TOWARD CONCEPT-BASED TEXT UNDERSTANDING AND MINING

BY

XIN LI

M.S., Peking University, 2000
B.S., Wuhan University, 1997

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2005

Urbana, Illinois

Abstract

There is a huge amount of text information in the world, written in natural languages. Most of the text information is hard to access compared with other well-structured information sources such as relational databases. This is because reading and understanding text requires the ability to disambiguate text fragments at several levels, syntactically and semantically, abstracting away details and using background knowledge in a variety of ways. One possible solution to these problems is to implement a framework of concept-based text understanding and mining, that is, a mechanism of analyzing and integrating segregated information, and a framework of organizing, indexing, accessing textual information centered around real-world concepts.

A fundamental difficulty toward this goal is caused by the concept ambiguity of natural language. In text, the real-world entities are referred using their names. The variability in writing a given concept, along with the fact that different concepts/enities may have very similar writings, poses a significant challenge to progress in text understanding and mining. Supporting concept-based natural language understanding requires resolving conceptual ambiguity, and in particular, identifying whether different mentions of real world entities, within and across documents, actually represent the same concept.

This thesis systematically studies this fundamental problem. We study and propose different machine learning techniques to address different aspects of this problem and show that as more information can be exploited, the learning techniques developed accordingly, can continuously improve the identification accuracy. In addition, we extend our global probabilistic model to address a significant application – semantic integration between text and databases.

Acknowledgments

A lot of people have supported me in this challenging, hard-working but also entertaining course of PH.D. study – a continuous process of discovering new problems and pursuing answers to unsolved questions. First, I am deeply grateful for my adviser, Dan Roth, for all of his teaching and guidance throughout my time as a doctorate student. Working with him makes me always able to notice the places that I can improve over myself, both in research and in personality. This pleasant learning process provides me great excitement when I look back at every step I have taken. I would also like to thank the other members of my committee, which include Jiawei Han, Gerald Dejong and Anhai Doan, for their valuable suggestions and comments on this thesis work.

Next, I would like to thank all the people in the cognitive computation group. Thanks for the many productive discussions at the lab, a lot of warm-hearted comments for improving my presentation skills and writing skills as a non-native speaker of English, and also numerous creative jokes to make this PH.D. life so entertaining. Particularly, I would like to thank Wen-tau Yih, Vasin Punyakanok, Dav Zimak, Paul Morie, Yuancheng Tu, Steve Hanneke and Kevin Small who have directly collaborated with me on many research projects.

I owe my great thinks to my family. My parents Zihua Li and Ronghua Zhang, and my little sister Yan Li, who have always been encouraging and supporting to myself. It is their every regard through the phone line across the Pacific Ocean that supports me in my deep heart in this process.

I also thank all the friends I have made here and thank Urbana-Champaign, this beautiful land and the first station when I am in the United States.

This research is also supported by NSF grants IIS-9801638, ITR IIS-0085836 and EIA-0224453, an ONR MURI Award and an equipment donation from AMD.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI Artificial Intelligence.

BLOG Bayesian LOGic.

CRF Conditional Random Fields

DB Database.

EM Expectation-Maximization.

IE Information Extraction.

IR Information Retrieval

LMR The pairwise classifier named after Li, Morie and Roth.

ME Matching Entities.

MEC Matching Entities with Context.

MEC^2 Matching Entities with More Context.

MEDIATE Matching Entities in Data Instances And TExt.

MLE Maximum Likelihood Estimation.

NLP Natural Language Processing.

POS Part-of-speech.

QA Question Answering.

SDC Supervised Discriminative Clustering.

SNoW Sparse Network of Winnow.

Chapter 1

Introduction

There is a huge amount of text information in the world. According to Google's statistics, there have been more than 8 billion web pages on the Internet by 2005. Every year, the United States publishes more than $150,000$ new books [1]. The DBLP database itself – a famous research paper collection, have collected more than $600,000$ research papers that are published in the area of computer science between 1980-2005. Moreover, millions of news articles about politics, sports, and business, come out every day.

Most of the above information is written in natural languages, which is hard to access compared with other well-structured information sources such as relational databases. This is because reading and understanding text requires the ability to disambiguate text fragments at several levels, syntactically and semantically, abstracting away details and using background knowledge in a variety of ways. Therefore, how to efficiently access a large collection of text information, that is, how accurately and automatically pinpoint relevant information for a user and help him understand it, is a very challenging problem.

One relatively successful technology to facilitate intelligent access to textual information is the search engine techniques (van Rijsbergen, 1979; Salton & McGill, 1983; Fuhr, 1992; Fuhr, 2001; Lafferty & Zhai, 2001; Lafferty & Zhai, 2002). Given a user query – typically a set of key words providing a description of the target information, a search engine searches the Internet or other text collection and identifies the most relevant documents related to the query. In the following example

[1] statistics in 2002, cited from "Self-publishing will spur book industry to modernize" by Laura Vanderkam.

(see Figure 1.1), when a user looks for George Bush's foreign policy, the process of identifying relevant web pages in a search engine is conducted on the basis of key-word matching – pages containing the same set of key words as in the query are assumed to be relevant.

One critical problem with this process (which we call *string or mention-level processing*) is that of ignoring semantic understanding and matching of text. One consequence in the above example is that semantically equivalent concepts could not be located in text when ambiguous mentions of them are used. Suppose there is a web page containing only the name "Bush", rather than "George Bush'. Even if it refers to the same person conceptually in the above example, it will not be treated as relevant, only because a different name from that in the query is used in the page.

Figure 1.1: **An example of searching information.**

Other commonly studied text understanding and mining tasks related to intelligent access to text information, are summarized in Figure 1.2[2]. For example, information extraction (Califf &

[2]modified based on an poster by NEC.

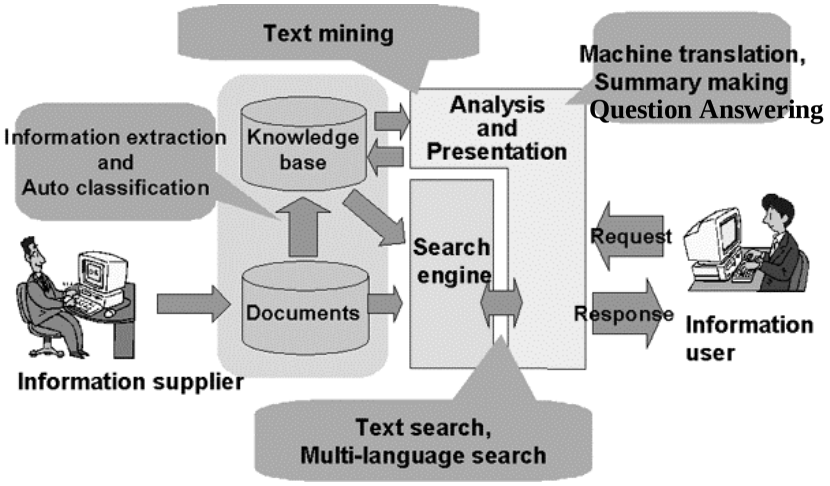

Figure 1.2: **Text understanding and mining.**

Mooney, 1999; Chieu & Ng, 2002; Freitag, 2000; Lafferty, McCallum, & Pereira, 2001; Roth & Yih, 2001) aim at extracting text segments that are related to a specific topic, such as the position and the company name of a job opening, the starting time or the speaker of a seminar. Compared with search engines, a question answering system (Light, Mann, Riloff, & Breck, 2001; Moldovan, Harabagiu, Girju, Morarescu, Lacatusu, Novischi, Badulescu, & Bolohan, 2002; Moldovan, Pasca, Harabagiu, & Surdeanu, 2002; Roth, Cumby, Li, Morie, Nagarajan, Rizzolo, Small, & Yih, 2002; Voorhees, 2002) attempts to support higher-precision information access, pinpointing the exact answer from a large collection of text for a user's question – asked in standard English or other natural languages. Most of these current techniques applied to these tasks still rely on string or mention-level processing. We will further analyze the problems with them and introduce later concept-based text understanding and mining, an idea of semantically processing real-world concepts and entities, rather than ambiguous mentions of them in text.

Machine Learning techniques (Charniak, 1993; Roth, 1999) such as rule-learning, decision trees, Neural Networks, Support Vector Machines, Graph-based Models, discriminative and generative models, have been widely applied to text understanding and mining tasks these days, to achieve different levels of analysis and understanding of text. Most of these text-related problems are typically formalized as different classification problems, such as binary classification (Khardon,

Roth, & Valiant, 1999), multi-class classification (Hindle, 1990; Even-Zohar & Roth, 2001; Li & Roth, 2002; Sang & Meulder, 2003) and structure-based classification and inference (Munoz, Punyakanok, Roth, & Zimak, 1999; Punyakanok & Roth, 2001), which aim at mapping language components into discrete classes or structures reflecting different syntactic and semantic abstraction and understanding of them. The text categorization (Dagan, Karov, & Roth, 1997; Zhang & Oles, 2001) which classifies text articles into a number of topics, such as politics, sports and education, a type of semantic understanding of articles, is modeled as a multi-class classification task. Examples of sequential and structure-based classification tasks include part-of speech tagging (Kupiec, 1992; Brill, 1995; Brill, 1997) which categorizes a word into a noun, verb, adjective and other classes, and parsing (Collins, 1997; Collins, 1999; Charniak, 2000; Collins & Duffy, 2002) which recognizes syntactic structures of sentences.

In the rest of this chapter, we will further discuss the motivation and ideas behind concept-based text understanding and mining in Section 1.1, introduce the common machine learning techniques that will be applied to achieving this goal in Section 1.2, and then summarize our contributions in these areas in Section 1.3. Finally in Section 1.4, we will briefly introduce the organization of this thesis.

1.1 Toward Concept-Based Text Understanding and Mining

1.1.1 An Overview of Current Text-Related Techniques

Most of the work in the direction of understanding and accessing text in the past two decades has concentrated on syntactic analysis — the study of the underlying mechanisms, structures and principles of how a natural language sentence is composed and generated from syntactic components (Charniak, 1997; Collins, 1997; Li & Roth, 2001; Punyakanok & Roth, 2001). Less attention has been paid to semantic analysis of text — analysis that focuses on mapping syntactic components to their corresponding real-world concepts, as well as on their properties and the relations

between them; that is, *the meaning* of sentences. In addition to information extraction and question answering, other applications on this list include named entity recognition (Collins & Singer, 1999; Sang & Meulder, 2003), text categorization (Yang, 1999; Zhang & Oles, 2001), reading comprehension (Hirschman, Light, Breck, & Burger, 1999) and so on.

While syntactic analysis tasks can achieve quite accurate performance by exploiting the local context of language fragments where a set of formalized machine learning techniques can be applied directly, most semantically related information is separated and distributed in a much broader range of texts. Therefore, we call the tasks that work on words and syntactic fragments, *string-based text processing*. One of the bottlenecks of these semantic tasks is the lack of an integrated analysis of these fragments and their connections to real-world concepts.

Figure 1.3 illustrates what is missing in current semantic understanding tasks of text, compared with the understanding by a human being. Given the sentence "Mary gave John an apple", the current natural language processing such as parsing can identify the syntactic fragments and the relation between them. Based on this understanding, higher-level processing can further understand the sen

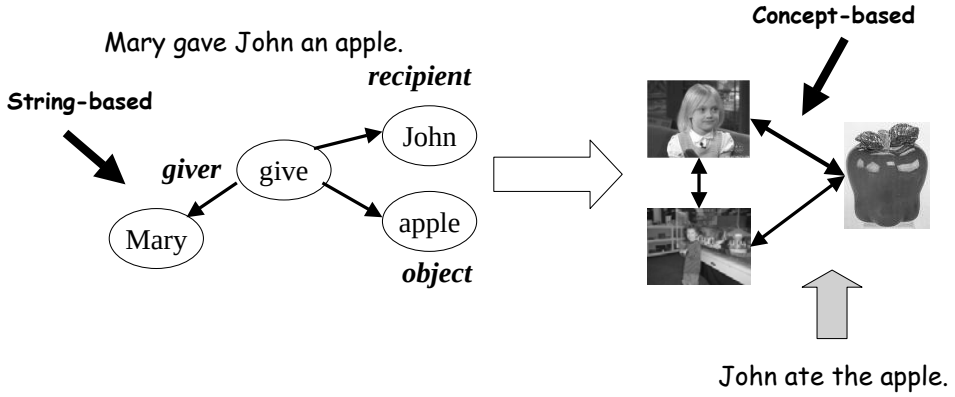

Figure 1.3: **Difference in text understanding by a computer and a human being**

One critical module is missing in these tasks which, however, lays the ground for the understanding by a human being is the automatic mapping from syntactic fragments to real-world concepts. That is, even if he does not know the specific persons "Mary" and "John", he may still

understand that there must exist some persons referred in the text, and "Mary" and "John" are references to them. When a new sentence "John ate the apple" comes in, he would be able to recognize that the second sentences mentions one same person, and thus integrate it with the information he has already known about this person. Based on this automatic entity identification and mapping, information is organized and integrated around real-world concepts in a his mind. Moreover, his further understanding can take these concepts as a basic unit, not being influenced by variations of the lower-level text fragments. The necessity of moving to concept-based text processing can be further illustrated in a question answering task.

Question: When were William Shakespeare's twins born?
Answer: 1585
Text having the answer: They are the evidence for **Shakespeare**'s marriage by special licence at 18, to Anne Hathaway, of nearby Shottery, aged 26. Anne gave birth to a child, Susanna, after six months of wedlock. Two years later came the birth of the **Shakespeare** twins Judith and Hamnet, girl and boy, baptized in **1585**. Hamnet died aged 11 in 1596.

Figure 1.4: **An example of the question answering task.**

In the open-domain question answering task (an example is shown in Figure 1.1.1) (Light, Mann, Riloff, & Breck, 2001; Moldovan, Harabagiu, Girju, Morarescu, Lacatusu, Novischi, Badulescu, & Bolohan, 2002; Moldovan, Pasca, Harabagiu, & Surdeanu, 2002; Roth, Cumby, Li, Morie, Nagarajan, Rizzolo, Small, & Yih, 2002; Voorhees, 2002), given a factual question[3] written in standard English or other languages, a question-answering system is required to be able to locate and extract the exact answer to it from a large collection of textual resources (for example, news article or web pages). To accurately answer a question related to a real-world entity or concept, as in the example, several critical problems need to be solved:

1. Can a question answering system precisely identify the entity from its references in both the question and the text ? In the previous example, the system needs to know that "Shakespeare" in the text refers to the same person as "William Shakespeare" in the question. A

[3]We do not address questions like "Do you have a light?", which calls for an action, but rather only factual Wh-questions.

more difficult example is “What is Bush’s foreign policy after 9-11?”. There are multiple prominent “Bush’s” in the real world, a human can easily figure out that the question refers to “George W. Bush” with a great probability given some background knowledge while a computer system can not.

2. Can a system automatically locate all the occurrences of this entity in the textual collection or knowledge base ? Most current information retrieval techniques (Baeza-Yates & Ribeiro-Neto, 1999; Kobayashi & Takeda, 2000) still work on term-based search and indexing, lacking the capacity of identify occurrences of an entity from those of other entities with similar writings.

3. Can a system automatically extract facts about this entity — its properties, the relations with other entities and what happened to it, after identifying the entity in the text collection ? The facts about an entity may be distributed in different places of the text. Current information extraction techniques (Chieu & Ng, 2002; Lafferty, McCallum, & Pereira, 2001; Roth & Yih, 2001) can only exploit the local context of a mention of an entity and do not provide an effective mechanism to integrate the segregated pieces of information and to make global inference using them.

The above problems are indispensable to many other semantic analysis tasks as well, but they have not been solved effectively so far. One possible solution to these problems and also the hope of these semantic tasks is to implement a framework of concept-based text understanding and mining, that is, a mechanism of analyzing and integrating segregated information, and a framework of organizing, indexing, accessing textual information centered around real-world concepts. This description is in contrast to the string or mention-level text understanding and mining which works directly over tokens or string without treat different occurrences of the same real-world entity as a whole, and without integrating scattered information about it together.

1.1.2 Implementing Concept-Based Text Understanding and Mining

One of the fundamental difficulties toward concept-based natural language processing is caused by the concept ambiguity of natural language. By *concepts and entities*, we refer to the real-world objects like people, companies, products, locations, and other abstract objects like events. We distinguish between the name of a real-world entity and the entity itself. For example, the name "George W. Bush" is only a string representation of a real person – the incumbent president of the United States, while "White House" is the name of a location. In text, the real-world entities are referred using their names. The variability in writing a given concept, along with the fact that different concepts/enities may have very similar writings, poses a significant challenge to progress in natural language processing.

The goal in this thesis is to describe our effort to move the level of understanding and mining from syntactic fragments representing concepts in text (*"mentions"* of concepts) to the real-world concepts represented by the fragments. Several tasks that are tightly related to the implementation of this idea, include: (1) named entity recognition, (2) entity disambiguation and Identification; (3) indexing and semantic integration of textual information based on real-world entities. The task of Named Entity Recognition is to recognize possible names of entities in text and categorize them into different semantic types, such as personal names, company names and names of locations. An example is of a tagged sentence from MUC 7 named entity recognition task (MUC-7, 1999) is as follows:

Example 1.1.1 (Named Entity Recognition) *[LOCATION Italy] 's business world was rocked by the announcement [DATE last Thursday] that Mr. [PERSON Verdi] would leave his job as vice-president of [ORGANIZATION Music Masters of Milan, Inc] to become operations director of [ORGANIZATION Arthur Andersen].*

Although the learning techniques for this task have gained significant improvement over a limited set of entity types recently (Zhang & Johnson, 2003; Sang & Meulder, 2003; Sarawagi & Cohen, 2004), how to accurately identify names of a broad of entity types such as professions,

colors and so on, is still a challenging problem. Since this task is not the focus of this thesis, we assume that the names and their types have been recognized and are given as input to the next stage.

