MEDIATE: Learning to Match Entity Mentions across Text and Databases

AnHai Doan[1] Xin Li[2] Dan Roth[2]

[1]University of Wisconsin-Madison [2]University of Illinois-Urbana

Abstract

Many real-world applications increasingly involve both structured data and text. A given real-world entity is often referred to in different ways, such as "Helen Hunt", and "Mrs. H. E. Hunt", both within and across the structured data and the text. Due to this *semantic heterogeneity*, it remains extremely difficult to glue together information about real-world entities from the available data sources and effectively utilize both types of information.

This paper describes the MEDIATE system which automatically matches entity mentions *within* and *across* both text and databases. The system can handle multiple types of entities (e.g., people, movies, locations), is easily extensible to new entity types, and operates with no need for annotated training data. Given a relational database and a set of text documents, MEDIATE learns from the data a *generative model* that provides a probabilistic view on how a data creator might have generated mentions, then applies it to matching the mentions. The model exploits the similarity of mention names, common transformations across mentions, and context information such as age, gender, and entity co-occurrence. To maximize matching accuracy, MEDIATE also propagates information across contexts. Experiments on real-world data show that MEDIATE significantly outperforms existing methods that address aspects of this problem, and that it can exploit text to improve record linkage, and vice versa.

1 Introduction

Many real-world applications increasingly involve a large amount of both structured data and text. The reason is two fold. First, certain kinds of information are best captured in structured data, and other kinds in text. Second, the information required for the application may need to be assembled from many sources, some of which contribute structured data, and others text. Examples of such applications arise in numerous domains, including enterprizes, government agencies, civil engineering, bioinformatics, health care, personal information management, and the World-Wide Web.

However, effectively utilizing both structured data and text in the above applications remains extremely difficult. A major reason is *semantic heterogeneity*, which refers to the variability in writing *real-world entities* in text and in structured data sources, or to using the same *mention* to refer to different entities.

Text documents naturally contain much ambiguity. For example, different movie reviews can mention actress Helen Hunt as "Helen Hunt", "Mrs. Hunt", "Helen", or even misspelled as "actress Helen Humt". In some domains the ambiguity is even worse. For example, news articles have referred to President John Kennedy as "JFK", "President Kennedy", "Representative Kennedy", etc. On the database side, different relational records often refer to the same person, but use different mentions, such as (Helen Hunt, Beverly Hills) and (H. E. Hunt, 145 Main St. Beverly Hills). Conversely, different records may use the same mention to refer to different real-world entities. For example, in the Internet Movie Database (*imdb.com*) the mention "Helen Hunt" refers to *three* different people: two actresses and a make-up artist. This problem is especially common when data is integrated from multiple databases, but arises often also in stand-alone databases, due to the nature of the data, misspelling, and errors in data entry [32, 18, 35, 9]. Finally, semantic heterogeneity is also pervasive *across* text and databases. For example, "Helen Hunt" in a relational record may refer to the same person as "Mrs. H. E. Hunt" in a text document, but not "Professor Hunt".

This paper considers the problem of resolving the above types of semantic heterogeneity, by matching mentions that refer to the same real-world entities, both *within* and *across* text and databases. This problem is more general than *record linkage* (a.k.a. record

Author names are listed alphabetically. The work was performed when the first author was at the University of Illinois.

Technical Report, Department of Computer Science University of Illinois at Urbana

matching), the well-known problem of deciding if two given relational records refer to the same real-world entity (e.g., [18, 35, 9]). It is also more general than mention matching in text, as Section 3 will discuss.

Resolving semantic heterogeneity across text and databases brings several significant benefits:

- **Entity Consolidation:** Many applications significantly benefit from being able to retrieve *all* information related to a given real-world entity, be it from text or structured data. Solving the above problem would immediately provide a solution: retrieve all mentions that belong to the given entity.
- **Improve Record Linkage:** Even when an application deals only with databases, it can still leverage text in the same domain to improve record matching, if it can link mentions across databases and text.
- **Improve Text Related Tasks:** Conversely, problems on the text side, such as information extraction, question answering, and robust reading [23], rely strongly on the ability to accurately match mentions in text. This, in turn can benefit from any available structured data.
- **Mining across Text and Databases:** The ability to link mentions can be leveraged to enable discovering groups of related entities, retrieving all entities that satisfy certain conditions and finding relationships among entities. So far, these have been limited to either on text or structured data.

Matching mentions can also enable new types of queries over the linked mentions graph, or improved information retrieval on both text and databases.

Despite significant potential benefits, as far as we know, no work has directly addressed mention matching in the context of integrating text and databases, and current solutions to related problems are not directly applicable. Solutions for record linkage are not well designed to handle the *unstructured* nature of mentions in text, and solutions for matching text mentions are not suited for exploiting the *structured* nature of databases.

In this paper we build on recent advances in both areas, and propose MEDIATE[1], a unifying solution that automatically matches mentions across text and databases. The key idea underlying MEDIATE is a generative model that specifies how entity mentions are generated within a database record or a text document. MEDIATE learns the model directly from the given data set, then applies it to predict the matches.

Specifically, we make the following contributions:

- An architecture for mentions matching in data sets that involve both text and structured data. The architecture builds on a generative probabilistic model, a commonly used approach in the AI community. The model provides a principled solution which can handle multiple types of entities, is highly extensible to new entity types, and operates without the need for expensive hand-crafted training data.
- An extension to the generative model that exploits context information in the neighborhood of the mentions as well as the co-occurrence of real-world entities, to make accurate matching decisions.
- A mechanism to transfer knowledge across contexts, to maximize matching accuracy.
- The MEDIATE system that embodies the above innovations, and a set of experiments on real-world data that illustrates the system's effectiveness. Our experiments show that MEDIATE achieves high overall matching accuracy of 77.2 - 81.7% F-1 across text and databases, that it significantly outperforms record linkage techniques on the database side, and achieves even higher accuracy with the use of text. The experiments further show that MEDIATE can exploit structured data when available to improve text mentions matching, and that it is robust to varying degrees of semantic heterogeneity.

[1]Matching Entities in Data Instances And TExt

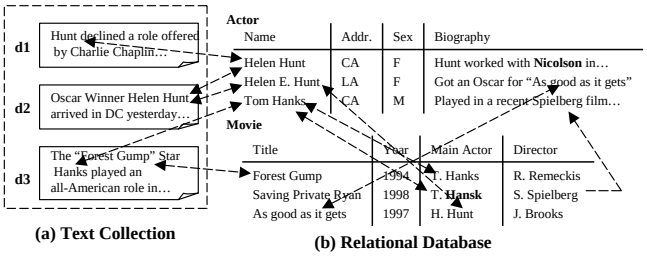


Figure 1: A simplified data set for a movie application, which contains both text and structured data. The arrows denote semantic matches that we want to establish among mentions of actors and movies.

The paper is organized as follows. The next section defines the mention matching problem. Sec. 3 reviews related work. Sec. 4 highlights the important points of our solution with examples. Sec. 5-7 describe the MEDIATE system. Sec. 8 presents experiments.

2 Problem Definition

We now describe the specific mention matching problem considered in this paper.

