 = v'_1, a_2 = v'_2, \dots, a_p = v'_p)$ are two names belonging to the same entity type; the transformation probabilities P_{MR} , P_{RE} , and P_{ME} are all modeled as a product distribution (naive Bayes) over attributes:

$$P(n_2 | n_1) = \prod_{k=1}^p P(v'_k | v_k).$$

We manually collected typical and nontypical transformations for attributes such as *titles*, *first names*, *last names*, *organizations*, and *locations* from multiple sources such as the U.S. government census and online dictionaries. For other attributes, such as *gender*, only *copy* transformation is allowed. The maximum likelihood estimation of the transformation probability $P(t, k)$ ($t \in TT$, $a_k \in A$) from annotated representative-mention pairs $\{(r, m)\}_1^n$ is:

$$P(t, k) = \frac{\text{freq}(r, m) : v_k^r \rightarrow_t v_k^m}{n} \quad (8)$$

$v_k^r \rightarrow_t v_k^m$ denotes the transformation from attribute a_k of r to that of m is of type t . Simple smoothing is performed here for unseen transformations.

A Comparison between the Models

We trained the generative model in an unsupervised way with all 8,000 names. The somewhat surprising results are shown in figure 9. The generative model outperformed the supervised classifier for the people and organizations entity types. That is, by incorporating a lot of unannotated data, the unsupervised learning could do better. To understand this better, and to compare our discriminative and generative approaches further, we addressed the following two issues: learning protocol and structural assumptions.

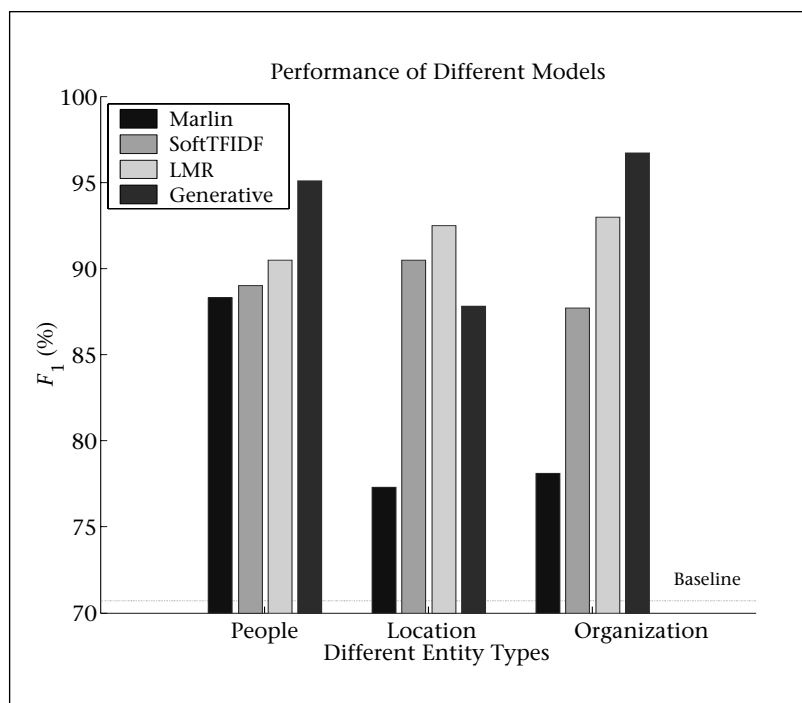


Figure 9. Discriminative and Generative Models' Results Are Evaluated by the Average F_1 Values over the Five Test Sets for Each Entity Type.

Marlin, SoftTFIDF, and LMR are the three pairwise classifiers; generative is the generative model. The baseline performance is 70.7 percent, given by a classifier that predicts only identical names as positive examples, and it is averaged over the three entity types.

F_1 (%)	LMR	Generative (All)	Generative (Unseen Data)
Peop	90.5	95.1	88.8
Loc	92.5	87.8	78.2
Org	93.0	96.7	84.2

Table 2. Results of Different Learning Protocols for the Generative Model.

This table shows the results of our supervised classifier (LMR) trained with 600 names, the generative model trained with all the 8,000 names, and the generative model trained with the part of 8,000 names not used in the corresponding test set. Results are evaluated and averaged over five test sets for each entity type.