Our major focus in this thesis is on entity disambiguation and identification, that of reading concepts from the named references in the context of text, and mapping them to their corresponding real world entities, that is, solving the *Name Ambiguity* in natural languages. Unfortunately, due to the difficulty caused by language ambiguity, most current techniques still directly deal with syntactic fragments and individual mentions of concepts, without considering the information of a concept as a whole.

After all the occurrence of a real-world entity has been identified, information about this entity which previously scatters in different texts or different context of the same text can be indexed and integrated based on it. Consequently, a lot of text understanding and mining tasks, such as Information Retrieval, Information Extraction Question Answering, Text Summarization and Reading Comprehension, can directly work on the concept-level rather than being bothered by ambiguous names of them.

Figure 1.5 presents the underlying mechanism supporting concept-based text understanding and mining and an example. All the occurrences of the same entity (in this case, George W. Bush) are identified and indexed together in text. When a user wants to find out "President Bush's foreign policy", even if the text does not contain the exact mention of "President Bush", the correct information can still be located with the help of this entity disambiguation and identification mechanism.

1.1.3 Entity Identification in Text

A description of the name ambiguity in text is: most names of people, locations, organizations and other concepts or entities, have multiple writings that are being used freely within and across documents (Li, Morie, & Roth, 2004a; Li, Morie, & Roth, 2004b).

Consider, for example, an open domain question answering system (Voorhees, 2002) that at-

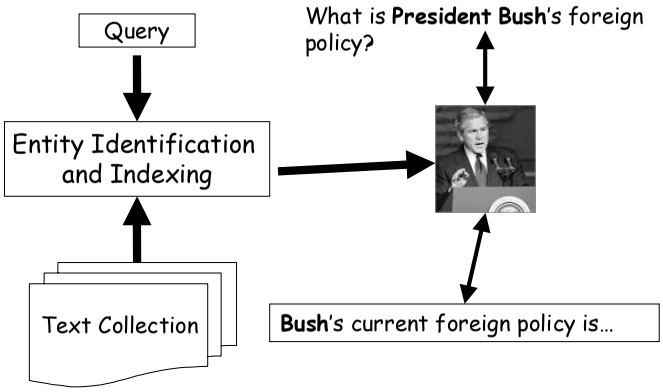

Figure 1.5: **An example of concept-based information access.**

tempts, given a question like: "When was President Kennedy born?", to search a large collection of articles in order to pinpoint the concise answer: "on May 29, 1917." 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 observing that the document discusses Kennedy's batting statistics. A similar problem exists for other entity types, such as locations and organizations. A further example is shown in Figure 1.6 which lists three passages containing mentions of two different Kennedy's.

Solving this fundamental problem – the cross-document entity identification problem – can already help to address several fundamental aspects of concept-based natural language processing, presented here from the perspective of the question answering task:

(1) *Entity Identity* - do mentions A and B (typically, occurring in different documents, or in a question and a document processed in search of an answer) refer to the same entity? This problem requires both identifying when different writings refer to the same entity, and when very similar or identical writings refer to different entities.

(2) *Name Expansion* - given a name of an entity (say, in a question), find other likely names of the same entity.

(3) *Prominence* - given a question "What is Bush's foreign policy?", and given that any large collection of documents may contain several Bush's, there is a need to identify the most prominent, or relevant "Bush", perhaps taking into account also some contextual information. Ad hoc solutions to this problem, as we show, fail to provide a reliable and accurate solution.

Document 1: The Justice Department has officially ended its inquiry into the assassinations of *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 2: In 1953, Massachusetts *Sen. John F. Kennedy* married Jacqueline Lee Bouvier in Newport, R.I. In 1960, Democratic presidential candidate *John F. Kennedy* confronted the issue of his Roman Catholic faith by telling a Protestant group in Houston, "I do not speak for my church on public matters, and the church does not speak for me."
Document 3: *David Kennedy* was born in Leicester, England in 1959. $\cdots$ *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 1.6: **An example of the name ambiguity.** There are many "Kennedy's" (in italic font) in the three documents.

There is little previous work we know of that directly addresses the problem of cross-document entity identification from their proper names in a principled way, but some problems related to the general entity disambiguation and identification problem have been studied.

From the natural language perspective, there has been a lot of work on the related problem of co-reference resolution (Soon, Ng, & Lim, 2001; Ng & Cardie, 2003; Kehler, 2002). The goal is to link occurrences of noun phrases and pronouns, typically occurring in a close proximity within a few sentences or a paragraph, based on their appearance and local context. Machine learning approaches to this 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 corresponds to a given noun phrase. Approaches differ in the algorithm used and features extracted.

In the context of databases (Cohen & Richman, 2002b; Hernandez & Stolfo, 1995a; Bilenko

& Mooney, 2003; Doan, Lu, Lee, & Han, 2003) several works have looked at the problem of record linkage - recognizing duplicate records in a database. (Pasula, Marthi, Milch, Russell, & Shpitser, 2002) considers the problem of identity uncertainty in the context of citation matching and suggests a relational probabilistic model. Other machine learning techniques(Cohen & Richman, 2002b; Bilenko & Mooney, 2003; Doan, Lu, Lee, & Han, 2003) to this problem are similar to the approaches used for co-reference resolution. They 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 labelled as matching or non-matching 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.

A few works address some aspects of the cross-document entity identification problem with text data and study it in a across-document setting (Mann & Yarowsky, 2003; Bagga & Baldwin, 1998; McCallum & Wellner, 2003; Gooi & Allan, 2004). (Mann & Yarowsky, 2003) considers one aspect of the problem – distinguishing occurrences of *identical* names in different documents, and only for 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 cross-document entity identification problem. (Bagga & Baldwin, 1998) builds a cross-document system based on an existing co-reference 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 co-reference chains. (McCallum & Wellner, 2003) uses a conditional model to address the problem of co-reference across documents. This work takes a more global view in that it defines a conditional probability distribution over partitions of mentions, give all observed mentions. The derived pairwise classification function that decides 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 to.

1.1.4 Semantic Integration Across Text and Databases

In addition to entity identification in text, we also study a more complex problem: semantic integration across unstructured text and structured databases. The goal of this task is to implement intelligent access to textual information from a different perspective, by combining efficient access mechanism in relational databases with a greater amount of textual information. Moreover, 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.

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:** 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 5.1, 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). 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 extrac-

tion, question answering, and cross-document entity identification (Li, Morie, & Roth, 2004b), 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.

1.2 Supervised and Unsupervised Learning

A very general definition of a learning problem given by Tom Mitchell (Mitchell, 1997) is as follows:

Definition 1.2.1 *A computer program is said to* learn *from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Usually, the choosing of the representation of the possible target (hypotheses, e.g. linear functions, DNF rules or others), the format of the experience (training examples), the performance measure and the learning algorithm, decides the learning process. Therefore, many learning problems fall into this general definition, but differ in different targets, different experiences, different performance measures and algorithms.

The advantage of machine learning techniques in most of these task, is that a system can automatically learn rules or discriminative and probabilistic models, with the help of limited training examples in the domain, that can accurately formalizes a problem and can be used for future prediction. In text categorization (Zhang & Oles, 2001), as an instance, a classifier (usually a function

defined over some informative features of an article) that can precisely categorize a set of training articles, is chosen for classifying new articles. In addition, machine learning techniques are very flexible in the sense that for a new problem, a new classifier for a new problem can be trained in a data-driven way, without the cost of rebuilding everything as manually-designed heuristic approaches.

True class labels of training examples are usually annotated or provided by domain experts according to their understanding of the problem, and serve as supervision in training. Depending on whether the training examples are labeled (the class label of each example in known) or unlabeled, machine learning techniques are categorized into the classes of supervised learning and unsupervised learning. However this process of acquiring supervision is typically unrealistic or very time-consuming in some problems. In the above example of text categorization, thousands of articles are required to be first annotated as related to different topics and then are used to train some accurate classifiers for new articles. Therefore, when the class labels of training examples are hard to acquire, clustering approaches (Brown, deSouza R. Mercer, Pietra, & Lai, 1992; Dagan, Lee, & Pereira, 1999; Kamvar, Klein, & Manning, 2002) are usually exploited as an unsupervised approach – an optimization procedure that classify a set of elements to optimize some criteria designed on the basis of the commonality and difference between language components, without exploiting the true labels of these elements. Clustering approaches have been widely applied in natural language processing and it has been shown repeatedly that its success depend on defining a good (similarity) distance metric to measure the commonality of language components, one that is appropriate for the task and the clustering algorithm used.

In this thesis, we study the problem of how to apply learning techniques to entity identification. As we will show, a set of commonly used learning techniques such as classification, clustering and generative probabilistic models can achieve decent performance in this domain. Moreover, we develop some new approaches based on the existing learning techniques that can significantly improve the performance.

1.2.1 Classification

One of the approach to perform syntactic and semantic abstraction of text fragments is classification: Given a set of labeled training examples S, the goal is to seek a hypothesis (classifier) $h : X \to C = \{1, 2, \cdots, K\}$ in a hypothesis space H that can map each example $x \in X$ to a class index $h(x) \in C$. When $K = 2$, it becomes a binary classification task ($C = \{0, 1\}$ in this case), compared with the multi-class classification task when $K > 2$. Each element $x \in X$ is represented as a feature vector $x =< v_1, v_2, \cdots, v_m >$. Features are a set of attributes that are chosen to describe a data element. For example, color can be used to describe clothes, while temperature and precipitation can be used to describe weather. Features are usually converted into numeric values in a hypothesis.

A hypothesis is typically parameterized as a function over a set of features. For example, a linear threshold hypothesis is defined as $h(x) = I(\sum_i w_i \cdot v_i) \geq T$ where T is a real-valued threshold, where $I(true) = 1$ and $I(false) = 0$ is an indicator. How to get informative features that can accurately reflect some property of an object is one of the most critical problems in a text-related learning tasks.

A target function p gives the true labeling of each data element $x \in X$, where $p(x)$ is the true label of x. p may or may not belong to the hypothesis space. In this sense, the learning procedure in classification is to find a hypothesis h in H to approximate p. The part-of-speech tagging, named entity recognition, text categorization and question classification can all be formalized as a multi-class classification task.

After the hypothesis space H (a family of candidate learning targets) is decided, each hypothesis h in H is typically represented as a parameterized function over the feature vector. The most commonly-used performance measure is the (empirical) classification accuracy: given a set of labeled elements $S = \{x_i, p(x_i)\}_1^n$, the accuracy $acc_S(h, p) = \frac{1}{n} \sum_{x_i \in S} I(h(x) = p(x))$. In a supervised setting, supervision (e.g. the true class labels) is exploited in measuring this accuracy, and thus incorporated into the training process. The goal of training in classification is to seek the

best parameters for the function that can maximize the classification accuracy on the training set S.

One example of text-related classification tasks is that of learning a question classifier (Li & Roth, 2002). Recent works (Hovy, Gerber, Hermjakob, Lin, & Ravichandran, 2001; Moldovan, Pasca, Harabagiu, & Surdeanu, 2002) in open-domain question answering have shown that locating an accurate answer hinges on first filtering out a wide range of candidates based on some categorization of answer types given a question. For example, we hope to know that the question **Q:** *What Canadian city has the largest population?*, asks for a *city*; and **Q:** *What is a prism?*, asks for a *definition* of a "prism". This kind of semantic understanding and abstraction is performed by classifying a question into more than 50 semantic categories. (Li & Roth, 2002) attempts to learn a linear classifier with about $5,500$ manually labeled questions based on the SNoW learning algorithm. It exploits different types of syntactic features such as words and phrases, and semantic features such as named entities in a question.

Sparse Network of Winnows (SNoW)

SNoW [4] (Sparse Network of Winnows)(Roth, 1998; Carlson, Cumby, Rosen, & Roth, 1999) is a multi-class learning architecture that is specifically tailored for large scale learning tasks and will be applied to the entity identification task later in our approaches. It learns a two-layer sparse network of linear functions, where nodes in the first layer (feature nodes) represent the input features, and the nodes in the second layer (target nodes) represent the target classes, which are linear functions over a common feature space. The weights of the linear functions are stored on the links between the target nodes and feature nodes. The network is sparse in that a link only appear when the corresponding feature is active often enough given the target class in training examples.

SNoW is built on a feature efficient learning algorithm, Winnow (Littlestone, 1989) that is suitable for learning in NLP-like domains, where the number of potential features is very large, but only a few of them are active in each example, and only a small fraction of them are relevant to the

[4] available at http://l2r.cs.uiuc.edu/∼cogcomp/

target concept.

While SNoW is usually used as a classifier and predicts using a winner-take-all mechanism over the activation values of the target classes, the activation values can also help in estimating the posteriors of each class given the features. The raw activation value SNoW outputs is the weighted linear sum of the features. It can be verified that the resulting values are monotonic with the confidence in the prediction. Moreover, when it is trained to classify whether a pair names refer to the same entity, the activation can be converted into a similarity metric between the two names too.

SNoW has already been used successfully for a variety of tasks in natural language and visual processing (Golding & Roth, 1999; Roth, Yang, & Ahuja, 2000; Roth, Yang, & Ahuja, 2002).

1.2.2 Clustering

While classification is usually used as a supervised learning task – examples of all the classes and labels should occur in the training set to get a reasonable classification accuracy, clustering is always viewed as an unsupervised learning approach. Clustering is the task of partitioning a set of elements into a disjoint decomposition (*partition*) [5]. When supervision (e.g. class index of elements) is unavailable, the quality of a partition function, is measured with respect to the distance metric defined over the data space.

Clustering approaches have been widely applied to natural language processing (NLP) problems. Typically, natural language elements (words, phrases, sentences, etc.) are partitioned into non-overlapping classes, based on some distance (or similarity) metric defined between them, in order to provide some level of syntactic or semantic abstraction. A key example is that of class-based language models (Brown, deSouza R. Mercer, Pietra, & Lai, 1992; Dagan, Lee, & Pereira, 1999) where clustering approaches are used in order to partition words, determined to be similar, into sets. This enables estimating more robust statistics since these are computed over collections of "similar" words. A large number of different metrics and algorithms have been experimented

[5] Overlapping partitions will not be discussed here.

on these problems (Lee, 1999; Lee, 1997; Weeds, Weir, & McCarthy, 2004). Similarity between words was also used as a metric in (Pantel & Lin, 2002), which used it in a distributional clustering algorithm and to show that functionally similar words and can be grouped together and even separated to smaller groups based on their senses. At a higher level, (Mann & Yarowsky, 2003) disambiguated personal names by clustering people's home pages using a TFIDF similarity, and several other researchers have applied clustering at the same level in the context of the entity identification problem (Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2003; McCallum & Wellner, 2003; Li, Morie, & Roth, 2004a). Similarly, approaches to coreference resolution (Cardie & Wagstaff, 1999) use clustering to identify groups of references to the same entity.

Clustering methods for the most part occur as an optimization procedure that takes as input (1) a collection of domain elements along with (2) a distance metric between them and (3) an algorithm selected to partition the data elements, with the goal of optimizing some form of clustering quality with respect to the given distance metric. For example, the K-Means clustering approach (J. Hartigan, 1979) seeks to maximize the well-defined tightness of the resulting clusters based on the Euclidean distance. It is typically called an unsupervised method, since data elements are used without labels during the clustering process and labels are not used to provide feedback to the optimization process. E.g., labels are not taken into account when measuring the quality of the partition. However, in many cases, supervision is used at the application level when determining an appropriate distance metric (e.g., (Lee, 1999; Weeds, Weir, & McCarthy, 2004; Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2003) and more).

1.2.3 Probabilistic Model Estimation

In addition to discriminative approaches such as learning a feature-based function for classification and distance-based clustering algorithms, another major category of learning approaches for text understanding and mining tasks are estimation and inference with probabilistic models. Some characteristics of probabilistic modeling and inference are: (1) prior knowledge about the specific problem can be expressed as the prior probabilistic correlation between different events, or as

structures between a collective of related variables; (2) global inference can be made over this set of related variables or based on the structure underlying these variables; and (3) decisions made over classification and clustering generate some probabilistic semantic interpretation.

To introduce generative models and probabilistic model estimation, we assume some standard definitions from probability theory (Bickel & Doksum, 1977) and (Collins, 1999). If the set ζ is a discrete event space, and P is a probability distribution over this space, then (1) $0 \leq P(A) \leq 1$ for all $A \in \zeta$; (2) $\sum_{A \in \zeta} P(A) = 1$. In most examples the probability measure will be parameterized: i.e., P will also be a function of some parameters θ. Then the probability of event A given some parameter setting θ as $P(A\theta)$. The *parameter space* Ω is then the space $\{\theta | P(A|\theta)$ is a probability measure over $\zeta\}$. As an example, take the case of flipping a coin that can appear as either heads (H) or tails (T), where the probability of it producing as heads is p. In the case:

- The event space ζ is the set $\{H, T\}$.
- The set of parameters θ has a single element, p.
- The probability measure $P(A|\theta)$ is defined as p if $A = H$, $1 - p$ if $A = T$.
- The parameter space Ω is the set $[0, 1]$ (p must take some real value between 0 and 1 for $P(A|\theta)$ to be a probability measure).

Generative Models

A more example is the case of flipping multiple coins, corresponding to a generative mixture model.

Suppose we use two different coins to generate a flip. The probability of getting a head using coin I is p, while that using the coin II is q. For the flip, we first choose a coin with probability r to be coin I and $1 - r$ to be coin II, then we flip the chosen coin to produce a head or tail. In this case:

- The event space ζ is still the set $\{H, T\}$.
- The set of parameters θ now have three elements: p, q and r – the prior probability of choosing coin I.
- The parameter space Ω is the set $[0, 1]$ (p,q and r must take some real value between 0 and

1 for $P(A|\theta)$ to be a probability measure).