Data Sets, Entities, and Mentions: We assume that an application deals with a data set that consists of relational tables and text documents (but the ideas here can be generalized to other data representations). Figure 1 shows a simplified movie data set, with two tables Actor and Movie, and three news articles.

Given such a data set, we define a set of *real-world entity types* that the application is interested in. For example, the above movie application may be interested in *people* and *movies*, whereas a bibliography application such as Citeseer may be interested in *authors, papers*, and *publication venues*. Next, we assume that instances of real-world entities of the above

types are referred to using *mentions* of their names in the data set. In Figure 1, examples of such mentions are underlined: "Helen Hunt", "T. Hanks", "R. Remeckis", "Forrest Gump", etc. Note that a record may contain multiple mentions of the same entity (e.g., "Helent Hunt" and "Hunt" in the first record of Actor.

Mention discovery in text has received much attention and success in the database, AI, KDD, and WWW communities, within the context of named entity recognition, information extraction, and text segmentation (e.g., [1, 6, 13]). The developed techniques also often benefit from learning methods. Mentions in relational records are often marked up by the record boundaries (e.g., "Helen Hunt", "Helen E. Hunt", etc. in Figure 1). For those which are not (e.g., mentions in the text field comment of Table Actor), we can apply the above techniques for mention discovery in text. For these reasons, in this paper we assume that mentions are already marked up in the data set, and focus on the problem of matching them.

The Mention Matching Problem: Given the marked up mentions in a set of relational tables and text documents, our goal is to link all pairs of mentions that refer to the same real-world entities. Figure 1 shows the links that we want to establish in the above movie data set.

3 Background & Related Work

We consider related work from several perspectives.

Problem Definition: As described, the mention matching problem is more general than both record linkage and mention matching in text. Record linkage typically treats each relational tuple as a description of a *primary entity*, then tries to link tuples that describe the same entity within a single table, or across different tables. For example, given table Actor in Figure 1, it may attempt to decide if the first and second records refer to the same actress, and so on. Thus, conceptually it matches mentions that occur only in certain attributes (e.g., name of Actor). In contrast, we match *all* mentions that occur in the database. For example, in Table ACTOR we also match mentions such as "Hunt" and "Spielberg" in attribute biography of the first and second records with all other mentions in the database. Our problem therefore subsumes record linkage. We show examples in Section 4 and demonstrate empirically in Section 8 that solving mention matching also improves record linkage accuracy.

Research on mention matching in text has mostly focused on co-reference resolution [20, 28, 36] within the context of a single document, 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 real-world entity. Only recently have people started to consider mention matching across documents [24, 23, 22, 3]. Our problem is more general in that beyond matching mentions within and across text documents, we also seek to link them with those in an associated database.

Schema Matching: It is also important to emphasize that we do not consider semantic heterogeneity at the *database schema* level, a related and important problem that has received much attention [31]. Instead, we consider semantic heterogeneity at the *data* level, in the context of integrating structured data and text.

Techniques: A wealth of techniques have been developed to match mentions, with respect to both record linkage and text contexts (e.g., [37, 8, 25, 40, 5, 2, 35, 17, 18, 15, 33, 11, 32]). For record linkage, early solutions employ manually specified rules [18], while subsequent works focus on learning matching rules from training data [37, 5, 35], efficient techniques to match strings [27, 17], powerful methods to match entity names [8, 17, 9], scaling up to large number of tuples [21, 16, 19, 25, 10], matching in online contexts [7], personal information management [12], matching XML data [38], and exploiting links [4].

Several recent works have also developed generative models to match mentions. The work [30] addresses citation matching in structured contexts, a much narrower problem. It proposes a full-blown probabilistic relational model [14], and as such is harder to understand, requires a lot of data (to learn the model parameters), and has a very high runtime complexity. The model proposed in [34] for matching tuples is much more efficient, but does not capture and exploit the notion of real-world entities, as we do here. The works [23, 22] propose generative models for mention matching in text. However, none of these works have considered the addition of structured data. Furthermore, even within the context of text, they have not considered exploiting context information, co-occurrence of entities, and recursive contexts, as we do in MEDIATE.

Several recent works have employed another probabilistic framework called *conditional random fields (CRF)* to match mentions [29, 39]. In particular, [39] attempts to solve both mention discovery and matching at the same time. However, the probabilistic model of CRFs is less expressive than ours and may not be sufficient for the problem we consider here, and yet, are well known to have very high runtime complexity and are thus not scalable to realistic database domains.

Exploiting Context: Several recent works have also exploited context in mention matching [30, 2, 4, 29, 39]. [2] was among the first to articulate the idea, but exploits context only at a *syntactic* level. For example, if "X" and "Y" are linked to two occurrences of "Helen Hunt", respectively, then it may decide that "X" and "Y" are related. In contrast, we will first find out if the two occurrence of Helen Hunt refer to the same person. The works [4, 29, 39] exploit context at a higher *semantic* level (as we do here) but not within the context of generative models, and do not combine text and databases.

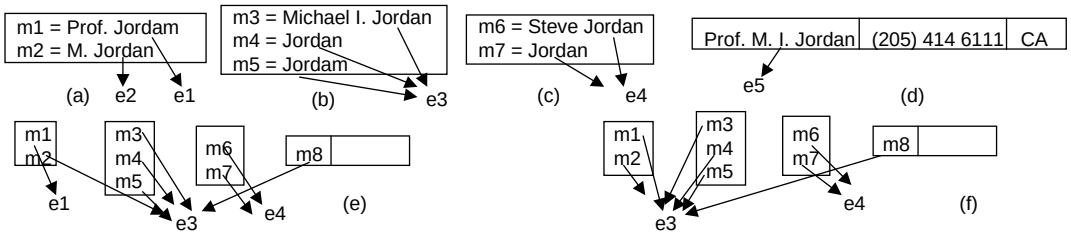


Figure 2: The generative model is constructed iteratively by assigning mentions to entities, re-learning model parameters, then re-assigning the mentions.

4 Overview of Our Approach

We now briefly describe our approach's key ideas.

Matching Mentions by Assigning to Entities: First, we convert the problem of matching pairs of mentions into two problems: (a) finding the set of real-world entities that are mentioned in the data, and (b) assigning mentions to entities. Thus, two mentions match if they are assigned to the same entity.

Developing a Generative Model: To solve the above two problems, we develop a generative model G that "explains" how mentions in the data are generated. G specifies probabilistically how to choose a set of real-world entities, create their mentions, perturb the mentions, then "sprinkle" them into the records and text documents. Thus, G provides a principled way to compute $P(m|e)$, the probability that a mention is generated from a real-world entity e.

Learning & Applying the Model: Now suppose we are given some training data (text documents and records), where we already know *all* entities, and the mentions have been assigned to the right entities. Knowing the parametrization of the generative model, we can use these assignments to learn the specific parameters of the model G. Afterward, given a new document or a record d, we apply G to assign mentions to entities. Briefly, for each mention m_i in d, we assign m_i to the entity e that maximizes $P(m_i|e)$, which is computed using the learned model G.

In practice, we do not have the annotated training data, so we learn the model using the EM algorithm, commonly employed in such situation [26]. First, we initialize the parameters of G, including the set of real-world entities, using a clustering method on the mentions in the data set. Next, we use this initial model to assign mentions to entities. Then we use the assignments to re-estimate the parameters for G and the set of entities, and so on, until the model converges.