Learning Protocol. A supervised learning approach is trained on a training corpus and tested on a different one, resulting, necessarily, in some degradation in performance. An unsupervised method learns directly on the target corpus.

This, as we show, can be significant. In a second experiment, in which we do not train the generative model on the names it will see in the test set, results clearly degrade (table 2).

Structural Assumptions. Our generative model benefits from the structural assumptions made in designing the model. For instance, the document structural information prevents the model from disambiguating mentions from different documents directly. This is done via the representatives—yielding a more robust comparison, since representatives are typically the full names of the entities. We exhibit this by evaluating a fairly weak initialization of the model and showing that, nevertheless, this results in a mention-matching model with respectable results. Table 3 shows that after initializing the model parameters with the heuristics used in the EM-like algorithm, and without further training (but with the inference of the probabilistic model), the generative model can perform reasonably well. Note, however, that the structural assumptions in the generative model did not help locations very much as shown in figure 9. The reason is that entities of this type are relatively easy to disambiguate even without structural information, since those that correspond to the same entity typically have similar mentions.

Conclusions and Future Work

While semantic integration of structured information has been widely studied, little attention has been paid to a similar problem in unstructured and semistructured data. This article describes one of the first efforts towards semantic integration in unstructured textual data, providing a promising perspective on the integration of structured databases with unstructured or semistructured information.

In this article, we present two learning approaches to the mention-matching problem—the problem of cross-document identification and tracing of names of different types, overcoming their ambiguous appearance in texts. In addition to a standard modeling of the problem as a classification task, we developed a model that aims at better exploiting the natural generation process of documents and the process of how names are “sprinkled” into them, taking into account dependencies among entities across types and an “author” model. We have shown that both models gain significantly from incorporating structural and relational information—features in the classification model; coherent data splits for clustering and the natural generation process in the probabilistic model.

In addition to further improving the discriminative and generative approaches to the mention-matching problem, there are several other critical issues we would like to address

F_1 (%)	Generative	Initial
Peop	95.1	84.1
Loc	87.8	85.0
Org	96.7	92.1

Table 3. Performance of Simple Initialization.

Generative—the generative model learned in a normal way. Initial—the parameters of the generative model initialized using some simple heuristics and used to cluster names. Results are evaluated by the average F_1 values over the five test sets for each entity type.

from the semantic-integration perspective.

The first issue is application of current approaches to record matching on structured and semistructured data, which shares many properties with the problem we addressed here. The reliable and accurate results provided by the generative model on texts suggest its potential success on structured data. Also important is the fact that our unsupervised model performs so well, since the availability of annotated data is, in many cases, a significant obstacle to good performance in semantic integration. The generative model can be applied almost directly to the problem of integrating ambiguous entity mentions across database records, by simply treating each record as a document and entity names in the record as mentions. Our model is more general in the sense that existing record-matching solutions can be incorporated to initialize the EM algorithm in the model.

The second issue is semantic integration of structured information with unstructured information. Relational databases are usually well structured and thus are able to provide users efficient and accurate access to a large collection of information. However, most available information resources are in textual form—including news articles, books, and online web pages. Lack of well-defined structure makes these resources much harder to access and exploit at the level this can be done on databases, both in terms of efficiency and accuracy. There is a need to access textual information efficiently and in-

telligently and a need to automatically build or augment databases from textual information. As more and more textual information becomes available online, this need becomes more and more urgent. A significant barrier towards the aforementioned goal is the problem of integrating mentions in database tuples and mentions in texts according to the entities they refer to. Although our current approaches can be directly applied to this problem, our target is to develop a unified solution that is competitive with both state-of-the-art record-matching and mention-matching techniques but will work better on databases if we have text, and vice versa.

Acknowledgment

We are grateful to AnHai Doan for numerous useful comments on this article. This research is supported by NSF grants IIS9801638 and ITR IIS0085836, an ONR MURI Award, and an equipment donation from AMD.