- The probability measure $P(A|\theta)$ is much more complex. $P(A|\theta)$ is $rp+(1-r)q$ if $A=H$, $r(1-p)+(1-r)(1-q)$ if $A=T$.

The distribution over A is a probabilistic mixture model since we use two individual models (two coins), with different head-generating probabilities p and q, to generate the outcome. The mixture parameters for the two models are r and $1-r$. When the probability of choosing each model is equally likely, that is, $P(choosingModeli) = 1/K(1 \leq i \leq K)$ for K models, the whole model becomes a *uniform* mixture model. Given the model – model parameters are known, we can make inference over complex events, e.g., what is the most likely outcome when generating four flips by repeating the above process.

In addition to inference, a more difficult problem is how to estimate model parameters θ given observations. Assuming we observe a sequence of n events $S =< x_1, x_2, \cdots, x_n >$, drawn from θ. For example, suppose somebody flips the one coin with an unknown probability $p = P(H|thiscoin)$ four times, and get $A = HTTT$. Can we estimate p according to this observation ? One commonly used solution to model parameter estimation is maximum likelihood estimation.

Maximum Likelihood Estimation

The maximum likelihood estimation (Dempster, Laird, & Rubin, 1977; Collins, 1999) seeks the parameter $\widehat{\theta_{ML}}$ that can generate the observation with the highest probability. Assuming that the individual events are independent of each other, the likelihood function, L, is defined as

$$L(S|\theta) = \prod_{i=1\cdots n} P(x_i|\theta). \tag{1.1}$$

The maximum likelihood estimate $\widehat{\theta_{ML}}$ is the parameters in Ω that maximizes this likelihood function:

$$\widehat{\theta_{ML}} = argmax_{\theta\in\Omega} L(S|\theta). \tag{1.2}$$

In the coin example, the likelihood of the sample $S =< HTTT >$ is

$$L(S|\theta) = p(1-p)^3$$

and the maximum likelihood estimate of p is

$$\widehat{p} = argmax_{p\in[0,1]} p(1-p)^3 = \frac{1}{4}$$

Maximum likelihood estimation can be easily computed in many cases when the class labels (the index to the individual model used to generate an event) of data elements are observed and given. I.e., it is known that which coin has been used to generate a flip. For complex mixture models, where the models that generating the a sequence of observations are unknown, the Expectation-Maximization (EM) algorithm is generally used to perform maximum likelihood estimation.

EM algorithm

The EM algorithm is typically viewed as an approach for learning a probabilistic mixture model in an unsupervised setting (Dempster, Laird, & Rubin, 1977; McLachlan & Krishnan, 1997; Friedman, 1998). The core idea is to iteratively seek model parameters that achieve a local optimum of the expected log-likelihood of a set of observed elements when the data is incomplete or has missing values.

Suppose a probabilistic mixture model (e.g., Gaussian mixture model), parameterized with $\theta \in \Omega$, defines a probability distribution P over $X \times C = \{1, 2, \cdots, K\}$. That is, the probability of each data element $x \in X$ belongs to a class $p(x) \in C$ is associated with a probability $P(x, p(x) = i|\theta)$. The complete log-likelihood of generating any set of labeled elements $(S, P(S)) = \{x_i, p(x_i)\}_1^n$ from $X \times C$ (i.i.d. sampled), is defined as:

$$LL(S, P(S)|\theta) = log[\Pi_{x\in S} P[x_i, p(x_i)|\theta]] \tag{1.3}$$

where $p(S)$ is a partition of S and $p(x_i) \in C$ is the corresponding class index of x_i. Suppose S is observed but the true labeling $p(S)$ is unknown, the standard EM attempts to estimate θ^* in which the expected log-likelihood is optimized:

$$\theta^* \equiv \underset{\theta \in \Omega}{\operatorname{argmax}} \, E[LL(S, p(S)|\theta)] \tag{1.4}$$

Instead of the expected log-likelihood directly, the practical EM algorithm chooses to iteratively update model parameters, to optimize an intermediate function. A Q function is defined to relate $\theta^{(i-1)}$ – the current parameters estimates, and θ – the new parameters to be optimized to increase Q:

$$Q(\theta, \theta^{(i-1)}) = E[logPr(S, P(S)|\theta)|S, \theta^{(i-1)}] \tag{1.5}$$

After randomly initializing the model parameters to be θ^0, the practical EM algorithm iterates over the following two steps: the E-step (Expectation) which evaluates ($Q(\theta, \theta^{(i-1)})$); and the M-step (Maximization) which seeks the new parameters that can maximize the expectation, s.t.,

$$\theta^{(i)} = argmax_{\theta \in \Omega} Q(\theta; \theta^{(i-1)}) \tag{1.6}$$

Each iteration is guaranteed to increase the expected log-likelihood, and thus the algorithm will converge to a local maximum of the likelihood function.

1.3 Thesis Contribution

This thesis systematically studies the fundamental problem toward concept-based text understanding and mining — identifying whether ambiguous names in text, within and across documents, refer to the same real-world concept. We study and propose different machine learning techniques to address different aspects of this problem and show that as more information can be exploited, the learning techniques developed accordingly, can continuously improve the identification accuracy.

Our first model is a discriminative approach that models the problem as deciding whether any two names mentioned in a collection of documents represent the same entity. This straightforward modelling of the problem results in a classification problem – as has been done by several other authors (Bilenko & Mooney, 2003; Cohen, Ravikumar, & Fienberg, 2003a) – allowing us to compare our results with these. This is a standard pairwise classification task, under a supervised learning protocol; our main contribution in this part is to show how relational – string and token-level features – and structural features, representing transformations between names, can significantly improve the performance of this classifier. A similarity metric between different types of names can be directly induced from the pairwise classification and being applied to global identification over a set of names.

Several attempts have been made in the literature to improve the results by performing some global optimization (i.e. clustering), with the above mentioned pairwise classifier as the similarity metric. The result of these attempts were not conclusive and we provide some explanation for why that is. We prove that, assuming optimal clustering, clustering reduces the error of a pairwise classifier in the case of two classes (corresponding to two entities); however, in the more general case, when the number of entities (classes) is greater than 2, global optimization mechanisms could be worse than pairwise classification. Our experiments concur with this proof.

We further analyze the setbacks of these clustering approaches, and propose our second approach – a new clustering framework, to resolve them. Clustering is an optimization procedure that partitions a set of elements to optimize some criteria, based on a fixed distance metric defined between the elements. It has been shown repeatedly that its success depends on defining a good distance metric – one that is appropriate for the task and the clustering algorithm used. Many recent works have made contributions in the direction of automatically learning a metric with supervision, but suffer some limitations. In order to resolve them, in this work we develop a unified framework for clustering, guided by supervision. The proposed supervised discriminative clustering framework (SDC) targets learning a partition function, parameterized by any chosen clustering algorithm, to minimize the clustering distortion from given supervision. A general learning al-

gorithm is also developed under this framework, that can be used to learn an expressive distance function over a feature space. Moreover, a theoretical and empirical study compares SDC with multiple variants of EM and existing metric learning approaches. Our experiments on entity identification task show that SDC which trains a similar metric for a chosen clustering can significantly outperforms existing clustering approaches, and other metric learning approaches where clustering is disjoint from the metric learning procedure.

This observation motivates our third approach. We develop a global probabilistic model for *Entity Identification*, 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, so that a document that mentions "President Kennedy" is more likely to mention "Oswald" or " White House" than "Roger Clemens"; (2) an "author" model, that makes sure that at least one mention of a name in a document is easily identifiable (after all, that's the author's goal), and then generates other mentions via (3) an appearance model, governing how mentions are transformed from the "representative" mention. This work presents the first study of our proposed model and several relaxations of it. Our goal is to learn the model from a large corpus and use it to support *Entity Identification in Text*. Given a collection of documents we learn the model in an *unsupervised* way; that is, the system is not told during training whether two mentions represent the same entity.

In our experimental study we evaluate different models on the problem of the cross-document identification for three entity types: People (Peop), Locations (Loc) and Organizations (Org). Our experimental results are somewhat surprising; we show that the unsupervised approach can solve the problem accurately, giving accuracies (F_1) around 90%, and better than our discriminative classifier (obviously, with a lot more data).

In addition, we extend our global probabilistic model to address a significant application – semantic integration between text and databases. Many real-world applications increasingly involve a large amount of both structured data and text. The reason is two-folded. First, certain kinds of information are best captured in structured data, and other kinds in text. Second, the informa-

tion required for the application may need to be assembled from many sources, some of which contribute to structured data, and others to 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.

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.

Based on the work of globally identifying real-world entities from a large collection of documents (for example, everyday news articles or the whole set of online web pages), our ultimate goal is to design and implement a unified framework for intelligent access of textual information. For this purpose, we perform a case of applying concept-level information in search engines and show promising perspective of concept-based text understanding and mining.

1.4 Outline of this Thesis

The rest of this thesis is organized as follows:

Chapter 2 gives a detailed introduction to the problem of learning a metric to capture the similarity between names of different types of entities and describes our approach in doing that which is based on pairwise classification with local features of names, both relational and structural fea-

tures. It also studies some natural clustering approaches that are built on the similarity metric learned in this setting.

Chapter 3 introduces our new, supervised discriminative clustering framework, as well as a general learning algorithm to train a metric for any chosen clustering algorithm (i.e. K-means, single-linkage and so on), guided by supervision given in the context of a given task. We study this framework theoretically by comparing with the EM algorithm, and empirically by comparing it with existing approaches both on some generated data, and on the real corpus of entity identification.

In Chapter 4, we design a generative probabilistic model for the same problem and show how to learn the three models, which are different relaxations to the basic model, in a completely unsupervised setting. In addition, we compare the pairwise classification and clustering approaches with the generative models, and further analyze the difference between them.

Chapter 5 studies an important application of our generative approach to the problem of semantic integration across text and databases.

In the end, Chapter 6 presents our thoughts about what concept-based text understand and mining would look like and performs a case study of exploiting concept-level information to help search engines. It also introduces the future steps beyond resolving the name ambiguity and summarize this thesis.

Chapter 2

Learning to Measure Name Similarity

Entity identification in text is the task of identifying whether names, within and across documents, refer to the same real-world concept. One type of information that is very critical to this problem is the appearance similarity between names, that is, whether two names themselves are similar or not without considering other information. Although sometimes the same name, like "Kennedy", could refer to different entities in different contexts of the text, the general intuition is still that similar names tend to refer to the same entity, while different names tend to refer to different entities. For this reason, we study in this chapter the influence of appearance similarity over entity identification.

Most prior work (Durban, Eddy, Krogh, & Mitchison, 1998; Monge & Elkan, 1996a; Jaro, 1995; Jaro, 1989; Winkler, 1999) in this direction focus on manually designing "good" string- or token-based metrics for measuring the appearance similarity between names, such as edit distance. (Cohen, Ravikumar, & Fienberg, 2003a) 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 Jaro-Winkler string-distance scheme. Although this is a fixed scheme, the threshold used by the SoftTFIDF classifier is trained. Machine learning techniques (Bilenko & Mooney, 2003; Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2003; Ristad & Yianilos, 1998) have been recently applied to automatically learning a metric based on a set of features extracted from the names, that are informative in measuring the similarity or difference. In most cases, a parameterized simi-

larity function is trained given annotated examples (i.e., pairs of names labelled as matching or non-matching entities). However, different learning-based similarity metrics may vary in their selection of features, parameterizations and learning algorithms. A pairwise classifier that can tell whether any two names refer to the same entity can be constructed naturally by thresholding the similarity between them.

Our first approach (Li, Morie, & Roth, 2004b) for entity identification is a discriminative approach following this general direction, that models the problem as that of deciding whether any two names mentioned in a collection of documents represent the same entity, based on the appearance similarity between the two names. This straightforward modelling of the problem results in a classification problem – as has been done by several other authors (Cohen, Ravikumar, & Fienberg, 2003a; Bilenko & 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. A similarity function between names can be naturally induced from the confidence of the prediction of the pairwise 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 attempting to use 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 – e.g., to groups of documents originated at about the same time period – prevents some of these problems and aids clustering significantly.

In Section 2.1, we further discuss and summarize various fixed metrics for measuring string and name similarity. In Section 2.2, we describe one learning technique that can be applied to acquiring

adaptive similarity metrics for each particular task, which is based on pairwise classification of names. The learned similarity metric is then applied to a clustering task in Section 2.3, that can perform some global optimization over entity identification – partitioning a set of names altogether, according to the entities.

2.1 Measuring Name Similarity

Each entity name can be viewed as a string or a token vector. Due to this difference in representation, the similar metrics between names can be categorized into three types: (1) string-based similarity metrics; (2) token-based similarity metrics; and (3) hybrid metrics of strings and tokens.

One of the most common string-level similarity metrics (Cohen, Ravikumar, & Fienberg, 2003a) is edit distance:

Definition 2.1.1 *Consider any two names x_1, x_2 as sequences of characters $< c_1 c_2 \cdots c_n >$ and $< c'_1 c'_2 \cdots c'_m >$. A few transformation operations from one name to another as follows:* ***delete*** *– delete a character from x_1;* ***insert*** *– insert a character to some position in x_1; and* ***substitute*** *– change a letter to a different one. Any string can be transformed to another string through a sequence of these operations $Q(x_1) =< op_1, op_2, \cdots, op_t >$ and Q is not unique for most pairs of strings. The cost of Q is typically defined as $|Q| = t$, where t is the number of operations performed. The edit distance $dist_{ed}(x_1, x_2)$ between x_1 and x_2 is defined as:*

$$dist_{ed}(x_1, x_2) \equiv min_{|Q|} Q(x_1) = x_2. \tag{2.1}$$

The standard edit distance – *Levenstein distance*, assigns a unit cost to all edit operations and is symmetric between two strings. That is, $dist_{ed}(x_1, x_2) = dist_{ed}(x_2, x_1)$ for any x_1, x_2. There is an efficient recursive procedure to compute $dist_{ed}(x_1, x_2)$. Let $D(x_1, x_2, i, j)$ denote the edit distance

between the first i letters in x_1 and the first j letters in x_2. Then we have,

$$D(x_1, x_2, 0, 0) = 0; \tag{2.2}$$

and

$$D(x_1, x_2, i, j) = \begin{cases} D(x_1, x_2, i-1, j-1) & \text{if } c_i = c'_j \text{ and you copy } c_i \text{ to } c'j; \\ D(x_1, x_2, i-1, j-1) + 1 & \text{if letter } t_j \text{ is substituted for } s_j; \\ D(x_1, x_2, i, j-1) & \text{if letter } t_j \text{ is inserted;} \\ D(x_1, x_2, i-1, j) & \text{if letter } s_i \text{ is deleted.} \end{cases} \tag{2.3}$$

Some variations of edit distance metrics are Smith-Waterman (Durban, Eddy, Krogh, & Mitchison, 1998) distance and Monger-Elkan distance (Monge & Elkan, 1996a), which adopt particular but non-uniform cost parameters for different editing operations. In the record-linkage domain, a broadly used similar metric is the Jaro metric (Jaro, 1995; Jaro, 1989), which is not based on an edit distance model. Instead, it is defined over the number and order of the common characters between two strings. A variant of Jaro metric was proposed by Winkler in (Winkler, 1999).

There is another set of distance metrics which treat each name as a sequence of tokens. For example, the name "John F. Kennedy" is represented as $<' John', ' F.', ' Kennedy' >$. The distance metrics are defined based on the common tokens inside two names, and the tokens are weighted differently according to some statistical models like TFIDF (Salton, 1988). Some other metrics like Jensen-Shannon distance (Borovkov, 1984) assume that there are some underlying probability distribution over tokens and measure the probability of two names matching based on it.

The third category of distance metrics are hybrids of the previous methods. Unlike the token-level metrics, which ignore different but similar tokens in two names, they count on these tokens in the measurement. For example, although "University of Illinois" and "University of Ill." have different tokens, the hybrid metrics still consider these two words as matching, due to the similarity

between “Illinois” and “Ill.”. The state-of-art distance metric evaluated in (Cohen, Ravikumar, & Fienberg, 2003a) is a hybrid scheme (SoftTFIDF) combining a TFIDF weighting scheme of tokens with the Jaro-Winkler string-distance scheme.

2.2 Adaptive Distance Metrics

A good distance (similarity) metric between names is one in which close proximity correlates well with the likelihood of being in the same class when applied in the entity identification task. A lot of recent works (Winkler, 1999; Cohen & Richman, 2002b; Cohen, Ravikumar, & Fienberg, 2003b; Cohen, Ravikumar, & Fienberg, 2003a) formalize the entity identification as a pairwise classification as follows:

Definition 2.2.1 *The goal of entity identification is to seek a pairwise function $f : X \times X \rightarrow \{0, 1\}$ which classifies two strings (representing entity writings) in the name space X, as to whether they represent the same entity (1) or not (0).*

A direct solution to this problem is to build a pairwise classifier through thresholding the name similarity between two names:

$$f(x_1, x_2) = 1 \iff d(x_1, x_2) \leq T \tag{2.4}$$

where T is a threshold and $T \geq 0$.

The classification accuracy for a classifier can be computed over all pairs of names in a test set and is used to evaluate the “goodness” of a similarity metric. Many experiments (Cohen, Ravikumar, & Fienberg, 2003b; Cohen, Ravikumar, & Fienberg, 2003a; Ristad & Yianilos, 1998) have shown the distance metrics suitable for matching names in different domains could be very different from each other. Even the distance metric performs the best in one data set could perform much worse in another data set of a similar but different domain. This is because the writing style of names tend to follow different rules in various domains and situations. For example, person

names in the news articles of different news agencies might have different forms.