We developed three generative models, where each builds on the previous one and exploits additional types of knowledge in the data set to improve matching accuracy. In what follows we illustrates the working of the models and the types of knowledge exploited during the matching process.

4.1 ME: Learning from Mention Names

In the first model ME we learn to match mentions using their names. Consider a simple data set of three text documents and one relational record, in the (fictional) area of "basketball research". Figures 2.a-d show the data set (only the relevant mentions are shown in text documents, to avoid clutter). To match the mentions $m_1 - m_8$, we proceed in iterations.

- *First iteration:* We cluster mentions within each document and record, using a text similarity measure. Next, we create an entity for each cluster, and assign all mentions in the cluster to the entity. Figure 2 shows the created entities $e_1 - e_5$ and the assignment of mentions. Notice that in document (a) the two mentions "Prof. Jordam" (where "Jordan" is misspelled as "Jordam") and "M. Jordan" have not been clustered together and assigned to the same entity because they are not sufficiently similar.

Next, we learn the characteristics, that is, the "profile" of each entity, based on the assigned mentions. For example, consider entity e_3. From mentions m_3 and m_5, we know that the person (corresponding to) e_3 has the first name "Michael", middle name initial "I", last name "Jordan", and that his last name could be misspelled as "Jordam".

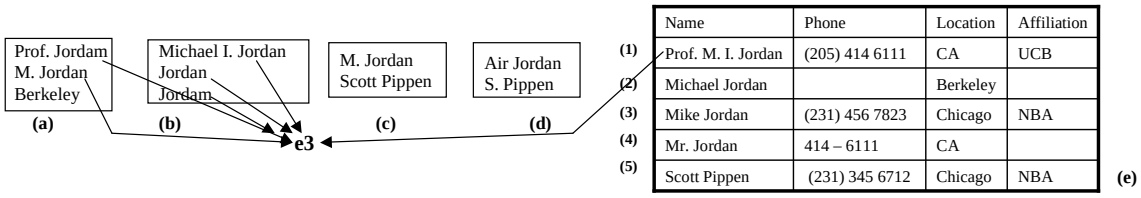
- *Second iteration:* Now given the entity profiles (i.e., the model learned in previous iteration), we reassign each mention m_i to to the best matching entity.

In our example, we end up assigning m_8 = "Prof. M. I. Jordan" (in the record) to entity e_3 because m_8 also has the middle initial "I" and share the first initial with e_3. We also assign m_2 to e_3, because m_2 = "Jordan" shares the same last name with e_3. Figure 2.e shows the reassignment. Note that entities e_2 and e_5 become empty and hence are dropped.

Now we relearn all entity profiles. Consider again person e_3. ¿From the mentions assigned to this entity, we know that, among others, his last name can be misspelled as "Jordam" and that he can have the title "Prof." (due to m_8).

- *Third iteration:* Leveraging the above profile of e_3, we can reassign m_1 = "Prof. Jordam" to e_3. Figure 2.f shows the reassignment, which also happens to be the final reassignment, as subsequent iterations do not change it.

ME then uses the above assignment to predict that mentions $m_1 - m_5$ and m_8 match, and m_6, m_7 match.



Name	Phone	Location	Affiliation
Prof. M. I. Jordan	(205) 414 6111	CA	UCB
Michael Jordan		Berkeley	
Mike Jordan	(231) 456 7823	Chicago	NBA
Mr. Jordan	414 – 6111	CA	
Scott Pippen	(231) 345 6712	Chicago	NBA

Figure 3: Examples of exploiting external attributes (e.g., phone, location), co-occurrence of entities, and transferring knowledge across matches.

The above example illustrates the iterative nature of learning our models directly from the data set. It also highlights the *global* nature of our methods, in which knowledge is transferred across matches to accumulate in entity "profiles", thus enabling more accurate matching. In contrast, a method that matches a mention pair by examining their names *in isolation* is *local* by nature, and will incorrectly match m_4, m_5, and m_7 in the above example, because their names ("Jordan" or "Jordam") are similar.

4.2 MEC: Learning from Context

The model ME exploits only characteristics of mention names, such as title, middle initial, etc. To improve matching accuracy, our second model MEC exploits context in the following two ways:

Exploiting External Attributes: Consider a slightly different data set, also in the area of "basketball research", as shown in Figure 3.

To match mentions in this data set, we begin by defining a set of *external attributes* for each person entity, such as phone, location, and affiliation in this case. Next, we apply the ME algorithm as described earlier, with some modification. In each iteration, when merging mentions to compute the profile for each entity, ME computes the values for *internal attributes*, such as title, first name, middle name, etc. Now, we also compute the values for *external attributes*. Then when reassigning mentions, we compute the probabilities $P(m|e)$ using *all* attributes, internal and external.

To illustrate, suppose using ME we have assigned all mentions in documents (a) and (b) as well as mention "Prof. M. I. Jordan" in tuple (1) to an entity e_3 (see Figure 3). After computing values for external attributes, we know that e_3 has phone = (205) 414 611 and location = CA (from tuple (1)). Then in the next iteration, we can assign mention "Mr. Jordan" in tuple (4) to e_3, because the external attributes phone and location of this mention and of e_3 share similar values. Notice that without using the external attributes, we would have incorrectly matched mention "Mr. Jordan" in tuple (4) with the first mention "Mr. Jordan" in document (c), as they share the same name.

Exploiting Entity Co-occurrence: Consider "Mr. Jordan" and "Air Jordan" in documents (c) and (d). ME would not match them, because the names are not sufficiently similar. However, consider the two associated mentions: "Scott Pippen" and "S. Pippen". If we already know that they refer to the same person, then "Mr. Jordan" and "Air. Jordan" co-occur with the same entity, and intuitively that would increase their chance to match. In MEC we develop a method to exploit such entity co-occurrence to further improve matching accuracy.

4.3 MEC^2: Knowledge Transfer via Contexts

Consider matching "Mr. Jordan" in document (c) with "Mike Jordan" in tuple (3). Our second model MEC would declare a no-match, because their names are not sufficiently similar, and they share no context information. However, suppose we know that "Scott Pippen" in document (c) matches "Scott Pippen" in tuple (5). Then since "Scott Pippen" in tuple (5) has location = Chicago and affiliation = NBA, it follows that "Scott Pippen" in document (c) also has the same location and affiliation. Since mention "Mr. Jordan" occurs close to "Scott Pippen" in document (c), it can "borrow" the context information about location and affiliation from "Scott Pippen". Armed with this, it can now match the mention "Mike Jordan" in tuple (3), since that mention also has the same location and affiliation. In MEC^2, our third and last model, we develop a method to enable such context transfer. The key challenge there is to transfer the right amount of context, with little noise.