Notes

1. Honorifics and suffixes like “Jr.” are considered part of a personal name.
2. The named entity tagger was developed by the Cognitive Computation Group at UIUC. A demo of this tool is available at <http://L2R.cs.uiuc.edu/?cogcomp/eoh/ne.html>.
3. Named after the initials of the authors’ last names.
4. We use FEX, a feature extractor tool available from <http://L2R.cs.uiuc.edu/~cogcomp/cc-software.html>.
5. See Jain and Dubes (1988).
6. One example of a heuristic is: If mention n_1 is a substring of mention n_2 , then their similarity is 0.8.
7. Note that the performance of the initialization algorithm is 97.3 percent precision and 10.1 percent recall.
8. The *copy* transformation type denotes that v'_k is exactly the same as v_k ; *missing* denotes a “missing value” for v'_k ; *typical* denotes that v'_k is a typical variation of v_k , for example, “Prof.” for “Professor,” “Andy” for “Andrew”; *nontypical* denotes a nontypical transformation.

References

- Andritsos, P.; Miller, R. J.; and Tsaparas, P. 2004. Information-Theoretic Tools for Mining Database Structure from Large Data Sets. In Proceedings of the ACM SIGMOD International Conference on Management

of Data. New York: Association for Computing Machinery.

Bagga, A., and Baldwin, B. 1998. Entity-based cross-document coreferencing using the vector space model. In Proceedings of the Seventeenth International Conference on Computational Linguistics, 79–85. East Stroudsburg, PA: Association for Computational Linguistics.

Bilenko, M., and Mooney, R. 2003. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery.

Carlson, A.; Cumby, C.; Rosen, J.; and Roth, D. 1999. The SNoW Learning Architecture. Technical Report UIUCDCSR992101, Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL.

Cohen, W.; Ravikumar, P.; and Fienberg, S. 2003. A Comparison of String Metrics for Name-Matching Tasks. Paper presented at the IJCAI-03 workshop on Information integration on the Web (IIWeb-03), Acapulco, Mexico, Aug. 9–10.

Cohen, W., and Richman, J. 2002. Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery.

Culotta, A.; Bekkerman, R.; and McCallum, A. 2004. Extracting Social Networks and Contact Information from E-mail and the Web. Paper presented at the First Conference on Email and Anti-Spam (CEAS 2004), Mountain View, CA, July 30–31 (www.ceas.cc/papers-2004/).

Dhamankar, R.; Lee, Y.; Doan, A.; Halevy, A.; and Domingos, P. 2004. iMAP: Discovering Complex Matches between Database Schemas. In Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: Association for Computing Machinery.

Doan, A.; Domingos, P.; and Halevy, A. 2001. Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach. In Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: Association for Computing Machinery.

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.; Nemes, E.; Sigurdsson, S.; and Domingos, P. 2004. Semex: Toward On-the-Fly Personal Information Integration. Paper presented at the VLDB II

Workshop on Information integration on the Web, Toronto, Canada, 30 August (cips.eas.asu.edu/iiweb-proceedings.html).

Gooi, C., and Allan, J. 2004. Cross-Document Coreference on a Large Scale Corpus. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics*. Philadelphia: University of Pennsylvania (acl.ldc.upenn.edu/hlt-naacl2004/main/)

Hernandez, M., and Stolfo, S. 1995. The Merge/Purge Problem for Large Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 127–138. New York: Association for Computing Machinery.

Jain, A., and Dubes, R. 1988. *Algorithms for Clustering Data*. Engelwood Cliffs, N.J.: Prentice Hall.

Jones, W., and Maier, D. 2004. Personal Information Management. Paper presented at the Information Retrieval and Databases: Synergies and Syntheses Workshop, Seattle, Sept. 14–16 (dada.cs.washington.edu/nsf2003).

Kehler, A. 2002. *Coherence, Reference, and the Theory of Grammar*. Stanford, CA: CSLI Publications.

Li, X.; Morie, P.; and Roth, D. 2004a. Identification and Tracing of Ambiguous Names: Discriminative and Generative Approaches. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Li, X.; Morie, P.; and Roth, D. 2004b. Robust Reading: Identification and Tracing of Ambiguous Names. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics*. Philadelphia: University of Pennsylvania (acl.ldc.upenn.edu/hlt-naacl2004/main/)

Madhavan, J.; Bernstein, P.; and Rahm, E. 2001. Generic Schema Matching with Cupid. In *Proceedings of the International Conference on Very Large Databases (VLDB)*. San Francisco: Morgan Kaufmann Publishers.