Machine learning techniques (Bilenko & Mooney, 2003; Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2003; Ristad & Yianilos, 1998) have been recently applied to automatically learning a metric in the context of a specific domain. These works have shown that the learned distance metrics in a specific domain tend to outperform the fixed metrics. For example, (Ristad & Yianilos, 1998) provides a stochastic model for the string edit distance. Each editing operation is associated a probability obtained from a learning process. A global model defines the probability of a string being transformed from another. In the application of learning the pronunciation of words in conversational speech, the learned edit distance has only about one fifth the error rate of the fixed Levenstein distance. (Bilenko & 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 classifier (SVM) against fixed distance measures (but not the one mentioned above) and showed some improvements in performance.

We propose a learning approach, *LMR* [1], 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 SNoW (Sparse Network of Winnows (Carlson, Cumby, Rosen, & Roth, 1999) as described in Section 1.2.1) learning architecture. A feature extractor[2] automatically extracts features in a data-driven way for each pair of names. Our decision is thus of the form:

$$f(x_1, x_2) = \arg\max_{c \in \{0,1\}} f^c(x_1, x_2) = \arg\max_{c \in \{0,1\}} \sum_{i=1}^{m} w_{i,c} f_i \tag{2.5}$$

where $w_{i,c}$ is the weight of feature $f_i (1 \leq i \leq m)$ in the function $f^c(x_1, x_2)$.

[1] Named after the initials of the designers' last names.

[2] We use FEX, a feature extractor tool available from http://L2R.cs.uiuc.edu/∼cogcomp/cc-software.html.

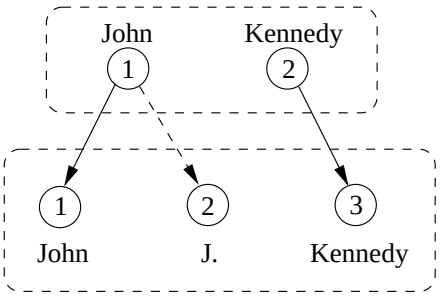

Figure 2.1: **An example of feature extraction.** There are two possible features (Equality and Initial) for token one in the smaller partition but only the higher priority Equality feature is activated.

2.2.1 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 modelled as a partition in a bipartite graph, with each token in that name as a vertex (see Figure 2.1) 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, it 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 only activate one feature. If a particular token in the smaller bipartition did not activate any features, then a "null" feature is activated for that token. We define thirteen types of token-based features, shown in the priority order as described above. See Figure 2.1.

Relational features are not sufficient, since a non-matching 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 in the relative order of tokens between the

Honorific Equal	active if both tokens are honorifics and identical.
Honorific Equivalence	active if both tokens are honorifics, not identical, but equivalent.
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.
Prefix Equality	active if the prefixes of both tokens are equal.
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.
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.
Symbol Map	active if one token is a symbolic representative of the other.
Structural	recording the location of the tokens that generate other features in two names.

Table 2.1: **Features employed by LMR and SDC.**

two names, by recording the location of the participating tokens in the partition. This results in a more expressive feature set, because the same feature activated by two sets of tokens with different relative positions can be distinguished from each other. E.g., 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 "$(1, 2, \emptyset)$".

2.2.2 Experiments

In our experimental study we evaluated different models on the problem of Entity Identification for three entity types – People (Peop), Locations (Loc) and Organizations (Org). The document segments shown in Figure 2.2 exemplify the preprocessed data given as input to the evaluation. The learning approaches were evaluated on their ability to determine whether a pair of entities (within or across documents) actually correspond to the same real-world entity.

We collected $8,000$ names from 300 randomly sampled 1998-2000 New York Times articles in the TREC corpus (Voorhees, 2002). These include about $4,000$ personal names[3], $2,000$ locations

[3]Honorifics and suffixes like "Jr." are considered part of a personal name.

and $2,000$ organizations. The documents were annotated by a named entity tagger [4]. The annotation was verified and manually corrected if needed and each name mention was labelled with its corresponding entity by two annotators. The distribution of mentions and entities in the above corpus as to the number of mentions refer to each entity is given by Figure 2.3. Tests were done by averaging over five pairs of sets, each containing 600 names, that were randomly chosen from the $8,000$ names.

Document 1: ***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 2: ***David Kennedy*** *was born in* ***Leicester****,* ***England*** *in 1959.? · · ·* ***Kennedy*** *co-edited The New Poetry (Bloodaxe Books 1993), and is the author of New Relations: The Refashioning Of* ***British*** *Poetry 1980-1994 (Seren 1996).?*

Figure 2.2: **Segments from two documents preprocessed by our named entity tagger.** Different types of entities are annotated with different grey scales. As shown, similar mentions within and across documents may sometimes correspond to the same entities and sometimes to different entities.

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 co-referring 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 co-refer, to avoid excessive negative examples in training sets, we adopt a ratio of $10:1$ between negative examples and positive examples.

The results in all the experiments in this chapter 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 (in-

[4]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.

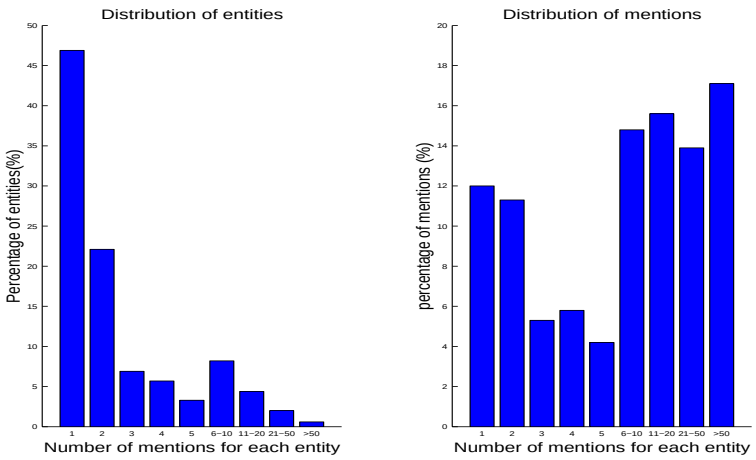

Figure 2.3: **Distribution of mentions and entities in different groups.** This data set has about $8,000$ mentions corresponding to $2,000$ entities. Mentions and Entities are partitioned into groups according to the number of mentions referring to an entity. The X-axis shows how many mentions of an entity in each group.

cluding non-matching pairs). Since most pairs are trivial negative examples, and the classification accuracy can always reach nearly 100%, 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:

$$P = \frac{|M_p \bigcap M_a|}{|M_p|} \qquad R = \frac{|M_p \bigcap M_a|}{|M_a|},$$

and summarized by

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

Only F_1 values are shown and compared in this paper.

Figure 2.4 presents the average F_1 for three different pairwise classifiers on the five test sets described in Section 2.2.2. The LMR classifier outperforms the SoftTFIDF classifier and the Marlin classifier when trained and tested on the same data sets.

Figure 2.5 shows the contribution of different feature types to the performance of the LMR

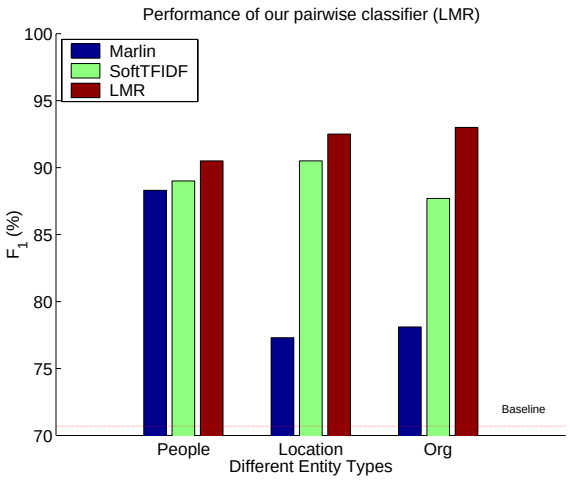

Figure 2.4: **Performance of different pairwise classifiers.** Results are evaluated using the F_1 value and are averaged over five test sets of 600 names each, for each entity type. The learned classifiers are trained using corresponding training sets with 600 names. The baseline performance in the experiment is 70.7% given by a classifier that predicts only identical names as positive examples, and it is averaged over the three entity types.

classifier. The *Baseline* classifier in this experiment only makes use 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 features types is very significant, and more so to *People* due to a larger amount of overlapping tokens between entities.

2.3 Clustering Using Similarity Metrics

Clustering methods are used for the most part as an optimization procedure in many areas, such as language processing (Pantel & Lin, 2002; Weeds, Weir, & McCarthy, 2004), computer vision (Jain, Murty, & Flynn, 1999; Shi & Malik, 2000) and data mining (Bradley, Fayyad, & Reina, 1998), and have been widely studied in the AI community (Kamvar, Klein, & Manning, 2002; Vilalta & Rish, 2003). Clustering takes as input (1) a collection of domain elements along with (2) a distance metric between them and (3) an algorithm selected to partition the data elements, with the goal of optimizing some form of clustering quality with respect to the given distance metric. For example,

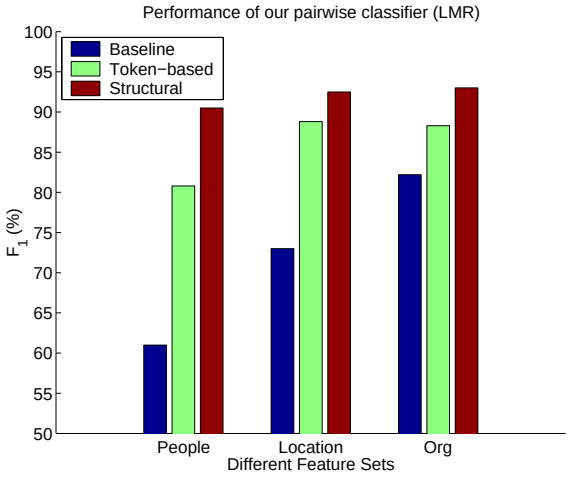

Figure 2.5: **Contribution of different feature sets.** The LMR classifier is trained with different feature sets using the five training sets. Results are evaluated using the F_1 value and are averaged over the five test sets for each entity type with 600 names in each of them. The *Baseline* classifier only uses string-edit-distance features and "Equality" features. The *Token-Based* classifier uses all relational token-based features while the *Structural* classifier uses, in addition, structural features.

seeks to maximize a well defined notion of the tightness of the resulting clusters, defined based on the Euclidean distance.

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, & Fienberg, 2003a; Cohen & Richman, 2002b; McCallum & 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.

2.3.1 Definitions of Clustering

Clustering is the task of partitioning a set of elements $S \subseteq X$ into a disjoint decomposition (*partition*)[5] $p(S) = \{S_1, S_2, \cdots, S_K\}$ of S. We associate with it a *partition function* $p = p_S : X \to C = \{1, 2, \ldots K\}$ that maps each $x \in S$ to a class index $p_S(x) = k$ iff $x \in S_k$. We will omit the subscript S in p_S and $p_S(x)$ when clear from the context. Notice that, unlike a classifier, the image $x \in S$ under a partition function depends on S.

A typical clustering algorithm views data points $x \in X$ to be clustered as feature vectors $x = (x_1, x_2, \ldots, x_m)$ in a m-dimensional feature space. From a generative perspective, the observed data points $S = \{< x_i, p(x_i) >\}_1^n$ are sampled i.i.d. from a joint probability distribution P defined over $X \times C$ (P is a mixture of K models). This distribution gives the sampling probability of a data point $(x, p(x))$ – $P(x, p(x) = i)(1 \leq i \leq K)$. A distance (equivalently, a similarity) metric d is commonly used in clustering to measure the proximity between two elements is a pairwise function $X \times X \to \mathbb{R}^+$.

2.3.2 Is Clustering Always Better Than Pairwise Classification ?

We now compare clustering with pairwise classification theoretically in the case of Gaussian mixture models. In the following definitions we assume that $(x_1, p(x_1)), (x_2, p(x_2)) \in X \times C$ are sampled i.i.d according to it, with x_1, x_2 observed and $p(x_1), p(x_2)$ hidden.

Definition 2.3.1 *The problem of Entity Identification is that of finding a function $f : X \times X \to \{0, 1\}$ which satisfies:*

$$f(x_1, x_2) = 1 \quad \textit{iff} \quad p(x_1) = p(x_2) \tag{2.6}$$

Definition 2.3.2 *Let $d : X \times X \to \mathbb{R}^+$ be a distance metric, and $T \geq 0$ is a constant threshold. The* pairwise classification *function f_p in this setting is defined by:*

$$f_p(x_1, x_2) = 1 \quad \textit{iff} \quad d(x_1, x_2) \leq T. \tag{2.7}$$

[5]Overlapping partitions will not be discussed here.

The clustering based decision f_c is defined by:

$$f_c(x_1, x_2) = 1 \quad \textit{iff} \quad argmax_i P\{x_1 | p(x_1) = i\} = argmax_i P\{x_2 | p(x_2) = i\}. \tag{2.8}$$

Definition 2.3.3 *Define $I(x_1, x_2)$ to be 1 when $p(x_1) = p(x_2)$ and 0 otherwise. The* error rate *of the function $f : X \times X \to \{0, 1\}$ is defined as:*

$$err(f) = E(P\{f(x_1, x_2) \neq I(x_1, x_2)\}) \tag{2.9}$$

where the expectation is taken over independent samples, according to $P_{X \times C}$, of pairs of points $\{(x_1, p(x_1)), (x_2, p(x_2))\}$.

The possible error cases of pairwise classification and clustering in a simple setting are shown in Figure 2.6.

Theorem 2.3.1 *Assume data is generated according to a uniform mixture of K Gaussians $G = \{g_1, g_2, \ldots, g_K\}$ 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 sampling according to the i-th Gaussian chosen. Suppose further that the clustering algorithms yields the correct K Gaussian distributions; then, $\forall$ threshold $T \geq 0$, if $K = 2$ then*

$$err(f_c) < err(f_p). \tag{2.10}$$

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

Proof 2.3.1 ***(sketch)***:[6] *It is easy to see that the probability density over tuples in $X \times C$ is $f(x, p(x) = i) = \frac{1}{K} * g_i(x)$. The error rates $err(f_c)$ and $err(f_p)$ can be computed using the density function. Thus, for $K = 2$, we get $err(f_c) = \frac{1}{2} - \frac{1}{2}A^2$, where $A = \int_{\mathcal{R}} [g_2(x) - g_1(x)]dx$,*

[6]The assumption of Gaussian distributions can be relaxed.

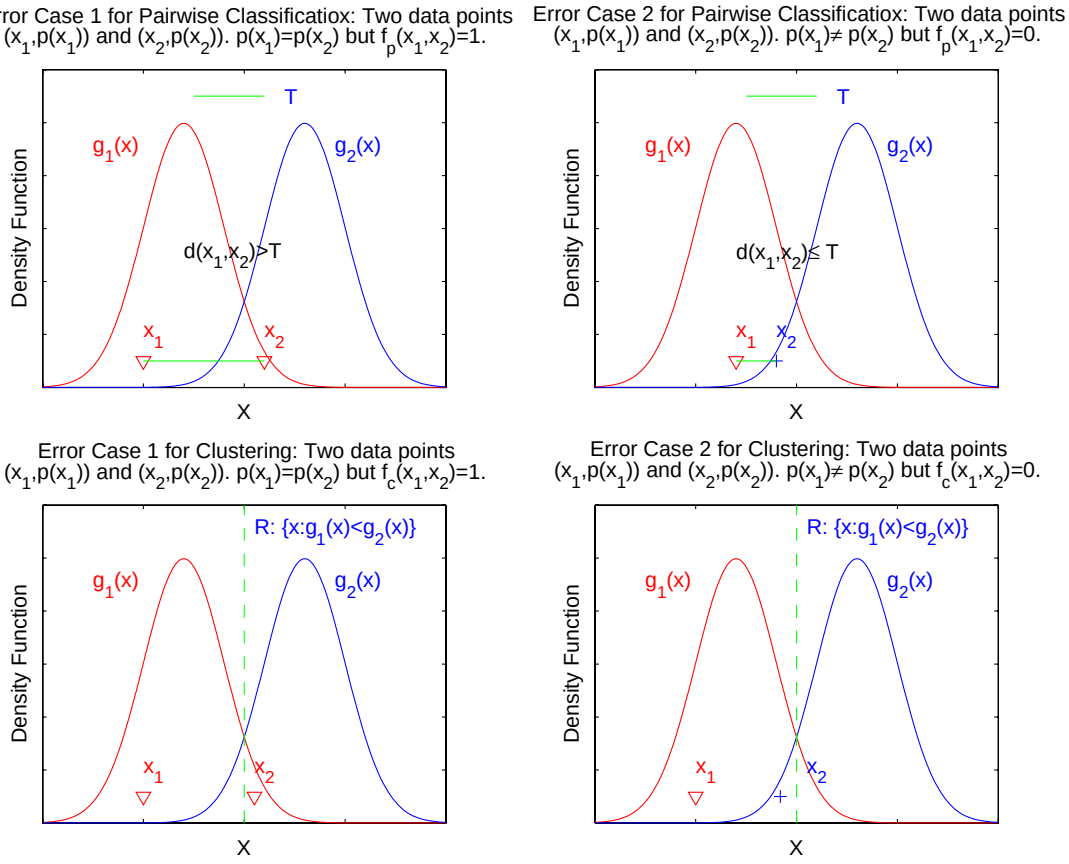

Figure 2.6: **Error cases of pairwise classification and clustering (for a uniform mixture of two Gaussian generative models)**. X is a one-dimensional data space. $(x_1, p(x_1))$ and $(x_2, p(x_2))$ are two data points with their class labels. We have two classes in this case, their density functions over the data space satisfy Gaussian models with the same variance and correspond to g_1 and g_2. T is the threshold used by the pairwise classifier. f_p and f_c are the decision functions of pairwise classification and clustering respectively.

and $\mathcal{R}$ is the area in the feature space that satisfies $g_2(x) > g_1(x)$ $(x \in X)$. $\mathcal{R}$ is a half space here.