4.4 Interplay between Text & Databases

The example in Figure 3 also shows that text can help record linkage, and vice versa. Given only the database, record linkage would have difficulty matching tuples (1) and (2), since they do not share much context. Now consider the text documents (a)-(d), and assume that our method has matched mention m = "Prof. M. I. Jordan" in tuple (1) with all person mentions in documents (a) and (b). Then we can infer that m has first name "Michael" (from document (b)) and is also associated with location Berkeley (from document (a)). This information would enable matching the two tuples (1) and (2). Similarly, we have shown in Section 4.1 that matching mentions in documents (a) and (b) is difficult unless we can bridge them via mention "Prof. M. I. Jordan" in tuple (1). This demonstrates that databases can help mention matching in text.

5 The MEDIATE Approach

We now describe our three generative models in detail.

Entities, Mentions, & Representatives: We consider matching mentions in a data set D =

$\{d_1, d_2, \ldots, d_m\}$. Each d_i is a relational record or a text document, and henceforth will be referred to as a "document". We assume D contains *mentions* (i.e. real occurrences) of $|T|$ types of real-world entities (e.g., person, movie, etc.). For each document d, we use $E_d = \{e_{di}\}$ to denote the set of entities mentioned in d, and $M_d = \{m_{di}\}$ to denote the set of mentions. For example, for entity "Tom Hanks", the corresponding set of mentions in a document may contain "Hanks", "T. Hanks" and "Actor Tom Hanks".

Among all mentions of an entity e_{di} in document d we select the one with the longest writing as the *representative* of e_{di}, and denote this mention as r_{di}. Representatives can be viewed as a typical representation of an entity, as mentioned at a specific time and place. For example, "Director Tom Hanks" and "Oscar Winner Tom Hanks" may be representatives of "Tom Hanks" in different documents. Other mentions are usually shorter and are considered to be transformed from the representative. We use $R_d = \{r_{di}\} \subseteq M_d$ to denote the set of representatives in document d.

We now can represent each document d as a collection of its entities, representatives and mentions $d = \{E_d, R_d, M_d\}$.

Internal Attributes: Let E, M, and R be the collection of all possible entities, mentions, and representatives in the world, respectively. We associate with each element in the set $W = E \cup R \cup M$ a set of *internal attributes* $A = \{a_1, \ldots, a_p\}$, which capture the most important characteristics of mention names, for matching purposes. (In the next section we define attributes *external* to names, to be used in model MEC.) For example, for person names we define title (Mr., Prof., etc.), first name, middle name, and last name. For movies we define title (e.g., "Lord of the Ring"), abbreviated title (e.g., LOTR), and sequence number (e.g., "III" in "Star War III"). Attribute values can be string or numeric, and can be missing for certain mentions.

Mention Matching: Let E_d^* be the most likely set of entities in each document d. Given a probability model with parameter θ, this set can be computed as:

$$E_d^* = argmax_{E' \subseteq E} P(d|\theta) = argmax_{E' \subseteq E} P(E', R_d, M_d|\theta). \quad (1)$$

For each mention $m \in d$, knowing E_d^* immediately gives the most likely entity e_m^* that m belongs to. As mentioned in Section 4, being able to assign mentions to entities solves the mention matching problem.

5.1 Constructing the ME Generative Model

We now describe a model to compute $P(E', R_d, M_d|\theta)$ in Equation 1. The model generates documents as follows (see Figure 4). For each document $d \in D$, we first decide its type: either a database record or a text article. Since the type of each document is known given D, we assume this step to be deterministic. Further, even though a database record or a text article could be generated differently in different domains, in the current models, we reduce the modeling complexity by generating them in the same fashion. Next, we select a number denoted as $size(E_d)$, then select a set of $size(E_d)$ entities $E_d \subseteq E$ to appear in a document d, according to a probability $P(E_d)$.

Next, for each entity $e_{di} \in E_d$, we choose a representative $r_{di} \in R$ according to a probability $P(r_{di}|e_{di})$. The set of all chosen representatives forms R_d. Then for each e_{di}, we generate a set of $size(M_{di})$ mentions (denoted as M_{di}), using representative r_{di}. Each mention $m_{dj} \in M_{di}$ is independently generated from r_{di} according to a transformation probability $P(m_{dj}|r_{di})$ (typically by "perturbing" representative r_{di}). Finally, we "sprinkle" the mentions into document d.

Example 5.1 *Figure 4.a shows the document generation process, while Figures 4.b-c show specific examples of generating a text document and a relational record. For instance, Figure 4.b shows how the basketball player Michael Jordan generates the representative "Michael Jordan", which in turn generates mentions "Michael Jordan" and "Mike" in the text document.*□

As described above, our generative model has the following four components (which forms the parameters θ): (1) a set of entities E; (2) a prior distribution $P(E)$ that governs how entities are distributed into a document d. In model ME we assume that entities are independently chosen into a document. Thus, we can compute $P(E_d) = \prod_{e_{di} \in E_d} P(e_{di})$; (3) quantities $size(E_d)$ – the number of entities in a document, and $size(M_{di})$ – the number of mentions for each entity e_{di}. In all three models (ME, MEC, and MEC^2), we make the simplifying assumption that these numbers are determined uniformly over a small plausible range; (4) the *transformation probability distribution* $P_{W|W}$ of a name being transformed from another. We use this distribution to compute $P(m|r)$ – the probability that a mention m is generated (i.e., transformed) from its representative r, as well as $P(r|e)$ – the probability that a representative r is generated from an entity e. We model $P(W|W)$ as a product distribution over relational transformations of internal attribute values (to be described shortly).

Assuming conditional independence between M_d and E_d given R_d, and ignoring the size components due to assumptions of uniform distributions, using the above model we can compute

$$\begin{aligned} P(d) &= P(E_d, R_d, M_d) = P(E_d)P(R_d|E_d)P(M_d|R_d) \\ &\approx \prod_{i=1}^{|E_d|=l_d} [P(e_{di})P(r_{di}|e_{di})] \prod_{(r_{dj}, m_{dj})} P(m_{dj}|r_{dj}). \end{aligned} \quad (2)$$

We can replace $P(d|\theta)$ in Equation 1 with this quantity, to compute E_d^*.

5.2 Learning the ME Model

To compute $P(d)$ in Equation 2, we must know the parameters θ of the generative model. If we have a set

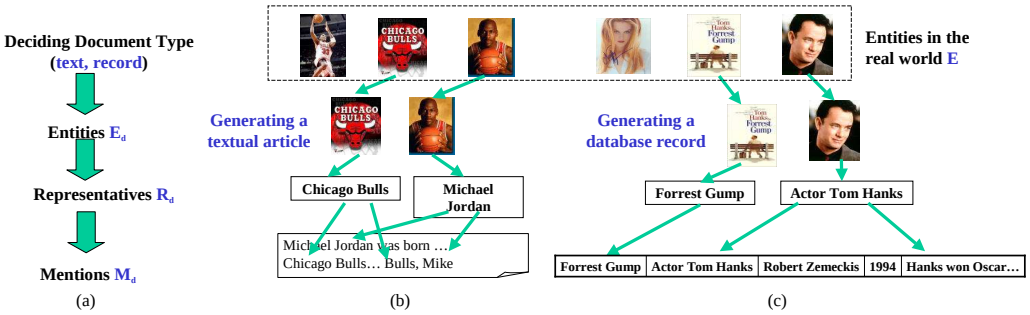


Figure 4: Generating database records and text documents.

of annotated training documents D_t, where for each document $d \in D_t$ we already manually assign each mention to the correct entity, then θ can be estimated by the common method of maximum likelihood estimation: $\theta^* \equiv argmax_\theta P(D_t|\theta)$. Specifically, the components of θ can be computed as follows:

- The set of entity E is computed as the set of all entities in the training data set D_t.
- We model the prior distribution $P(E)$ as a multinomial distribution. Suppose the training data set D_t contains n pairs of (mention,entity) assignments, then for any entity e, we compute $P(e) = freq(e)/n$, where $freq(e)$ is the number of pairs containing e.
- Computing the transformation probability distribution $P(W|W)$ reduces to computing $P(n_2|n_1)$, where $n_1, n_2 \in W$). This intuitively is the probability that name n_2 is obtained by transforming name n_1, and is modeled as a product of the transformation probabilities over internal attribute values.