Mann, G., and Yarowsky, D. 2003. Unsupervised Personal Name Disambiguation. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2003)*. San Francisco: Morgan Kaufmann Publishers.

McCallum, A., and Wellner, B. 2003. Toward Conditional Models of Identity Uncertainty with Application to Proper Noun Coreference. Paper presented at the IJCAI-03 workshop on Information integration on the Web (IIWeb-03), Acapulco, Mexico, Aug. 9–10.

Moldovan, D.; Pasca, M.; Harabagiu, S.; and Surdeanu, M. 2002. Performance Issues and

Error Analysis in an Open-Domain Question Answering System. In *Proceedings of the Fortieth Annual Meeting of the Association for Computational Linguistics*, 33–40. East Stroudsburg, PA: Association for Computational Linguistics.

Ng, V., and Cardie, C. 2002. Improving Machine Learning Approaches to Coreference Resolution. In *Proceedings of the Fortieth Annual Meeting of the Association for Computational Linguistics*. East Stroudsburg, PA: Association for Computational Linguistics.

Ouksel, A., and Seth, A. P., eds. 1999. Special Issue on Semantic Interoperability in Global Information Systems. *SIGMOD Record* 28(1).

Pasula, H.; Marthi, B.; Milch, B.; Russell, S.; and Shpitser, I. 2002. Identity Uncertainty and Citation Matching. In *Advances in Neural Information Processing (NIPS)*. Cambridge, MA: MIT Press.

Rahm, E., and Bernstein, P. 2001. On Matching Schemas Automatically. *VLDB Journal* 10(4): 334–350.

Roth, D., and Yih, W. 2001. Relational Learning Via Propositional Algorithms: An Information Extraction Case Study. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 1257–1263. San Francisco: Morgan Kaufmann Publishers.

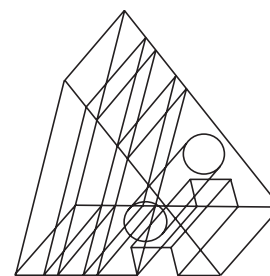
Roth, D.; Cumby, C.; Li, X.; Morie, P.; Nagarajan, R.; Rizzolo, N.; Small, K.; and Yih, W. 2002. Question Answering Via Enhanced Understanding of Questions. In *Proceedings of the Eleventh Text Retrieval Conference, NIST*, 592–601. Gaithersburg, MD: National Institute of Standards and Technology.

Soon, W.; Ng, H.; and Lim, D. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics* (Special Issue on Computational Anaphora Resolution) 27(4): 521–544.

Tejada, S.; Knoblock, C.; and Minton, S. 2002. Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification. In *Proceedings of the Eighth SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery.

Voorhees, E. 2002. Overview of the TREC2002 question answering track. In *Proceedings of the Eleventh Text Retrieval Conference, NIST*, 115–123. Gaithersburg, MD: National Institute of Standards and Technology.

Xin Li (xli1@cs.uiuc.edu) is currently a fifth-year Ph.D. student in the Cogcomp Computation group of the Department of



Reminder:
Student Scholar
Applications for AAAI-05
Are Due April 29

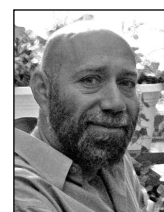


Computer Science at the University of Illinois at Urbana-Champaign. He earned a master's degree at Peking University, China. Li's research interests are in machine learning and natural language processing.



Paul Morie received his B.S. in computer engineering from the University of Illinois at Urbana-Champaign in 2003. He currently works as a research programmer in the Cognitive Computation Group at UIUC. His

e-mail address is morie@cs.uiuc.edu.



Dan Roth is an associate professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign (UIUC) and the Beckman Institute and a Willett Faculty Scholar of the College of Engineering at UIUC. He earned his B.A. summa cum laude in mathematics from the Technion, Israel, in 1982 and his Ph.D. in computer science from Harvard University in 1995. Roth's research spans both theoretical and applied work in machine learning and intelligent reasoning—focusing on learning and inference for natural language processing and intelligent access to information. He is on the editorial board of several journals in AI and natural language processing and served as the program cochair of ACL'03, the main international meeting of the Association for Computational Linguistics and the natural language processing community.