We also have:

$$err(f_p) = \quad \tfrac{1}{2} - \tfrac{1}{2}\int_{\mathcal{R}}[g_2(x_1) - g_1(x_1)]dx_1 \times$$
$$\int_{M(x_1,T)}[g_2(x_2) - g_1(x_2)]dx_2 \; > \; err(f_c)$$

where $M(x_1, T)$ is the sphere area in the data space X whose element x satisfies $d(x_1, x) \leq T$. This is so since $\int_{M(x_1,T)}[g_2(x_2) - g_1(x_2)]dx_2 < \int_{\mathcal{R}}[g_2(x_2) - g_1(x_2)]dx_2$.

For $K > 2$, we can compute $err(f_c)$ and $err(f_p)$ in a similar way and compare them. We found that in this case, each can be smaller than the other in different cases, depending on the configuration of the K Gaussians, for example, the distances between the centers of these Gaussians. Specifically, when $\sum_{i=1}^{K} g_i(x) > 2 \cdot g_j(x)$ for all j and $x \in X$, and $T \to 0$, we have $err(f_c) > err(f_p)$. ■

2.3.3 Entity Identification with Clustering

To study this issue experimentally, we designed and compared several clustering schemes for the *Entity Identification* 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 x_1 and x_2; then, $sim(x_1, x_2) = \frac{e^p}{e^p + e^n}$.

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 generate 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 complete-link. 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 2.7, shows a 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. After that, we first cluster names belonging to each group, then choose a representative for the names in each cluster and, hierarchically, cluster these representatives across groups into final clusters. The

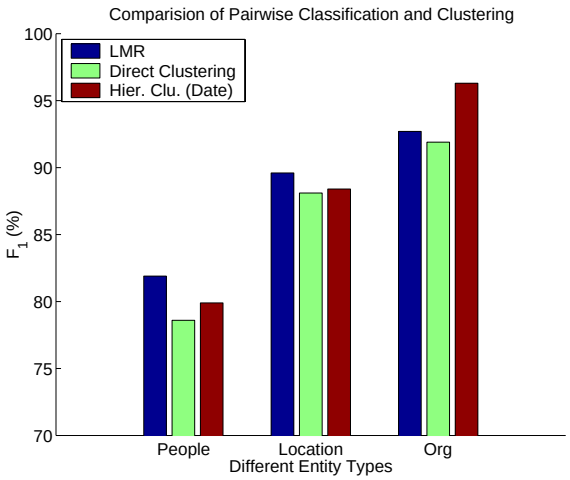

Figure 2.7: **Best performance of different clustering approaches** (Various parameter settings, including different numbers of clusters were experimented with in direct clustering and the hierarchical clustering.) 'LMR' represents our pairwise classifier. It is compared with different clustering schemes, based on it as a similairty metric. Results are evaluated using F_1 values. The test set has 900 names for each entity type.

complete-link 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 set. 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.

2.3.4 Discussion

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 they refer to 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 Entity Identification problem necessitates better exploitation of supervision in training a local similarity metric, and better exploitation of global and structural aspects of data. Our supervised clustering framework in Chapter 3 was designed to address the former issue of applying supervision to clustering, while the generative model in Chapter 4 was developed to exploit structural information of documents.

Chapter 3

Supervised Discriminative Clustering

Clustering is typically called an unsupervised method, since data elements are used without labels during the clustering process, or labels are not taken into account when measuring the quality of the partition in the optimization process. What's more, when clustering with a given algorithm and a fixed metric, one makes some implicit assumptions, perhaps unintended, on the data and the task (e.g., (Kamvar, Klein, & Manning, 2002) and more on that below), which may not hold in reality[1]. This scenario, however, has severe drawbacks, such as potential disparity from one's intention, and a lack of flexibility due to a fixed distance metric and algorithm.

Several works (Cohen, Ravikumar, & Fienberg, 2003a; Cohen & Richman, 2002b) have attempted to remedy these problems by learning a domain-specific metric. Other works (Bach & Jordan, 2003; Bar-Hillel, Hertz, Shental, & Weinshall, 2003; Schultz & Joachims, 2004; Xing, Ng, Jordan, & Russell, 2002; Mochihashi, Kikui, & Kita, 2004; Bilenko, Basu, & Mooney, 2004) have also pursued this general direction, and some have tried to learn a metric with a limited amount of supervision, no supervision, or by incorporating other information sources such as constraints on the class memberships of data elements. They have shown significant performance improvement over traditional clustering in various tasks. Most of these approaches, though, suffer a variety of limitations (compared and analyzed later). For example, (Bach & Jordan, 2003; Bilenko, Basu, & Mooney, 2004) can only learn a metric for one specific clustering algorithm, such as spectral

[1] For example, the optimal conditions under which K-Means works occur when the data is generated from a uniform mixture of Gaussian models in the assumed metric space.

clustering or K-Means respectively.

In this thesis we develop a unified framework for clustering that is guided by supervision. Our framework provides a way to exploit supervision in the metric learning problem and do it in a way that is parameterized by any chosen clustering algorithm. The proposed framework, Supervised Discriminative Clustering (SDC), provides a unified perspective to address an important problem, several aspects of which have been addressed previously. In particular, this view allows us to develop some theoretical understanding of the problem, relate supervised clustering to existing algorithms and suggest variations of them.

In SDC, clustering is explicitly defined as a learning problem in the context of a given task. The training stage is formalized as an optimization problem in which a partition function is learned in a way that minimizes a clustering error. The clustering error is well-defined and driven by feedback from labeled data. Training a distance metric with respect to any given clustering algorithm seeks to minimize the clustering error on training data that, under standard learning theory assumptions, can be shown to imply small clustering error also in the application stage. One distinctive property of this framework is that the metric is learned in an algorithm-specific way and any clustering algorithms (e.g. K-Means, agglomerative clustering, and spectral clustering), that rely on distance between data elements, can be applied. A general learning algorithm is also developed under this framework, that can be used to learn an expressive distance function over a feature space (e.g., it can make use of kernels). While this approach makes explicit use of labeled data, we argue that, in fact, many clustering applications also make use of this information off-line, when exploring which metrics are appropriate for the task. Our framework makes better use of this resource by incorporating it directly into the metric training process; training is driven by true clustering error, computed via the specific algorithm chosen to partition the data.

Motivated by the same idea, we also consider integrating metric learning into the traditional EM framework (McLachlan & Krishnan, 1997) for probabilistic models, but with two extensions: (1) rather than learning probabilistic models, we directly learn a partition function, parameterized by a metric and a chosen clustering algorithm; and (2) we exploit labeled data in parameter estimation.

A Supervised EM* algorithm is then developed to learn a distance function in an iterative way, guided by supervision.

Further analysis shows that SDC and Supervised EM* are not only discriminative and probabilistic approaches respectively, motivated by the same idea, but also equivalent in some case – that of learning a metric for a data space satisfying a uniform mixture of Gaussian models. In our empirical study, SDC exhibits significant improvement (over 20% error reduction) over traditional clustering, on both an artificial data set and a real task of matching names.

The rest of this chapter first discusses the problem of metric learning in clustering and summarizes related works in Section 3.1, and then introduces the supervised discriminative clustering framework in Section 3.2 and some variants of the EM algorithm that handle metric learning and supervision 3.3. This section also involves a comparison between them through both theoretical analysis and empirical simulation in the case of Gaussian mixture models. Finally, we apply the supervised discriminative approach to the entity identification problem in Section 3.4 and summarize this chapter in Section 3.5.

3.1 Metric Learning in Clustering

As we mentioned before, clustering is the task of partitioning a set of elements $S \subseteq X$ into a disjoint decomposition (*partition*)[2] $p(S) = \{S_1, S_2, \cdots, S_K\}$ of S. We associate with it a *partition function* $p = p_S : X \rightarrow C = \{1, 2, \ldots K\}$ that maps each $x \in S$ to a class index $p_S(x) = k$ iff $x \in S_k$. In practice, we use a clustering algorithm $\mathcal{A}$, (e.g. K-Means), and a distance metric d, (e.g., Euclidean distance) to generate a function h to approximate the true partition function p. Denote $h(S) = \mathcal{A}_d(S)$, the partition of S by h.

A distance (equivalently, a similarity) metric d that measures the proximity between two elements is a pairwise function $X \times X \rightarrow R^+$, which can be parameterized to represent a family of functions. For example, given any two element $x_1 =< x_1^{(1)}, \cdots, x_1^{(m)} >$ and $x_2 =<$

[2]Overlapping partitions will not be discussed here.

$x_2^{(1)}, \cdots, x_2^{(m)} >$ in an m-dimensional space, the family of weighted Euclidean distances with parameters $\theta = \{w_l\}_1^m$ is defined as:

$$d_\theta(x_1, x_2) \equiv \sqrt{\sum_{l=1}^{m} w_l \cdot |x_1^{(l)} - x_2^{(l)}|^2} \tag{3.1}$$

When supervision (e.g. class index of elements) is unavailable, the quality of a partition function h operating on $S \subseteq X$, is measured with respect to the distance metric defined over X. Suppose h partitions S into $\{S'_k\}_1^K$, a typical *quality function* (used in K-Means) is defined as:

$$q_S(h) \equiv -\sum_k \sum_{x \in S'_k} d(x, \mu'_k)^2, \tag{3.2}$$

where μ'_k is the mean of elements in S'_k. We note that the rational for using this quality function is that when the data is sampled from a mixture of Gaussians $\{N(\mu_k, \sigma^2)\}_1^K$ with equal covariance, $q_S(h)$ measures the likelihood of the data. However, this measure can be computed (and is being used in practice) irrespective of that and the algorithm used.

In many cases, supervision is used at the application level when determining an appropriate distance metric (Lee, 1999; Weeds, Weir, & McCarthy, 2004; Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2003). When applying clustering to a given task, one typically decides on the clustering quality measure one wants to optimize, and then chooses a specific clustering algorithm and a distance metric. The "goodness" of a metric is empirically measured when combined with different clustering algorithms on different problems. Without any supervision, the resulting partition function is not guaranteed to agree with the target function (or the user's original intention).

Moreover, it is not clear whether there exists any 'universal' metric that is good for different problems (or even different data sets for similar problems) and is appropriate for any clustering algorithm. We illustrate this critical point in Figure 3.1. The 12 points are clustered into 2 groups, represented by solid and hollow points respectively. Data elements are positioned in a two-dimensional space $< X^{(1)}, X^{(2)} >$. (a) and (b) show that even for the same data collection,

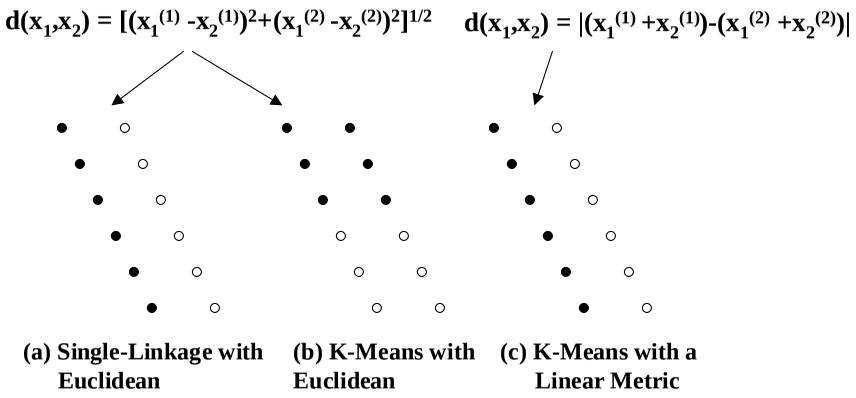

Figure 3.1: **Different combinations of clustering algorithms with distance metrics.**

different clustering algorithms with the same metric could generate different outcomes. (b) and (c) show that with the same clustering algorithm, different metrics could also produce different outcomes. *Therefore, a good distance metric should be both domain-specific and associated with a specific clustering algorithm.*

Several recent works have pursued the general direction of exploiting supervision or learning to improve clustering performance, but the idea of learning an algorithm-specific metric has not been fully considered for the general case. For example, (Cohen & Richman, 2002b; Cohen, Ravikumar, & Fienberg, 2003a; Li, Morie, & Roth, 2004a) make use of supervision in a pairwise classification task and learn a metric, but do it independent of the clustering algorithm they use. (Xing, Ng, Jordan, & Russell, 2002; Bar-Hillel, Hertz, Shental, & Weinshall, 2003; Schultz & Joachims, 2004; Mochihashi, Kikui, & Kita, 2004) formalize the problem of metric learning as an optimization problem, without exploiting a clustering algorithm or only implicitly exploiting one (e.g. K-Means) by optimizing the same objective function. That is, the clustering algorithm is not explicitly taken into account in the learning procedure.

Several works (Bach & Jordan, 2003; Bilenko, Basu, & Mooney, 2004) suggest to learn a distance function directly and develop their ideas for a specific clustering algorithm. The former learns a metric for spectral clustering, and optimizes a quality measure of the partition, but without exploiting feedback from supervision. The latter actually learns a metric for K-Means with feedback from supervision, but the learning procedure is specific to this clustering algorithm.

EM can be used, in some cases (Frey & Jojic, 2003; Tsuda, Akaho, & Asai, 2003) as a standalone partition function in clustering, and could even be used to learn a metric directly by extending the parameter space of a generative model to include the metric parameters. However, this approach is only effective when the clustering algorithm used in evaluation is itself an EM algorithm defined over the same generative model used in training.

The Supervised Discriminative Clustering framework that we describe below proposes a way to resolve these limitations. Namely, it allows learning a distance function for any given clustering algorithm in the context of a given task, exploiting supervision.

3.2 Supervised Discriminative Clustering Framework

When supervision is available, we design a general discriminative clustering approach, the Supervised Discriminative Clustering framework (SDC), to exploit it in clustering (as shown in Figure 3.2).

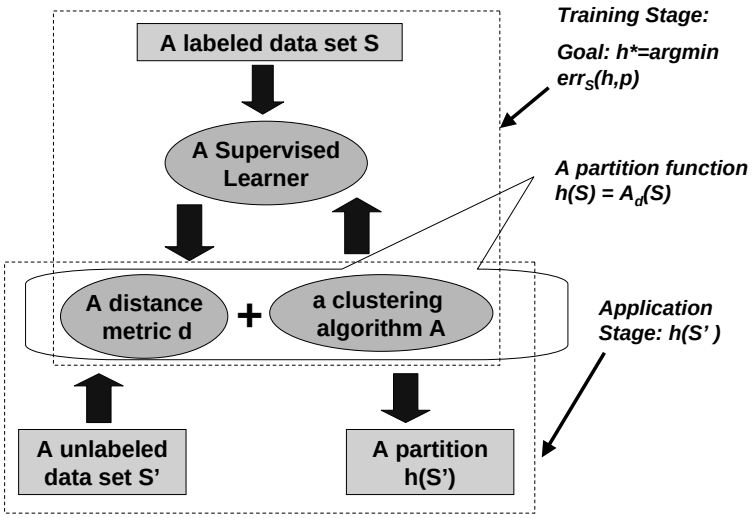

Figure 3.2: **Supervised Discriminative Clustering**

In this framework, a clustering task is split into training and application stages, and *the chosen clustering algorithm is explicitly involved in both stages*. In the training stage, supervision is directly integrated into the error function $err_S(h, p)$, and the goal is to find a partition function

$h \in H$ (H is the chosen hypothesis space), parameterized by a clustering algorithm $\mathcal{A}$ and a metric d to approximate the true function p, by minimizing the error . Consequently, given new data S' in the application stage, under some standard learning theory assumptions, the learned partition function is expected to generalize well and achieve small error.

3.2.1 Error Functions

Let p be the target partition function over X, and let $h \in H$ be a partition hypothesis, and $h(S) = \{S'_k\}_1^K$. In principle, given data set $S \subseteq X$, if the true partition $p(S) = \{S_k\}_1^K$ of S is available, one can measure the deviation of h from p over S, using an *error function* $err_S(h, p) \rightarrow R^+$. Please note that we distinguish an error function from a quality function q (e.g. as in Equation (3.2)) in this paper: an error function measures the disagreement between clustering and one's intention when supervision is given, while a quality is defined without any supervision.

For clustering, there generally is no direct way to compare the true class index $p(x)$ of each element with that given by a hypothesis ($h(x)$), so we measure the disagreement between p and h over pairs of elements. Figure (3.3) below provides examples for error functions that can be used for that purpose. There are two types of error over a pair of elements $x_i, x_j \in S$: misclassified apart or together, represented by $A_{ij} \equiv I[p(x_i) = p(x_j)\ \&\ h(x_i) \neq h(x_j)]$ and $B_{ij} \equiv I[p(x_i) \neq p(x_j)\ \&\ h(x_i) = h(x_j)]$, respectively. $I[true] = 1$ is an indicator function.

$$err_S^1(h,p) \equiv \frac{1}{|S|^2} \sum_{x_i,x_j \in S} \left[d(x_i,x_j)^2 \cdot A_{ij} + (d^{*2} - d(x_i,x_j)^2) \cdot B_{ij} \right]$$

$$err_S^2(h,p) \equiv \frac{1}{|S|} \left| \sum_{k=1}^{K} \sum_{x \in S'_k} d(x,\mu'_k)^2 - \sum_{k=1}^{K} \sum_{x \in S_k} d(x,\mu''_k)^2 \right|$$

$$err_S^3(h,p) \equiv \frac{1}{|S|^2} \left| \sum_{k=1}^{K} \sum_{x_i,x_j \in S'_k} d(x_i,x_j)^2 - \sum_{k=1}^{K} \sum_{x_i,x_j \in S_k} d(x_i,x_j)^2 \right|$$

Figure 3.3: **Examples of error functions.**

err_S^1 is defined as the sum, over all pairs in S, of the two types of pairwise errors, weighted by the squared distance between each pair. We integrate the metric d into the error, in order to penalize large distances between pairs misclassified apart and small distances between pairs misclassified together. $d^* = \max_{ij} d(x_i, x_j)$ is the maximum distance between any two elements in S, which is used to avoid negative error. The error is normalized by $|S|$ – the size of S. Error function err_S^2 and err_S^3 do not directly measure the pairwise errors. Instead, err_S^2 measures the difference between the quality (e.g. as in Equation (3.2)) of the partition by h and that of the true partition. It can be used in the case that the quality is more pursued in application, when considering the tradeoff between accurate partition and better quality. In err_S^2, μ'_k and μ''_k are the mean of elements in the cluster $S'_k \in h(S)$ and the mean in $S_k \in p(S)$, respectively. err_S^3 is a pairwise version of err_S^2.