The transformation probability for each internal attribute is further modeled as a multinomial distribution over a set of predetermined transformation types: $TT = \{copy, missing, typical, non-typical\}$. Type *copy* denotes v'_k is exactly the same as v_k; *missing* denotes "missing value" for v'_k; *typical* denotes v'_k is a typical variation of v_k, for example, "Prof." for "Professor", "Andy" for "Andrew"; finally, *non-typical* denotes a non-typical transformation.

Suppose n_1=(a_1=v_1,a_2=v_2,...,a_p=v_p) and n_2=(a_1=v'_1,a_2=v'_2,...,a_p=v'_p) are two names belonging to the same entity type, then $P(n_2|n_1)$ is modeled as a product distribution (naive Bayes) over internal attributes:

$$P(n_2|n_1) = \Pi_{k=1}^{p} P(v'_k|v_k). \quad (3)$$

The maximum likelihood estimation of the transformation probability $P(t,k)$ ($t \in TT, a_k \in A$) from annotated representative-mention pairs $\{(r,m)\}_1^n$ in the training data is: $P(t,k) = (freq(r,m) : v_k^r \rightarrow_t v_k^m)/n$, where $v_k^r \rightarrow_t v_k^m$ denotes the constraint that the transformation from attribute a_k of r to that of m is of type t. We perform simple smoothing for unseen transformations.

5.3 Applying the ME Model

Once the model has been learned from the training data, as described above, we can apply it to match mentions in an unseen document d, by solving Equation 1, that is, assign mentions M_d in d to the most likely representatives R_d and entities E_d.

However, solving Equation 1 directly is impractical, due to an exponential number of possible assignments of E_d and R_d to M_d. Hence we develop the following approximate algorithm. First, we compute R_d by sequentially clustering the mentions: for each mention $m \in M_d$, we compute the transformation probability $P(m|r)$ for each representative r that have already been created, then use a fixed threshold to decide whether to create a new group for m or to add it to one of the existing group with the highest $P(m|r)$ value. We then select the longest mention in each group as a representative. In the second step, we assign each representative $r_{di} \in R_d$ to the entity $e^* = argmax_{e \in E} P(e) \cdot P(r_{di}|e)$.

5.4 Learning in an Unsupervised Setting

We have described how to learn the generative model from training data. However, in practice, manually annotating data for training is very labor intensive. Hence, in this paper we propose to learn the model directly, in a unsupervised fashion. from the input data set D (recall that D is the set of relational records and text articles whose mentions we want to match). Toward this goal, we adopt the EM algorithm commonly employed for learning with unlabeled data.

We use an EM variant (called *Truncated EM*) that performs greedy search. Given the observed mentions M^d in each document $d \in D$, the algorithm iteratively updates the model parameter θ (i.e., the probability distributions and set of entities, as described earlier) and the structure (i.e., E^d and R^d) of d as follows:

We note that the above algorithm also differs from standard EM in that in the expectation step it seeks the most likely E^d and R^d for each document d rather than the expected assignment. Also, the algorithm produces a set of assignments between mentions and entities, which amounts to solving mention matching.

We have described Steps 2-3 in Sections 5.2 - 5.3. We now complete the algorithm description by dis-

Truncated EM Algorithm

(1) *(Initialization):* Assign an initial (E_0^d, R_0^d) to each document $d \in D$. The set of documents is now annotated and is denoted as $D_0 = \{(E_0^d, R_0^d, M^d)\}$.

(2) *(Maximization):* Compute model parameter θ_{t+1} that maximizes $P(D_t|\theta)$. Given the annotations supplied in the previous initialization or expectation step (see below), this is the maximum likelihood estimation described in Section 5.2.

(3) *(Expectation:)* For each document $d \in D$, compute (E_{t+1}^d, R_{t+1}^d) that maximizes $P(D_{t+1}|\theta_{t+1})$ where $D_{t+1} = \{(E_{t+1}^d, R_{t+1}^d, M^d)\}$. This is the procedure of applying the generative model described in Section 5.3.

(4) *(Convergence:)* Exit if no increase is achieved for $P(D_t|\theta_t)$, otherwise repeat Steps 2-3.

Figure 5: Sketch of the Truncated EM Algorithm

cussing Step 1: initialization. In this step, to compute (E_0^d, R_0^d), we perform clustering to group mentions within the document d. We treat each mention as a text string, then apply SoftTFIDF [9], to compute a similarity score for each mention pair. Pairs with similarity score exceeding a threshold are clustered together. Next, based on the assumption that mentions are generated from, and typically shorter than their corresponding representatives, we select the mention with the longest name in each cluster as a representative, and create an entity with the same name for the cluster. The global set E of entities is then the union of all entities created in the documents.

6 Learning from Context

The aforementioned model ME exploits only mention names, i.e. title, middle initial, etc. to match mentions. We now build on ME to develop the model MEC, which exploits the context of mentions to improve matching accuracy. We consider two types of context: attributes external to names (age, gender, co-star, studio, etc.), and co-occurrence of entities.

6.1 Exploiting External Attributes

We associate with each mention a set of external attributes, defined based on the attributes of the database as well as the types of mentions we can automatically discover from the text. Recall (model ME Sec. 5) that after selecting a set of entities E_d for d, we generate a representative r for each entity $e \in E_d$, then generate mention m from representative r, by transforming the internal attributes of r.

In the current model MEC, we generate mention m from representative r by transforming *both internal and external attributes* of r. We compute this transformation probability as follows. Given any two elements $n_1, n_2 \in W$ (e.g., n_1 is a mention m and n_2 is a representative r), assuming independence among all attributes (both internal and external), we can compute probability $P(n_2|n_1)$ as a product distribution over attributes as in Eq. 3. The independence assumption clearly does not hold, but it reduces runtime complexity, and is shown empirically to work well (Sec. 8).

Let the set of internal and external attributes be $a_1, \ldots, a_p$. Let $n_1 = (a_1 = v_1, a_2 = v_2, ..., a_p = v_p)$ and $n_2 = (a_1 = v_1', a_2 = v_2', ..., a_p = v_p'))$. Since the external attributes could be of binary, numeric and textual value, we adopt a more general model to compute $P(v_k'|v_k)(k = 1, \cdots, p)$ for each pair of attribute values. (1) We first measure the distance between the corresponding attribute values $dist_k(v_k', v_k) = d_k$ using an attribute-specific distance metric. As default metrics, for textual attributes, we convert the Soft-TFIDF [9] similarity between them into a distance; for numeric attributes, such as user rating, we measure the Manhattan distance. This approach is general in that any state-of-art distance metric can be integrated into the model, as it becomes available. (2) We then compute $P(v_k'|v_k)$ as a variation of the Gaussian distribution (because d_k is always non-negative):

$$f_k(v_k'|v_k) \equiv \frac{1}{\sqrt{\pi/2}\sigma_k} \cdot \exp\left(-d_k^2/\sigma_k^2\right) \qquad (4)$$

Currently we assign a constant density to missing values, and found that it empirically works well, though more sophisticated methods are clearly possible.

We learn the standard variance σ_k for each attribute when learning the model as in Sec. 5.2. Given a set of annotated entity-representative pairs $\{(e, r)\}_1^n$, we compute the maximum likelihood estimation of σ_k for each attribute a_k as: $\sigma_k = [\frac{\sum_{(e,r)} dist_k(v_k, v_k')^2}{n}]^{1/2}$, where $dist_k(v_k, v_k')$ is the distance between corresponding attribute values in e and r.

6.2 Exploiting Entity Co-occurrence

Exploiting entity co-occurrence further improves matching accuracy (see example in Sec. 4.2. We currently exploit by: (1) modeling entity co-occurrence as conditional probability between entities $P(e_2|e_1)$; and (2) integrating it as an external attribute.

Modeling as Conditional Probabilities: In the document generation process (Section 5.1), instead of assuming independence among entities, we select entities sequentially according a conditional probability $P(e_i|E_d^{i-1})$: each entity e_i is selected into a document d according to the set of entities E_d^{i-1} selected before it. This gives $P(E_d) = \prod_{i=1}^{|E_d|}[P(e_{di}|E_d^{i-1})]$, where $E_d^0 = \emptyset$ and $P(e_{d1}|E_d^0) = P(e_{d0})$. Thus we have

$$P(d) \approx \prod_{i=1}^{|E_d|}[P(e_{di}|E_d^{i-1})P(r_{di}|e_{di})] \times \prod_{(r_{dj}, m_{dj})} P(m_{dj}|r_{dj}). \qquad (5)$$

Computing $P(e_{di}|E_d^{i-1})$ raises the challenge of ranking entities in a document in a sequential order, and also the sparsity problem when learning the model in an

unsupervised setting. To address these, we approximate $P(e_{di}|E_d^{i-1})$ as $max_{e_{dj} \in E_d, i \neq j} P(e_{di}|e_{dj})$. Next, we approximate $P(e_{di}|e_{dj})$ as $P(e_{di})$, if e_{di} and e_{dj} never co-occur, and as 1 otherwise. We now can apply these formulas directly in the Truncated EM algorithm (Section 5.4), to compute $P(e_{di}|E_d^{i-1})$.

Integrating as an External Attribute: We also integrate entity co-occurrance as an additional external attribute to each representative/entity. That is, we expand the representation of a representative/entity with an external attribute con. This set-valued attribute contains all other representative names in the same document. For example, for an author in a citation, its con attribute contains all other coauthor names. The con attribute of an entity is the combination of the con attributes of all its representatives in different documents.

Let the *con* attribute of a representative r and an entity e be $\mathsf{con}(r) = \{n_1, n_2, \ldots\}$ and $\mathsf{con}(e) = \{n'_1, n'_2, \ldots\}$. To measure their distance, we first apply the SoftTF-IDF [9] string metric to compute the similarity $s(n_i, n'_j) \in [0, 1]$ ($n_i \in \mathsf{con}(e), n'_j \in \mathsf{con}(e)$), then compute the distance as

$$dist_{\mathsf{con}}(r, e) \equiv \prod_{n_i \in \mathsf{con}(r)} [1 - max_{n'_j \in \mathsf{con}(e)} s(n_i, n'_j)] \quad (6)$$

The probability of the context of a representative being transformed from that of an entity is still computed by Equation 4. Note that we do not expand the representation of a mention since this co-occurring information does not benefit the mention level.

7 Knowledge Transfer via Contexts

As we have motivated in Section 4.3, often an entity can "borrow" some context from its neighboring entities, and leverage the augmented context to increase matching accuracy. Hence, in the final extension, MEC^2, we enable such "borrowing".

Similar to model MEC, in MEC^2 we add to each representative/entity a context attribute con. However, unlike MEC, this attribute now not only contains the co-occurring names in the same document, but also the names of "distant": co-occurring entities (e.g., co-occurring entities of co-occurring entities).

However, exploiting more distant entity dependency can hurt matching accuracy, if it links irrelevant entities together. The problem is then how far we should follow context of entities. Currently we adopt the following mechanism. Let $c_l(e)$ be the l-th context of e, namely, the set of entities that have a *recursive co-occurring relation* of distance no larger than l from entity e. We then consider the con attribute of an entity e to be the set $\{c_1(e), \ldots, c_k(e)\}$, for a pre-specified k (currently set at three in our experiments). The distance between the contexts of a representative and an entity is then a weighted sum of the distance over each level of context: $dist_{\mathsf{con}}(r, e) \equiv \sum_l w_l \cdot dist_{c_l}(r, e)$, where $dist_{c_l}(r, e)$ is defined as a distance between two sets of names, measured in the same way as in Equation 6. We currently apply a reciprocal weighting: $w_l \equiv 1/l$, to reflect the intuition that more distant contexts contribute less to the matching process.

8 Empirical Evaluation

We now present experimental results that demonstrate the utility of MEDIATE. We show that MEDIATE significantly increases accuracy over current baseline matching methods, and that it can utilize text to improve accuracy for record matching, and vice versa.

8.1 Experimental Settings

Data Sets: We evaluated MEDIATE on two data sets obtained from the Internet Movie Database IMDB at *imdb.com* and the CS Bibliography DBLP at *dblp.uni − trier.de*. From IMDB, we downloaded all news articles in 2003-2004 (to be treated as text documents in our experiments), then retrieved the IMDB home pages of people (such as actors, directors) and movies mentioned in the news articles. Next, we converted each home page into a structured record, thereby obtaining two tables: PEOPLE and MOVIES, whose schemas are shown in Figure 6.

From DBLP, we downloaded 472 home pages of authors, focusing on home pages with high degree of ambiguity. For each paper X in the downloaded home pages, we followed URL links to retrieve home pages of the conference that X was published in, as well as the HTML abstract (wherever available) that is a text blurb listing the conference name, author affiliation, and the paper abstract. The conference home pages and HTML abstracts are treated as the text documents in our experiments. Finally, we converted each paper citation to a structured record, thereby obtaining a table: CITATIONS, whose schema is shown in Figure 6.

We then marked up the mentions (people names, movie titles, and author names), exploiting the already existing HTML markups and employing an automatic tagger method whenever necessary. Next, we manually found all pairs of matching mentions, to be used in evaluating experimental results.