3.2.2 Supervised and Unsupervised Training

In the training stage, the goal is to learn a good partition function given a set of observed data. Depending on whether the data is labeled or unlabeled, we can further define supervised and unsupervised training.

Definition 3.2.1 ***Supervised Training:*** *Given a labeled data set S and $p(S)$, a family of partition functions H, and the error function $err_S(h, p)(h \in H)$, the problem is to find an optimal function h^* s.t.*

$$h^* = argmin_{h \in H}\ err_S(h, p).$$

Definition 3.2.2 ***Unsupervised Training:*** *Given an unlabeled data set S ($p(S)$ is unknown), a family of partition functions H, and a quality function $q_S(h)(h \in H)$, the problem is to find an optimal partition function h^* s.t.*

$$h^* = argmax_{h \in H}\ q_S(h).$$

By fixing the clustering algorithm, we can further define supervised metric learning, a special case of supervised training.

Definition 3.2.3 ***Supervised Metric Learning:*** *Given a labeled data set S and $p(S)$ and a family of partition functions $H = \{h\}$ that are parameterized by a chosen clustering algorithm $\mathcal{A}$ and a family of distance metrics d_θ ($\theta \in \Omega$), the problem is to seek an optimal metric d_{θ^*} with respect to $\mathcal{A}$, s.t. for $h(S) = \mathcal{A}_{d_\theta}(S)$*

$$\theta^* = argmin_\theta \, err_S(h, p). \tag{3.3}$$

Learning the distance metric parameters requires parameterizing h as a function of d_θ, where the algorithm $\mathcal{A}$ is chosen and fixed in h. One example of this task is that of learning weighted Euclidean distances: $d_\theta(x_1, x_2) = \sqrt{\sum_{l=1}^{m} w_l \cdot |x_1^{(l)} - x_2^{(l)}|^2}$ for the K-Means algorithm in our later experiments. Note that in this case one needs to enforce some constraints, such as a normalization $\sum_{l=1}^{m} |w_l| = 1$ so that the error will not be scale-dependent (e.g., metrics giving smaller distance are always better).

3.2.3 A General Learner for SDC

In addition to the theoretical SDC framework, we also develop a learning algorithm based on gradient descent that can train a distance function for the setting of supervised metric learning (in Figure 3.4). The training procedure incorporates the clustering algorithm (step 2.a) so that the metric is trained with respect to the specific algorithm that will be applied in evaluation. The convergence of this general training procedure depends on the convexity of the error as a function of θ. For example, since the error function we use is *linear* in θ, the algorithm is guaranteed to converge to a global minimum. In this case, for rate of convergence, one can appeal to general results that typically imply, when there exists a parameter vector with zero error, that convergence rate depends on the 'separation" of the training data, which roughly means the minimal error archived with this parameter vector. Results such as (Freund & Schapire, 1998) can be used to extend the rate of convergence result a bit beyond the separable case, when a small number of the

pairs are not separable.

Algorithm: SDC-Learner
Input: S and p: the labeled data set. $\mathcal{A}$: the clustering algorithm. $err_S(h, p)$: the clustering error function. $\alpha > 0$: the learning rate. T (typically T is large) : the number of iterations allowed.
Output: θ^* : the parameters in the distance function d.

1. In the initial (I-) step, we randomly choose θ^0 for d. After this step we have the initial d^0 and h^0.
2. Then we iterate over t $(t = 1, 2, \cdots)$,
   (a) Partition S using $h^{t-1}(S) \equiv \mathcal{A}_{d^{t-1}}(S)$;
   (b) Compute $err_S(h^{t-1}, p)$ and update θ using the formula: $\theta^t = \theta^{t-1} - \alpha \cdot \frac{\partial err_S(h^{t-1}, p)}{\partial \theta^{t-1}}$.
   (c) Normalization: $\theta^t = \frac{1}{Z} \cdot \theta^t$, where $Z = ||\theta^t||$.
3. Stopping Criterion: If $t > T$, the algorithm exits.

Figure 3.4: **A general training algorithm for SDC**

3.3 Metric Learning with the EM Algorithm

The EM algorithm is typically viewed as an approach for learning a probabilistic mixture model in an unsupervised setting (McLachlan & Krishnan, 1997; Friedman, 1998). The core idea is to iteratively seek model parameters that achieve a local optimum of the expected log-likelihood of a set of observed elements with hidden class labels. In this paper, the EM approach is extended within the SDC framework, yielding some variants of EM that can be used to learn a metric for a given clustering algorithm. Please note that in these approaches, the clustering algorithm that will be used in actual application always explicitly participates in the training process, following our previous claim that a good metric should be algorithm-specific.

Suppose a probabilistic mixture model (e.g., Gaussian mixture model), parameterized with $\theta' \in \Omega'$, defines a probability distribution P over $X \times C$. The log-likelihood of generating any set

of labeled elements $(S, p(S)) = \{x_i, p(x_i)\}_1^n$ from $X \times C$ (i.i.d. sampled), is defined as:

$$LL(S, p(S)|\theta') = log[\Pi_{x \in S} Pr[x_i, p(x_i)|\theta']] \tag{3.4}$$

where $p(S)$ is a partition of S and $p(x_i) \in C$ is the corresponding class index of x_i. Suppose S is observed but $p(S)$ is unknown, the standard EM attempts to estimate θ'^* in which the expected log-likelihood is optimized:

$$\theta'^* \equiv \underset{\theta' \in \Omega'}{\operatorname{argmax}} \, E_{p(S)}[LL(S, p(S)|\theta')] \tag{3.5}$$

EM can be used to learn a metric directly by training metric parameters the same way as model parameters. However, as previously noted, this approach is only effective when the clustering algorithm used in evaluation is itself an EM algorithm. Thus, it is restricted to a specific class of clustering algorithms. In order to overcome this restriction, we develop a variant of EM – EM* to learn a distance metric for any clustering algorithm in the setting of probabilistic mixture models. Given a set of unlabeled elements S, a family of distance metrics d_θ (parameterized by $\theta \in \Omega$), and a clustering algorithm $\mathcal{A}$, the goal of EM* is to seek an optimal metric d_{θ^*} s.t.,

$$\theta^* \equiv \underset{\theta \in \Omega}{\operatorname{argmax}} \, LL(S, A_{d_\theta}(S)|\theta) \tag{3.6}$$

where $A_{d_\theta}(S)$ is the partition of S by the chosen clustering algorithm $\mathcal{A}$ with the distance function d_θ. For example, as in our experiments, we learn weighted Euclidean distances for K-Means for a mixture of Gaussian models with EM*.

There are several issues in the framework. One is how to compute $LL(S, A_{d_\theta}(S)|\theta)$, when θ is the parameters of the distance metric rather than those of the underlying probabilistic model θ'. In general, log-likelihood of a sampled data set can be computed independent of a metric, since it just relies on a distribution over a data space. Sometimes, though, it is natural to make some assumptions and link these two notions: that is, to define a metric as a function of an assumption

on the generative process of the distribution, or vice versa. In principle, we can define

$$LL(S, A_{d_\theta}(S)|\theta) \equiv max_{\theta'} LL(S, A_{d_\theta}(S)|\theta') \tag{3.7}$$

where $LL(S, A_{d_\theta}(S)|\theta')$ can be computed in the same way as in Equation (3.4). For a Gaussian mixture model $\{N(\mu_k, \sigma_k^2)\}_1^K$ and $\mathcal{A}_{d_\theta}(S) = \{S_k\}_1^K$, when the distance metric d_θ is a weighted Euclidean distance, the optimal θ' in Equation (3.7) is

$$\mu_k = \frac{1}{|S_k|} \sum_{x \in S_k} x, \qquad \sigma_k = \sqrt{\frac{1}{|S_k|} \sum_{x \in S_k} d_\theta(x, \mu_k)^2}.$$

Another issue is that, in order to make it sound to compare log-likelihoods of clusterings under different distance functions, which can also be scale-dependent, some constraints should be enforced, (e.g. $\sum_{l=1}^m |w_l| = 1$ for weighted Euclidean distances).

The practical EM* algorithm finds a locally optimal θ^* by iterating the following two steps ($t = 0, 1, 2, \cdots$):

$$E - step: \qquad h^t(S) = A_{d_{\theta^t}}(S)$$
$$M - step: \qquad \theta^{t+1} = \operatorname*{argmax}_\theta LL(S, h^t(S)|\theta)$$

After randomly initializing the metric d_{θ^0}, in each E-step, we run the clustering algorithm $\mathcal{A}$ with the distance function d_{θ_t} to partition the data set S. In each M-step, we re-estimate θ^{t+1} based on a gradient descent approach. Some normalization is performed to satisfy the enforced constraint.

A further extension of EM* with labeled data generates a Supervised EM* algorithm. Given a set of labeled elements S and $p(S)$, the Supervised EM* algorithm seeks a metric d_{θ^*} that achieves the minimum difference between the likelihood of the partition by $\mathcal{A}$ with $d_\theta(\theta \in \Omega)$and that of the true partition $p(S)$, under constraints such as $||\theta|| = 1$.

$$\theta^* \equiv \operatorname*{argmin}_{\theta \in \Omega} |LL(S, \mathcal{A}_{d_\theta}(S)|\theta) - LL(S, p(S)|\theta)| \tag{3.8}$$

3.3.1 Relations between SDC and Supervised EM*

SDC and Supervised EM* are discriminative and generative approaches based on the same idea — learning distance functions to minimize disagreement between clustering and the true partition, and can be applied to discriminative clustering and model-based clustering respectively. In a specific case, they are equivalent.

Suppose that we are learning a distance function d_θ in which a distribution over the data space $X \times C$ satisfies a uniform mixture of K Gaussian models $\{g_k \sim N(\mu_k, \sigma^2)\}_1^K$ with the same covariance. That is, any $(x, p(x)) \in X \times C$ satisfies $Pr(x, p(x) = k) = \frac{1}{2\pi\sigma K} \exp(-\frac{d_\theta(x,\mu_k)^2}{2\sigma^2})$. In this case, we have the following theorem [3]:

Theorem 3.3.1 *Given a set of labeled elements $S \subseteq X$ and $p(S)$, the Supervised EM* algorithm, and SDC with error function err_S^2 (in Figure 3.3) have equivalent objectives in learning a distance metric d_θ for the K-Means algorithm $\mathcal{A}$ (let $h(S) = \mathcal{A}_{d_\theta}(S)$). That is,*

$$\operatorname*{argmin}_\theta |LL(S, h(S)|\theta) - LL(S, p(S)|\theta)| = \operatorname*{argmin}_\theta err_S^2(h, p)$$

The complete proof is omitted here. One critical property used: when $p(S) = \{S_k\}_1^K$ and $h(S) = \{S'_k\}_1^K$,

$$LL(S, h(S)|\theta) = a \cdot \sum_{k=1}^{K} \sum_{x \in S'_k} d_\theta(x, \mu'_k)^2 + b \tag{3.9}$$

where a,b are two constants in θ, and μ'_k is the mean of elements in S'_k. Similarly, $LL(S, p(S)|\theta) = a \cdot \sum_{k=1}^{K} \sum_{x \in S_k} d_\theta(x, \mu''_k)^2 + b$, where μ''_k is the mean of elements in S_k.

3.3.2 Simulation with Gaussian Mixture Models

As an empirical study, we create an artificial data set generated from a uniform mixture of Gaussians from a linear weighted Euclidean metric space (as defined in Equation (3.1)). Specifically, we use a mixture of 3 Gaussians of variance 4, with means placed opposite each other on an 18-

[3] Proof is omitted here, but available on request.

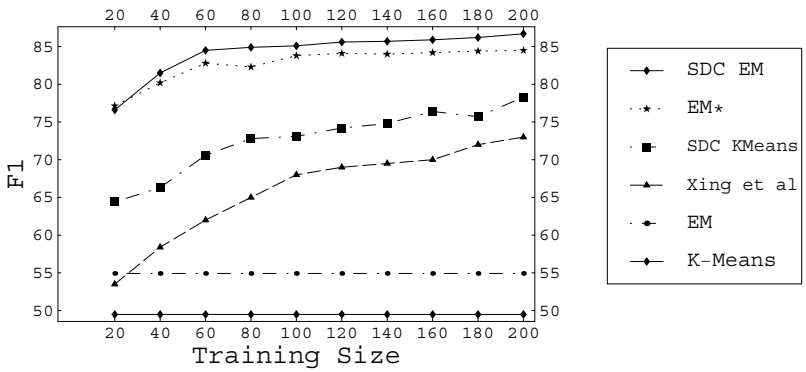

Figure 3.5: **Performance of different clustering approaches.** Different algorithms are evaluated on data generated from a Gaussian mixture model in a weighted Euclidean metric space. The plot shows number of elements in the training set versus F_1-Measure.

dimensional sphere of radius 4 in the metric space defined by a weight vector selected uniformly from $[0,1]^{18}$. The approaches are evaluated by their accuracy in partitioning a set of unlabeled elements into 3 clusters, reported in F_1 measure of classifying pairs of elements as "together" or "not together" in a test set. $F_1 = \frac{2|M_p \bigcap M_a|}{|M_p|+|M_a|}$, where M_p and M_a are the set of pairs of elements predicted to be "together", and the set of pairs annotated as "together", respectively. We display results for two comparisons, obtained by repeated runs with varying generating metrics, and evaluated on $1,000$ test elements.

Comparison of Different Approaches: The first comparison (in Figure 3.5) is among different approaches we have discussed. The approaches include EM over a Gaussian mixture model and K-Means, both of which assume a standard Euclidean metric, as well as EM* (see Equation 3.6) and SDC with err_S^2 (see Equation 3.3) , both of which learn over linear weighted Euclidean metrics with additional training sets of elements (labeled elements for SDC). For SDC, we show results for using both a K-Means clustering algorithm and EM as a standalone clustering algorithm, whereas for EM* we show only the results for using EM. We also compare with a method proposed by (Xing, Ng, Jordan, & Russell, 2002), which optimizes a linear weighted squared Euclidean distance by a constrained Newton-Raphson method separate from the clustering algorithm; we use this as an example of an algorithm that learns a metric without reference to the particular cluster-

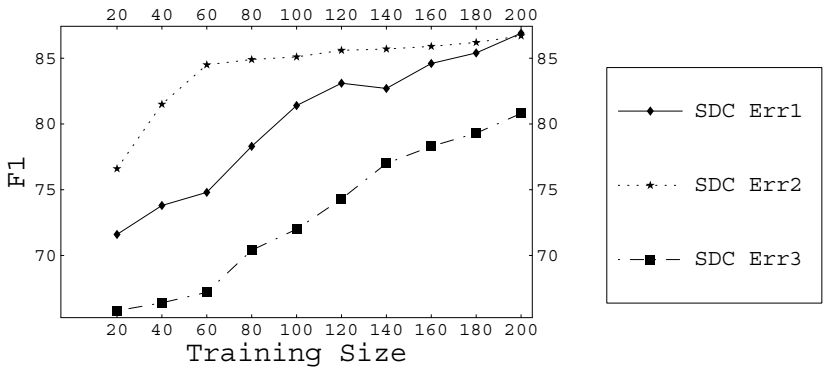

Figure 3.6: **Different error functions for SDC.** These error functions are described in Figure 3.3. The plot shows number of elements in the training set versus F_1-Measure.

ing algorithm that will eventually be used, in this case EM over a Gaussian mixture model.

As the figure shows, metric learning provides clear advantages over standard clustering (over 20% in F_1), and the addition of supervision provides an extra boost in performance ($2\% - 5\%$). These results make good sense, since data in an appropriate metric space will fit the model in a much better way and thus improve clustering. The figure also demonstrates the advantage of learning a metric with respect to the specific clustering algorithm that will be used.

One interesting point to note is that, although SDC with an EM clustering algorithm outperforms EM*, this is not the case for SDC with K-Means. One possible explanation is that while EM* performs the full gradient descent procedure in its M-step, SDC alternates between iterations of gradient descent and clustering, which may introduce randomness and prevent it from fully converging. This problem suggests a possible improvement for SDC.

Comparison of Different Error Functions: As a secondary comparison, we display the results of the SDC approach for different error functions (in Figure 3.6). Specifically, SDC is evaluated on the simulated data and employs the EM algorithm for a Gaussian mixture model as its clustering algorithm. The results for this data appear to strongly support err_S^1 and err_S^2.

3.4 Application to Entity Identification

Several approaches as below are compared in this task. Some of them (for example, those based on SoftTFIDF similarity) do not use any domain knowledge, while others do exploit supervision, such as LMR and SDC. Other works (Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2003) also exploited supervision in this problem but were shown to be inferior.