In the next step, following common research practice in record linkage [18, 2], we perturbed the tables of the data sets, to generate varying degrees of semantic ambiguity for experimental purposes. For the IMDB tables, we randomly selected records with a probability p, then perturbed each selected record in several ways, e.g., randomly adding titles and misspelling, and abbreviating the first names. For movie titles we randomly removed articles (a, an, the) and sequel numbers (e.g., Star War III → Star War), and added misspelling. We also randomly split records, by keeping certain mentions (e.g., certain actor names in attribute actors of table MOVIES in Figure 6), and dropping others. We also perturbed the DBLP table by randomly

IMDB: Two tables: people and movies; with 2,043 records and 868 text documents;

People: *<name, gender, brithdate, birthplace, deathdate, deathplace, movies>*

Movies: *<title,year,genre,runtime,language,country,director,color,rating,actors>*

Contains 9,725 mentions of 1,687 entities, and 55,147 correct matching pairs.

People have 1,231 records 4,227 mentions.

Movies have 812 records 5,498 mentions.

DBLP: One table of citations with 944 records and 721 text documents;

Citations: *<title, authors, coference/journal, pages, year>.*

Contains 7,356 mentions of 1672 authors, and 55,186 correct matching pairs.

Figure 6: Characteristics of the data sets.

removing middle names, and abbreviating first names.

Figure 6 describes a data set where all IMDB records were perturbed (i.e., $p = 1$). Our goal is to match the mentions of three types of entities: people, movies, and authors, in these data sets. We use this data set for experiments in Sections 8.2- 8.4. In Section 8.5, we present sensitivity analysis with data sets perturbed using varying p values.

Baseline Matching Methods: We compare MEDIATE with three methods commonly used in record linkage and matching mentions in text.

- *Pairwise matching of names:* This method declares two mentions matched if the similarity of their names exceeds a threshold. For computing similarities, we use SoftTF-IDF, a measure described in [9] and shown empirically to be the best among several.
- *Clustering:* Many different clustering algorithms have been developed for record linkage (e.g., [25, 10]), as well as mention matching in text [22]. We implemented a variation of these algorithms, using the SoftTF-IDF measure [9] to compute similarities between mention names.
- *Pairwise LW (linear weight) record linkage:* When examining MEDIATE's performance on the task of record linkage, we also want to compare it to state-of-the-art record linkage methods. Numerous such methods have been developed in the past few years (see Section 3), but no comprehensive study is available yet to evaluate them. For our experiments, we implemented the pairwise attribute-based method, which has been applied successfully in many database and AI works [18, 35, 9]. Given two records, this method computes a similarity score between each pair of corresponding attributes (using attribute-specific similarity measures), then combines the scores and deciding the match using linear weighted sum, or learning methods such as decision tree, SVM, etc. [35]. We experimented with a small set-aside developing set and found linear weighted sum work best.

Performance Measures: We convert the outcome of each matching method into a set M_p of mention pairs that are predicted to match. Since we want to retrieve all and only matching pairs, we use precision, recall, and F-1 to measure the method's performance. Specifically, let M_a be the set of all matching pairs from the data set (as determined manually). Then precision $P = |M_p \cap M_a|/|M_p|$, recall $R = |M_p \cap M_a|/|M_a|$, and $F-1 = (2P \cdot R)/(P + R)$. These measures are commonly used in record linkage and mention matching in text [22].

8.2 Overall Matching Accuracy

Table 1: Matching accuracy over both databases & text.

F_1 (R/P)	IMDB		DBLP
Entity Type	Person (4227)	Movie (5498)	Author (6356)
Pairwise	60.5 (65.7/56.0)	75.0 (84.4/67.4)	67.4 (66.0/68.9)
Clustering	54.2 (74.7/42.5)	76.7 (77.3/76.1)	61.9 (68.1/56.9)
Model ME	74.1 (63.6/88.8)	77.5 (75.7/79.3)	77.7 (86.3/70.6)
Model MEC	74.7 (63.3/91.0)	80.7 (76.7/85.1)	78.5 (86.3/72.0)
Model MEC^2	77.2 (67.3/90.5)	81.7 (78.1/85.6)	81.6 (85.9/77.8)

Table 1 shows the accuracy of different methods for mention matching over both databases and text. The rows show the F-1 values (with R and P in parentheses) for pairwise matching, clustering, ME, MEC, and MEC^2 (i.e., the complete MEDIATE system). Note that the LW record linkage method is not applicable because it cannot extract attribute values for mentions in the text documents.

The results show that MEC^2 achieves high accuracy across the entity types in both IMDB and DBLP, ranging from 77.2 to 81.7% F-1. In contrast, the best baseline methods (pairwise for actors and authors, and clustering for movies) obtain only 60.5 - 76.7% F-1.

Compared to the best baseline, applying ME significantly improves accuracy by 10.3 - 13.6% (except 0.8% for movies). Exploiting context and entity co-occurrence in MEC further improves accuracy by 0.6 - 3.2%. Exploiting recursive context in MEC^2 adds 1 - 3.1%. In all our experiments, subsequent versions of MEDIATE outperform previous ones, confirming that our generative model is able to exploit immediate context, entity co-occurrence, and recursive context.

An analysis of the results shows that the accuracy gains depend on the nature of transformations for mentions, as well as the discriminative power of the context. For instance, movie titles usually are not transformed as frequently or significantly as person names. This explains why the basic MEDIATE which relies only on movie titles to match movies obtained only a minimal improvement over pairwise and clustering.

Finally, Table 2 shows the number of real-world entities that MEC^2 estimated in each iteration of the Truncated EM algorithm. The final estimated numbers of entities, and the correct number of entities are in the last second lines, respectively.

Table 2: Number of entities, as estimated in each iteration.

	Person	Movie	Author
Initialization	2111	2611	4124
1st Iteration	1423	1559	2145
Last Iter. (between 5-8)	963	927	1382
Annotated	890	797	1672

Accuracy over Databases, Text, and Cross-Linking: To further understand the above results, we break the accuracy down into "within database",

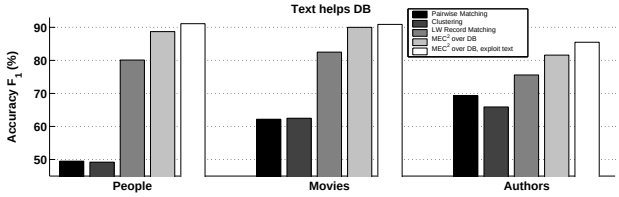


Figure 7: MEC^2 achieves significantly higher accuracy than LW record linkage when applied to databases, and obtains even higher accuracy when exploiting text.

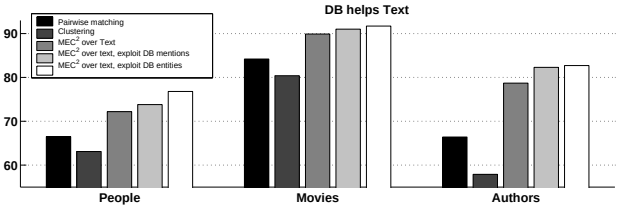


Figure 8: MEC^2 can exploit databases to improve accuracy over text.

"within text", and "across database and text", respectively. The results (not shown on figures) demonstrates that MEC^2 (i.e., the complete MEDIATE) outperforms the baselines across all three entity types, and achieves accuracy of 70 - 91% F-1, while the best baseline method achieves 51.8 - 84.8% F-1. This suggests that MEDIATE can link mentions within databases, text, and across them with high accuracy.

8.3 Exploit Text to Improve Record Linkage

Figure 7 shows the accuracy of MEDIATE in matching records on the database side. For each of the three entity types people, movies, and authors, the first four bars show the F-1 accuracy of the pairwise matching method, clustering, LW record matching, and MEC^2, when they are given only the databases (with no associated text). The last bar shows the accuracy of MEC^2 when it is also given text documents (as described in Figure 6) and can exploit them for record matching purposes. The results show that, first of all, record matching beats baseline methods, which exploit only names, to reach accuracy of 75.6-82.5% F-1. Second, MEC^2 even without the help of text beats record matching significantly, improving accuracy by 6 - 8.6% F-1, to reach 81.6 - 90%. Finally, when text is available, MEC^2 can exploit it to improve accuracy across all three entity types, by 0.9 - 3.9%.

8.4 Exploit DBs to Match Text Mentions

Figure 8 shows the accuracy of MEDIATE in matching mentions in text. Again, for each entity type, the first three bars show the accuracy of pairwise matching, clustering, and MEC^2 when they are not given any associated database. The fourth bar shows the accuracy of MEC^2 when it is given a database to aid in matching mentions in text. The fifth bar describes a situation similar to that of the fourth bar, but here MEC^2 is also told that the database contains *all* entities whose

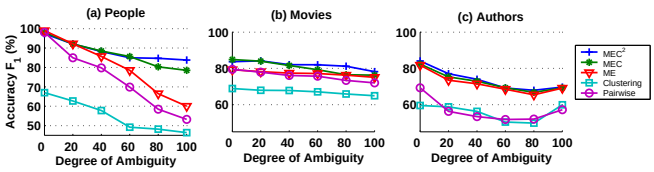


Figure 9: The MEDIATE system is robust across a broad range of degrees of semantic ambiguity.

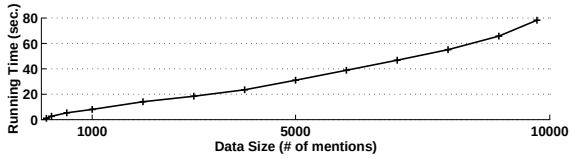


Figure 10: Runtime of MEC^2 as a function of data size.

mentions appear in text (a situation that commonly arises in practice).

The results show that, on text side alone, best baseline (pairwise or clustering) achieves 66.4 - 84.2%, whereas MEC^2 achieves 72.2 - 90%, resulting in a gain of 5.7 - 12.4%. It also shows that MEC^2 can exploit the given databases to improve accuracy by 1.8 - 4.6%, to reach 76.8 - 91.7%.

8.5 Sensitivity Analysis

Figure 9 shows the accuracy of the matching methods over different degrees of semantic ambiguity. The data points at, say, value 60 on the X axis, represent the F-1 accuracies when run on a data set that was created by perturbing the original IMDB and DBLP data sets with $p = 0.6$. The results show that MEDIATE is robust to varying degrees of semantic ambiguity.

8.6 Efficiency

In our implementation, we have optimized MEDIATE for efficiency, using the canopy/window techniques detailed in [25, 18], as well as inverted indexing, to reduce the number of representative/entity pairs that are compared in each iteration of the Truncated EM algorithm (Section 5.4). Because of space limitation, we defer the detailed description of these techniques to the full paper. With these optimizations, our implementation ran under three minutes on an Intel Xeon, with 1G memory and 2.5G CPU. Figure 10 shows the runtime of MEDIATE as a function of the data size.

9 Conclusion and Future Works

We have proposed the MEDIATE approach which automatically matches entity mention *within* and *across* both text and databases. At the heart of our approach is a *generative model* that provides a probabilistic view on how a data creator might have generated mentions into a database record or a text document. MEDIATE can employ the model to match multiple types of entities, with no need for expensive annotated training data. MEDIATE can also integrate multiple types of relevant information into the model, such as the similarity of mention names, context information such as

age, gender, and co-occurrence among entities. Experiments on real-world data show that MEDIATE significantly outperforms existing methods, and that it can exploit text to improve record linkage, and vice versa. Besides more extensive evaluation of MEDIATE, we plan to extend it to exploit limited supervision (e.g., known number of entities) to better learn the models, and to perform aggregate operations on the found matches (e.g., to obtain reliable answers to questions such as "how many movies has Peter Jackson directed?").

References

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

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proc. of VLDB-02*.

[3] A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. pages 79–85. Association for Computational Linguistics, 1998.

[4] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *Proc. of the 9th ACM SIGMOD DMKD Workshop*, 2004.

[5] M. Bilenko and R. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report Technical Report AI 02-296, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX, Feb. 2002.

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

[7] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proc. of SIGMOD-03*, 2003.

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

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

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

[11] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley and Sons, 2003.

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

[13] D. Freitag. Multistrategy learning for information extraction. In *Proc. 15th Int. Conf. on Machine Learning (ICML-98)*, 1998.

[14] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proc. of IJCAI-99*, 1999.

[15] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An extensible framework for data cleaning. In *Proc. of ICDE-00*.

[16] V. Ganti, S. Chaudhuri, and R. Motwani. Robust identification of fuzzy duplicates. In *Proc. of ICDE-05*, 2005.

[17] L. Gravano, P. Ipeirotis, N. Koudas, and D. Srivastava. Text join for data cleansing and integration in an rdbms. In *Proc. of ICDE-03*.

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

[19] L. Jin, C. Li, and S. Mehrotra. Efficient record linkage in large data sets. In *Proc. of DASFAA-03*, 2003.

[20] A. Kehler. *Coherence, Reference, and the Theory of Grammar*. CSLI Publications, 2002.

[21] N. Koudas, A. Marathe, and D. Srivastava. Flexible string matching against large databases in practice. In *Proc. of VLDB-04*, 2004.

[22] X. Li, P. Morie, and D. Roth. Identification and tracing of ambiguous names: Discriminative and generative approaches. 2004.

[23] X. Li, P. Morie, and D. Roth. Robust reading: Identification and tracing of ambiguous names. In *Proceedings of HLT-NAACL*, 2004.

[24] G. Mann and D. Yarowsky. Unsupervised personal name disambiguation. 2003.

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

[26] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[27] A. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, 1996.

[28] V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. 2002.

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

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

[31] E. Rahm and P. Bernstein. On matching schemas automatically. *VLDB Journal*, 10(4), 2001.

[32] E. Rahm and H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[33] V. Raman and J. Hellerstein. Potter's wheel: An interactive data cleaning system. In *The VLDB Journal*, pages 381–390, 2001.

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

[35] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proc. of SIGKDD-02*.

[36] W. Soon, H. Ng, and D. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics (Special Issue on Computational Anaphora Resolution)*, 27:521–544, 2001.

[37] S. Tejada, C. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of the 8th SIGKDD Int. Conf. (KDD-2002)*, 2002.

[38] M. Weis and F. Naumann. Dogmatix tracks down duplicates in xml. In *Proc. of SIGMOD-05*, 2005.

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

[40] W. Yih and D. Roth. Probabilistic reasoning for entity and relation recognition. In *Proc. of COLING'02*, 2002.