1. *SoftTFIDF Classifier* – a pairwise classifier deciding whether any two names refer to the same entity, implemented by thresholding a state-of-art SoftTFIDF similarity metric for string comparison (Cohen, Ravikumar, & Fienberg, 2003a). Different thresholds have been experimented but only the best results are reported.

2. *LMR Classifier (P|W)* – a SNoW-based pairwise classifier (Li, Morie, & Roth, 2004a) (described in Section 2.2) that learns a linear function for each class over a collection of relational features between two names: including string and token-level features and structural features (listed in Table 2.1). We train the classifier with both Perceptron (P) and Winnow (W) algorithms.

3. *Clustering over SoftTFIDF* – a clustering approach based on the SoftTFIDF similarity metric.

4. *Clustering over LMR (P|W)* – a clustering approach (Li, Morie, & Roth, 2004a) by converting the LMR classifier into a similarity metric (see Section 2.3.3).

5. *SDC* – our new supervised discriminative clustering approach. In the following experiments, the error function err_S^1 (as defined in Table 3.3) is applied to training a weighted Euclidean distance function under the SDC framework. Given binary pairwise features are extracted for any elements $x_1, x_2 \in X$, $(x_1, x_2) =< \phi_1, \phi_2, \cdots, \phi_m > (\phi_i \in \{0, 1\})$, the distance function, parameterized by ($\theta = \{w_l\}_1^m$), is

$$d(x_1, x_2) \equiv \sqrt{\sum_{l=1}^{m} w_l \cdot \phi_l(x_1, x_2)^2} = \sqrt{\sum_{l=1}^{m} w_l \cdot \phi_l(x_1, x_2)}$$

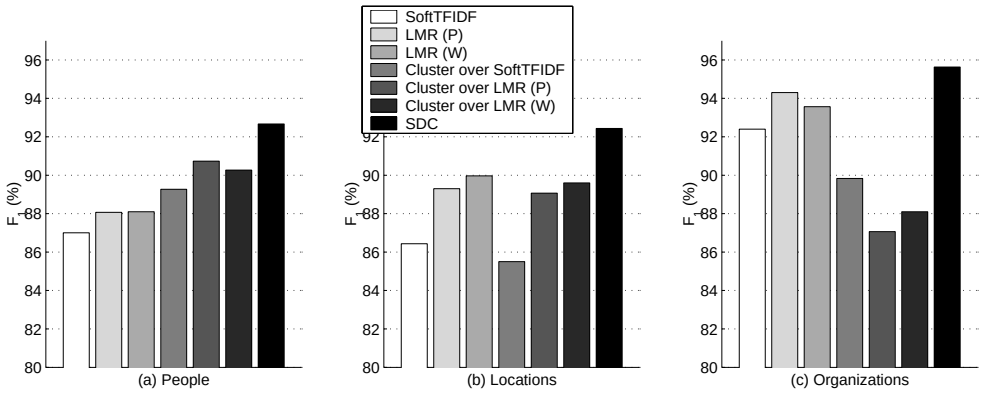

Figure 3.7: **Performance of different approaches.** The results are reported for SDC with a learning rate $\alpha = 100.0$. The Single-Linkage algorithm is applied whenever clustering is performed. Results are reported in F_1 and averaged over the three data sets for each entity type and 10 runs of two-fold cross-validation. Each training set typically contains 300 annotated names.

The above approaches (2), (4) and (5) learn a classifier or a distance metric using the same feature set as in Table 2.1. Different clustering algorithms, such as Single-Linkage, Complete-Linkage, Graph clustering (seeking a minimum cut of a nearest neighbor graph), Repeated Bisections (George, 2003) and K-medoids (Chu, Roddick, & Pan, 2001) are experimented in (3), (4) and (5). We use the clustering package *Cluster* by Michael Eisen at Stanford University for K-mediois clustering and *CLUTO* by (George, 2003) for the other algorithms[4]. The number of classes (entities) that a data set has is known for them.

Our experimental study focuses on (1) evaluating the supervised discriminative clustering approach on entity identification; (2) comparing it with existing pairwise classification and clustering approaches widely used in similar tasks; and (3) further analyzing the characteristics of this new framework.

We use the TREC corpus to evaluate different approaches in identifying three types of entities: People, Locations and Organization. For each type, we generate three data sets, each containing about 600 names. We note that the three entity types yield very different data sets, exhibited by some statistical properties[5]. Results on each entity type will be averaged over the three sets and ten

[4]The weighted version of Single- and Complete-Linkage algorithms in the package are actually used in experiments.

[5]The average SoftTFIDF similarity between names of the same entity is 0.81, 0.89 and 0.95 for people, locations and organizations respectively.

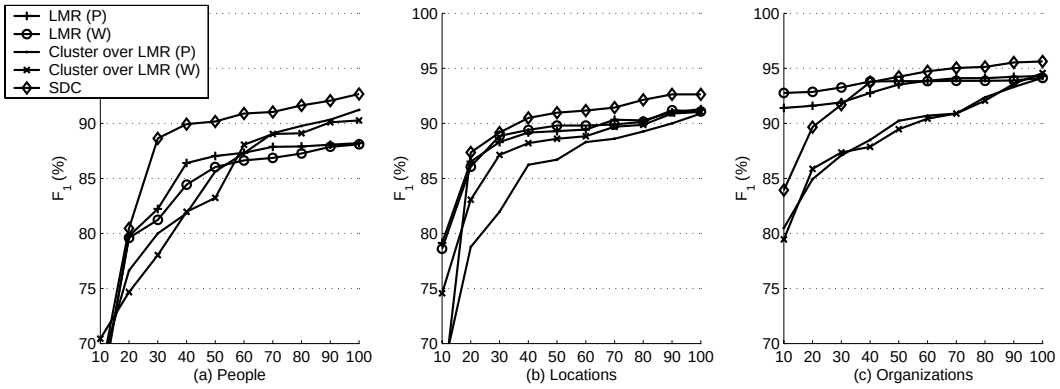

Figure 3.8: **Performance for different training sizes.** Five learning-based approaches are compared. Single-Linkage is applied whenever clustering is performed. X-axis denotes different percentages of 300 names used in training. Results are reported in F_1 and averaged over the three data sets for each entity type.

runs of two-fold cross-validation for each of them. For SDC, given a training set with annotated name pairs, a distance function is first trained using the algorithm in Figure 3.4 (in 20 iterations) with respect to a clustering algorithm and then be used to partition the corresponding test set with the same algorithm.

3.4.1 Comparison of Different Approaches

Figure 3.7 presents the performance of different approaches (described in Section 3.4) on identifying the three entity types. We experimented with different clustering algorithms but only the results by Single-Linkage are reported for *Cluster over LMR (P|W)* and SDC, since they are the best.

SDC works well for all three entity types in spite of their different characteristics. The best F_1 values of SDC are 92.7%, 92.4% and 95.7% for people, locations and organizations respectively, about $20\% - 30\%$ error reduction compared with the best performance of the other approaches. This is an indication that this new approach which integrates metric learning and supervision in a unified framework, has significant advantages [6].

[6]We note that in this experiment, the relative comparison between the pairwise classifiers and the clustering approaches over them is not consistent for all entity types. This can be partially explained by the theoretic analysis in (Li, Morie, & Roth, 2004a) and the difference between entity types.

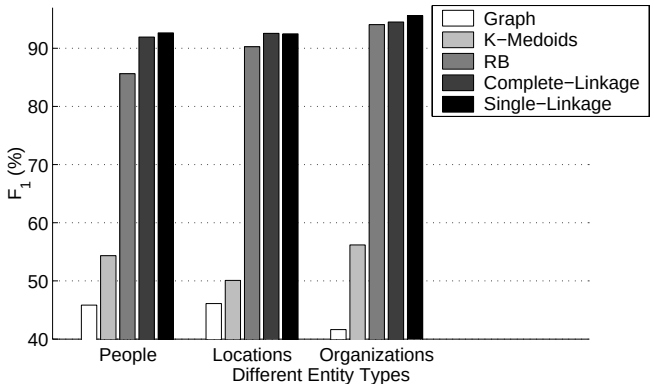

Figure 3.9: **Different clustering algorithms.** Five clustering algorithms are compared in SDC ($\alpha = 100.0$). Results are averaged over the three data sets for each entity type and 10 runs of two-fold cross-validations.

3.4.2 Further Analysis of SDC

In the next experiments, we will further analyze the characteristics of SDC by evaluating it in different settings.

Different Training Sizes Figure 3.8 reports the relationship between the performance of SDC and different training sizes. The learning curves for other learning-based approaches are also shown. We find that SDC exhibits good learning ability with limited supervision. When training examples are very limited, for example, only 10% of all 300 names, pairwise classifiers based on Perceptron and Winnow exhibit advantages over SDC. However, when supervision become reasonable ($30\%+$ examples), SDC starts to outperform all other approaches.

Different Clustering Algorithms Figure 3.9 shows the performance of applying different clustering algorithms in the SDC approach. Single-Linkage and Complete-Linkage outperform all other algorithms. One possible reason is that this task has a great number of classes ($100 - 200$ entities) for 300 names in each single data set. The results indicate that the metric learning process relies on properties of the data set, as well as the clustering algorithm. Even if a good distance metric could be learned in SDC, choosing an appropriate algorithm for the specific task is still important.

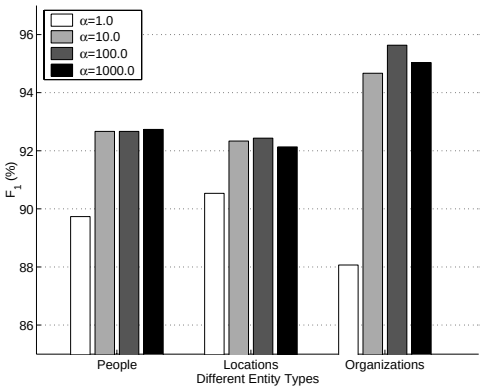

Figure 3.10: **Performance for different learning rates.** SDC with different learning rates ($\alpha = 1.0, 10.0, 100.0, 1000.0$) compared in this setting. Single-Linkage clustering algorithm is applied.

Different Learning Rates We also experimented with different learning rates in the SDC approach as shown in Figure 3.10. It seems that SDC is not very sensitive to different learning rates as long as it is in a reasonable range.

3.4.3 Discussion

The reason that SDC can outperform existing clustering approaches can be explained by the advantages of SDC – training the distance function with respect to the chosen clustering algorithm, guided by supervision. However, it is not so obvious why it can also outperform the pairwise classifiers. The following analysis could give some intuitive explanation.

Supervision in the entity identification task or similar tasks is typically given on whether two names (elements) correspond to the same entity – entity-level annotation. Therefore it does not necessarily mean whether they are similar in appearance. For example, "Brian" and "Wilson" could both be names for a person "Brian Wilson" in different context, and thus this name pair is a positive example in training a pairwise classifier. However, with features that only capture the appearance similarity between names, such apparently different names become training noise. This is what exactly happened when we train the LMR classifier with such name pairs. SDC, however, can employ this entity-level annotation and avoid the problem through transitivity in clustering. In the above example, if there is "Brian Wilson" in the data set, then "Brian" and "Wilson" can be

both clustered into the same group with “Brian Wilson”. Such cases do not frequently occur for locations and organization but still exist .

3.5 Conclusion

This chapter presents a unified framework for clustering that is guided by supervision. In this framework, we explicitly formalize clustering as a learning task, and propose two new approaches for training an algorithm-specific distance metric. Our experiments exhibit their advantages over existing clustering approaches in the case of Gaussian mixture models and real problems. The view provided in this work allows us to develop some theoretical understanding to the problem, relate supervised clustering to existing algorithms and suggest variations of them.

The supervised clustering framework can be viewed as an analog to multi-class classification, but can handle a more general learning scenario. A classifier trained with labeled data can only be used to classify new data which is sampled from the same classes as in training. Clustering, however, can be used to partition examples from new classes, as long as the learned metric works for the whole data space. One example is the entity identification problem where there exist millions of entities (classes) in the world and only few are seen in training. In this sense, SDC cannot be replaced by multi-class classification. One can Combine a clustering algorithm with a metric to create a partition function, (in some sense more expressive than a classifier). The idea of supervised clustering is significant in that few people have considered transplanting the general ideas of classification into clustering, although it seems so natural and intriguing.

In addition to further theoretical analysis of the convergence of the SDC approach and developing more efficient learning algorithms, we also hope to extend it to train kernel-based metrics for more complex feature spaces. Further research in this direction will focus on (1) applying it to more NLP tasks, such as coreference resolution and word sense discovery; (2) analyzing the related theoretic issues, for example, the convergence of the algorithm; and (3) extending it to train kernel-based metrics for more complex feature spaces.

Chapter 4

Generative Models for Entity Identification

All of the discriminative approaches described in the prior chapters, such as pairwise classification and different clustering approaches, only take input the local information existing in the names in order to identify entities. This is clearly not enough since different names can refer to the same entity in different contexts of text, while similar names may refer to different entities. An example we have given is “Kennedy”, there are more than tens of thousands of people share this name. Examples of names of other types that could refer to different entities are “World Trade Center” in different cities, and “Department of Justice” in different countries . The specific context that a name occurs in the text is critical to disambiguate these cases and identify entities in a context-sensitive setting.

In this chapter, we focus on the following three types of contextual information: (1) The notion of ‘Document’. If two names are within the same document and are similar in writing, they are very likely refer to the same entity, while the chance for two similar names in different documents is much lower. Separating the within-document entity identification problem from the across-document problem, and apply different approaches to them, seems a better choice than treating them the same way. (2) Entity co-occurring dependency, e.g., a document that mentions “President Kennedy” is more likely to mention “Oswald” or “ White House” than “Roger Clemens”. (3) The notion of “Representative” of an entity in a document. Typically, the name of an entity occurring the first in a text is the longest name of it. For example, an author tends to write the full name of a person at the beginning of an article, and then write only the last or first names later. Other mentions

of the same entity in the same text, can be viewed as being transformed from the representative by string operations.

Motivated by the above observation, we describe next a generative model for *cross-document entity identification* that is designed to exploit the structure of the documents and assumptions on how they are generated. At the heart of this model 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, (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, governing how mentions are transformed from the "representative" mention. Several relaxations of this model are also described and compared in the experiments. We show that this approach performs very accurately, in the range of $90\% - 95\%$ F_1 measure for different entity types, even better than the discriminative approaches trained in a supervised setting. Given a collection of documents we learn the models in an unsupervised way; that is, the system is not told during training whether two mentions represent the same entity. We only assume the ability to recognize names, using a named entity recognizer run as a preprocessor.

The rest of this chapter is organized as follows: We first define the components of the generative probabilistic model in Section 4.1. Section 4.2 then describes a generative view of documents' creation, three practical probabilistic models designed based on it, and discusses how to make inference with these models. After that, We illustrate how to learn these models in an unsupervised setting in Section 4.3, and describes the experimental study of this model in Section 4.4. It will then be compared with our previously developed discriminative approaches along several dimensions in Section 4.5. Finally, Section 4.6 will conclude this chapter.

4.1 Basic Definitions

We consider reading a large number of documents $D = \{d_1, d_2, \ldots, d_m\}$, each of which may contain *mentions* (i.e. real occurrences) of $|T|$ types of *entities*. In the current evaluation we

consider $T = \{Person, Location, Organization\}$.

An *entity* refers to the "real" concept behind the mention and can be viewed as a unique identifier to an object in the real world. Examples might be the person "John F. Kennedy" who became a president, "White House" – the residence of the US presidents, etc. E denotes the collection of all possible entities in the world and $E^d = \{e_i^d\}_1^{l^d}$ is the set of entities mentioned in document d. M denotes the collection of all possible mentions and $M^d = \{m_i^d\}_1^{n^d}$ is the set of mentions in document d. $M_i^d (1 \leq i \leq l^d)$ is the set of mentions that refer to entity $e_i^d \in E^d$. For example, for entity "John F. Kennedy", the corresponding set of mentions in a document may contain "Kennedy", "J. F. Kennedy" and "President Kennedy". Among all mentions of an entity e_i^d in document d we distinguish the one occurring first, $r_i^d \in M_i^d$, as the *representative* of e_i^d. In practice, the representative is usually the longest mention of an entity in the document as well, and other mentions are variations of it. Representatives can be viewed as a typical representation of an entity mentioned in a specific time and place. For example, "President J.F.Kennedy" and "Congressman John Kennedy" may be representatives of "John F. Kennedy" in different documents. R denotes the collection of all possible representatives and $R^d = \{r_i^d\}_1^{l^d} \subseteq M^d$ is the set of representatives in document d. This way, each document is represented as the collection of its entities, representatives and mentions $d = \{E^d, R^d, M^d\}$.

Elements in the name space $W = E \cup R \cup M$ each have an identifying writing (denoted as $wrt(n)$ for $n \in W$)[1] and an ordered list of attributes, $A = \{a_1, \ldots, a_p\}$, which depends on the entity type. Attributes used in the current evaluation include both *internal* attributes, such as, for *People*, {*title, firstname, middlename, lastname, gender*} as well as *contextual* attributes such as {*time, location, proper-names*}. *Proper-names* refer to a list of proper names that occur around the mention in the document. All attributes are of string value and can be empty when the values are missing or unknown[2].

The fundamental problem we address in cross-document entity identification is to decide what

[1]The observed writing of a mention is its identifying writing, i.e., "President Kennedy". For entities, it is a standard representation of them, i.e. the full name of a person.

[2]Contextual attributes are not part of the current evaluation, and will be evaluated in the next step of this work.

entities are mentioned in a given document (given the observed set M^d) and what the most likely assignment of entity to each mention is.

4.2 A Model of Document Generation

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 $P(E^d)$ that governs how entities (of different types) are distributed into a document and reflects their co-occurrence dependencies.

(2) The number of entities in a document, $size(E^d)$, and the number of mentions of each entity in E^d, $size(M_i^d)$, need to be decided. The current evaluation makes the simplifying assumption that these numbers are determined uniformly over a small plausible range.

(3) The *appearance probability* of a name generated (transformed) from its representative is modelled 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.

Given these, a document d is assumed to be generated as follows (see Figure 4.1): A set of $size(E^d)$ entities $E^d \subseteq E$ is selected to appear in a document d, according to $P(E^d)$. For each entity $e_i^d \in E^d$, a representative $r_i^d \in R$ is chosen according to $P(r_i^d|e_i^d)$, generating R^d. Then mentions M_i^d of an entity are generated from each representative $r_i^d \in R^d$ – each mention $m_j^d \in M_i^d$ is independently transformed from r_i^d according to the appearance probability $P(m_j^d|r_i^d)$, after $size(M_i^d)$ is determined. Assuming conditional independency between M^d and E^d given R^d,

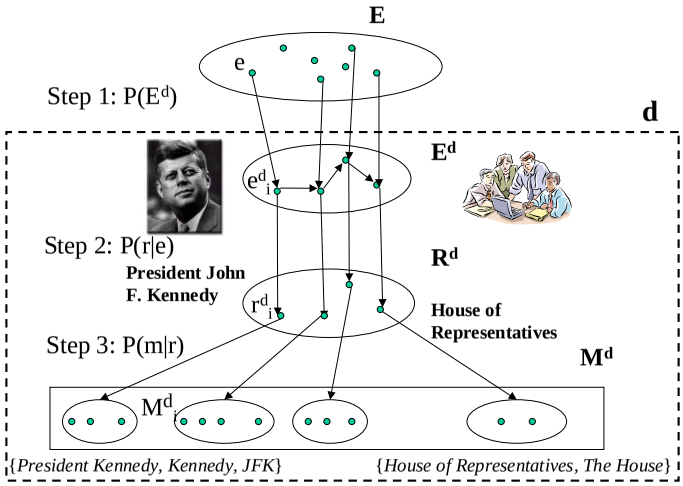

Figure 4.1: **Generating a document.** A document is generated in three steps according to underlying probability distribution.

the probability distribution over documents is therefore

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

and the probability of the document collection D is:

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

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:

$$E^d = argmax_{E' \subseteq E} P(E^d, R^d|M^d, \theta) = argmax_{E' \subseteq E} P(E^d, R^d, M^d|\theta), \tag{4.1}$$

where θ is the learned model's parameters. This gives the assignment of the most likely entity e^*_m for m.

4.2.1 Relaxations of the Model

In order to simplify model estimation and to evaluate some assumptions, several relaxations are made to form three simpler probabilistic models.

4.2.2 Model I (the simplest model)

The key relaxation here is in losing the notion of an "author" – rather than first choosing a representative for each document, mentions are generated independently and directly given an entity.

That is, an entity e_i is selected from E according to the prior probability $P(e_i)$; then its actual mention m_i is selected according to $P(m_i|e_i)$. Also, an entity is selected into a document independently of other entities. In this way, the probability of the whole document set can be written in a simpler way:

$$P(D) = P(\{(e_i, m_i)\}_{i=1}^{n}) = \prod_{i=1}^{n} P(e_i)P(m_i|e_i),$$

and the inference problem for the most likely entity given m is:

$$e^* = argmax_{e \in E} P(e|m, \theta) = argmax_{e \in E} P(e)P(m|e). \tag{4.2}$$

4.2.3 Model II

The major relaxation made here is in assuming a simple model of choosing entities to appear in documents. Thus, in order to generate a document d, after we decide $size(E^d)$ and $\{size(M_1^d, size(M_2^d), \dots\}$ according to uniform distributions, each entity e_i^d is selected into d independently of others according to $P(e_i^d)$. Next, 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 $P(m_j^d|r_j^d)$. Here, we have individual documents along with representatives, and the distribution over documents is:

$$
\begin{aligned}
P(d) = \quad & P(E^d, R^d, M^d) = P(E^d)P(R^d|E^d)P(M^d|R^d) \\
= \quad & [P(size(E^d)) \prod_{i=1}^{|E^d|} P(e_i^d)] \times [P(size(M_1^d), size(M_2^d), ...) \\
& \times \prod_{i=1}^{|E^d|} P(r_i^d|e_i^d)] \times \prod_{(r_j^d, m_j^d)} P(m_j^d|r_j^d) \\
\approx \quad & \prod_{i=1}^{|E^d|} [P(e_i^d)P(r_i^d|e_i^d)] \prod_{(r_j^d, m_j^d)} P(m_j^d|r_j^d)
\end{aligned}
$$

after we ignore the size components. The inference problem here is the same as in Equation 4.1.

4.2.4 Model III (Least Restrictions)

This model performs the least relaxation. After deciding $size(E^d)$ according to a uniform distribution, instead of assuming independency among entities which does not hold in reality (For example, "Gore" and "George. W. Bush" occur together frequently, but "Gore" and "Steve. Bush" do not), we select entities using a graph based algorithm: entities in E are viewed as nodes in a weighted directed graph with edges (i, j) labelled $P(e_j|e_i)$ representing the probability that entity e_j is chosen into a document that contains entity e_i. We distribute entities to E^d via a random walk on this graph starting from e_1^d with a prior probability $P(e_i^d)$. Representatives and mentions are generated in the same way as in Model II. Therefore, a more general model for the distribution over documents is:

$$
P(d) \approx P(e_1^d)P(r_1^d|e_1^d) \prod_{i=2}^{|E^d|} [P(e_i^d|e_{i-1}^d)P(r_i^d|e_i^d)] \times \prod_{(r_j^d, m_j^d)} P(m_j^d|r_j^d).
$$

The inference problem is the same as in Equation 4.1.

4.2.5 Inference

The fundamental problem in cross-document entity identification can be solved as inference with the models: given a mention m, seek the most probable entity $e \in E$ for m according to Equation 4.2 for Model I or Equation 4.1 for Model II and III. The inference algorithm for Model I (with time complexity $O(|E|)$) is simple and direct: just compute $P(e, m)$ for each candidate entity $e \in E$ and then choose the one with the highest value. Due to exponential number of possible assignments of E^d, R^d to M^d in Model II and III, precise inference is infeasible. Approximate algorithms are therefore designed:

In Model II, we adopt a two-step algorithm: First, we seek the representatives R^d for the mentions M^d in document d by sequentially clustering the mentions according to the appearance model. The first mention in each group is treated as the representative. Specifically, when considering a mention $m \in M^d$, $P(m|r)$ is computed for each representative r that have already been created and a fixed threshold is then used 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. In the second step, each representative $r_i^d \in R^d$ is assigned to its most likely entity according to $e^* = argmax_{e \in E} P(e) * P(r|e)$ [3]. This algorithm has a total time complexity of $O(|M^d|^2 + |E| * |R^d|)$.

Model III has a similar two-step algorithm as Model II. The only difference is that we need to consider the global dependency between entities. Thus in the second step, instead of seeking an entity e for each representative r separately, we determine a set of entities E^d for R^d in a Hidden Markov Model with entities in E as hidden states and R^d as observations. The prior probabilities, the transitive probabilities and the observation probabilities for this HMM are given by $P(e)$, $P(e_j|e_i)$ and $P(r|e)$ respectively. In this step we seek the most likely sequence of entities given these representatives in their appearing order using the Viterbi algorithm. The total time complexity is $O(|M^d|^2 + |E|^2 * |R^d|)$. However, it can be reduced by filtering out most irrelevant entities of a mention beforehand using some simple heuristics.

[3] E is known after learning the model in a closed document collection that d belongs to.

4.2.6 Discussion

Besides different assumptions of the models, there are some fundamental differences in inference with the models as well. In Model I, the entity of a mention is determined completely independently of other mentions, while in Model II the way of figuring out the entity relies on local similarity among mentions in the same document. In Model III, it is not only related to other mentions but to a global dependency over entities. The following conceptual example illustrates these differences as in Figure 4.2.

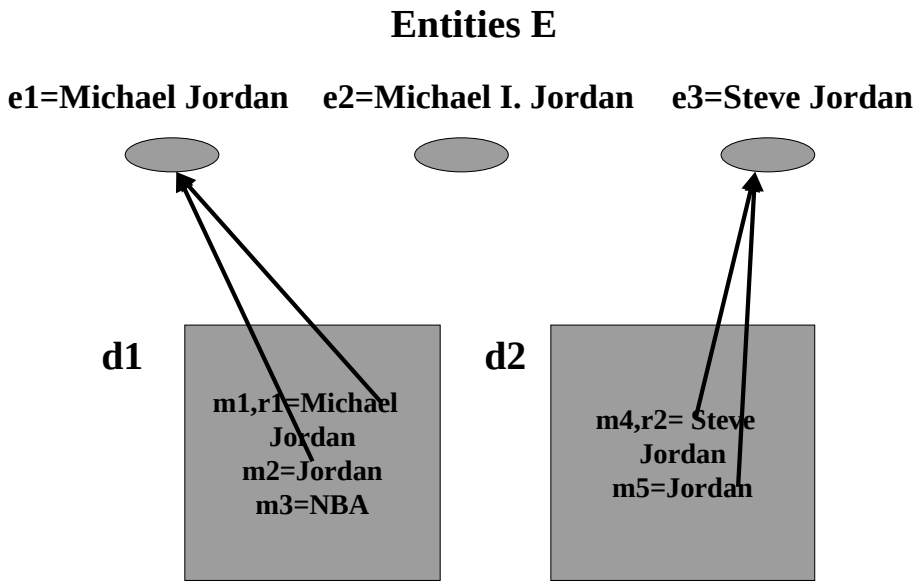

Figure 4.2: **A conceptual example showing the differences of Model I,II,III.** There are five mentions $\{m_i\}_1^5$ observed in two documents $\{d_1, d_2\}$ and three entities $\{e_j\}_1^3$. The arrows represent correct assignment of entities to mentions. r_1, r_2 are representatives.

Example 4.2.1 *Given $E = \{$George Bush, George W. Bush, Steve Bush$\}$, documents d_1, d_2 and five mentions in them, and suppose the prior probability of entity "George W. Bush" is higher than those of the other two entities, the probable assignment of entities to mentions in the three models could be as follows:*

For Model I, $mentions(e_1) = \phi$, $mentions(e_2) = \{m_1, m_2, m_5\}$ and $mentions(e_3) = \{m_4\}$. The result is caused by the fact that a mention tends to be assigned to the entity with higher prior probability when the appearance similarity is not distinctive.

For Model II, $mentions(e_1) = \phi$*,* $mentions(e_2) = \{m_1, m_2\}$ *and* $mentions(e_3) = \{m_4, m_5\}$*. Local dependency (appearance similarity) among mentions inside each document enforces constraints that they should refer to the same entity, like "Steve Bush" and "Bush" in* d_2*.*

For Model III, $mentions(e_1) = \{m_1, m_2\}$*,* $mentions(e_2) = \phi$*,* $mentions(e_3) = \{m_4, m_5\}$*. With the help of global dependency among entities, for example, "George Bush" and "J. Quayle", an entity can be distinguished from another entity with a similar writing.*

4.3 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 entity. A search algorithm modified after the standard EM algorithm (McLachlan & Krishnan, 1997) (We call it 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.

4.3.1 Truncated EM Algorithm

The basic framework of the Truncated EM algorithm to learn Model II and III is as follows:

The algorithm for Model I is similar to the above algorithm but much simpler in the sense that it does not have the notions of documents and representatives. So in the E-step we only need to seek the most possible entity e for each mention $m \in D$ and this simplifies the parameter estimation in the M-step accordingly. It usually takes $3 - 10$ iterations before the algorithm stops for all the models in our experiments.

1. In the initial (I-) step, an initial E_0^d and R_0^d is assigned to each document d using an initialization algorithm. After this step, we can assume that we have labelled documents $D^0 = \{(E_0^d, R_0^d, M^d)\}$.
2. In the M-step, we seek the model parameter θ^{t+1} that maximizes $P(D^t|\theta)$. Given the "labels" supplied by the model in the previous I- or E-step, this amounts to the maximum likelihood estimation as described in Section 4.3.3.
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 is the same inference problem in Section 4.2.5.
4. Stoping 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 E-step.

Figure 4.3: **The Truncated EM algorithm**

Theorem 4.3.1 ***Convergence of TEM:*** *The Truncated EM algorithm converges to a local maxima of* $P(D|\theta)$.

Proof: The proof is somewhat trivial. During each iteration $t = 1, 2, 3, \cdots$, in the M-step, we recompute the model parameter θ^t to maximize $P(D|\theta^t)$, so $P(D|\theta^{t+1}) \geq P(D|\theta^t)$; in the E-step, we re-annotate $D(E^d, R^d)$ to maximize $P(D^{t+1}|\theta^{t+1})$, so $P(D^{t+1}|\theta^{t+1}) \geq P(D^t|\theta^{t+1})$. $P(D^t|\theta^t)$ is non-decreasing in each iteration and it has an upper bound 1, so TEM converges to a local maxima of $P(D|\theta)$.■

However, due to computational complexity, we design approximate algorithms in M- and E-step, and therefore convergence is not guaranteed in this case[4].

4.3.2 Initialization

The purpose of the initial step is to acquire an initial guess of document structures and to seek the set of entities E in a closed collection of documents D. The hope is to find all entities without loss even if repeated entities might be created. For all the models, we apply the same algorithm:

[4]K-means is a Truncated EM algorithm rather than a standard EM.

First, a local clustering is performed to group all mentions inside each document. A set of simple heuristics of matching attributes is applied to calculating the similarity between mentions and pairs of mentions with similarity above a threshold are clustered together. The first mention in each group is chosen as the representative (only in Model II and III) and an entity having the same writing with the representative is created for each cluster[5].

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.

4.3.3 Model Parameter Estimation

In the learning process, assuming we have obtained labelled documents $D = \{(e, r, m)\}_1^n$ from previous I- or E-step, several probability distributions underlying the relaxed models are estimated according to maximum likelihood estimation in each M-step. The model parameters include a prior distribution over entities P_E, a transitive probability distribution over pairs of entities $P_{E|E}$ (only in Model III) and the appearance probability $P_{W|W}$ of a name in the name space W being transformed from another name.

- The prior distribution P_E is modelled as a multi-nomial distribution. Given a set of labelled entity-mention pairs $\{(e_i, m_i)\}_1^n$,

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

where $freq(e)$ denotes the number of pairs containing entity e.

- Given all the entities appearing in D , The transitive probability between entities $P(e|e)$ is estimated by

$$P(e_2|e_1) \sim P(wrt(e_2)|wrt(e_1)) = \frac{doc^{\#}(wrt(e_2), wrt(e_1))}{doc^{\#}(wrt(e_1))}.$$

Here, the conditional probability between two real entities $P(e_2|e_1)$ is backed off to the conditional probability between the identifying writings of the two entities $P(wrt(e_2)|wrt(e_1))$ in the

[5]Note that the performance of the initialization algorithm is 97.3% precision and 10.1% recall.

document set D to avoid sparsity problem. Given $D = \{d_1, d_2, ..., d_m\}$. And $doc^{\#}(w_1, w_2, ...)$ denotes the number of documents having the co-occurrence of writings $w_1, w_2, ...$

- Appearance Probability, the probability of one name being transformed from another, denoted as $P(n_2|n_1)$ $(n_1, n_2 \in W)$, is modelled as a product of the transformation probabilities over attribute values. The transformation probability for each attribute in A is further modelled as a multi-nomial distribution over a set of predetermined "typical" transformation types that depend on the entity types: $TT = \{copy, missing, typical, non-typical\}$[6].

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, the transformation probabilities $P_{M|R}$, $P_{R|E}$ and $P_{M|E}$, are all modelled as a product distribution (naive Bayes) over attributes:

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

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

$$P(t, k) = \frac{freq(r, m) : v_k^r \rightarrow_t v_k^m}{n} \tag{4.3}$$

$v_k^r \rightarrow_t v_k^m$ 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.

[6] **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"; **non-typical** denotes a non-typical transformation.

4.4 Experimental Study

Our experimental study here focuses on (1) evaluating the three models on identifying three entity types (People, Locations, Organization); (2) evaluating the contribution of the global nature of our model, and finally, (3) evaluating our models on name expansion and prominence ranking.

The document segments shown in Figure 2.2 exemplify the preprocessed data given as input to the evaluation. The learning approaches were evaluated on their ability to determine whether a pair of entities (within or across documents) actually correspond to the same real-world entity.

We still make use of the New York Times articles in the experiments as in Section 4.4. The training process gets to see all of the 300 documents and extracts attribute values for each mention, but no supervision is supplied. These records are used to learn the probabilistic models. In testing, all of the $130,000$ pairs of mentions that correspond to the same entity in the 300 documents are generated, and are used to evaluate the models' performance. Since the probabilistic models are learned in an unsupervised setting, testing can be viewed simply as the evaluation of the learned model, and is thus done on the same data. The same setting was used for all models and all comparison performed (see below). To evaluate the performance, we pair two mentions if and only if the learned model determined that they correspond to the same entity. The list of predicted pairs is then compared with the annotated pairs. Precision, Recall and F_1 are computed the same way as in Section 2.2.2.

The generative models are compared with two straightforward discriminatory models. The first one is a simple baseline algorithm according to which two names are co-referred if and only if they have identical writings. The second is a pairwise classifier based on a state-of-art similarity measure for entity names (SoftTFIDF with Jaro-Winkler distance and $\theta = 0.9$) as described in Section 2.1; it was ranked the best measure in a recent study (Cohen, Ravikumar, & Fienberg, 2003a).

4.4.1 Comparison of Different Models

Table 4.1 presents a detailed evaluation of the different approaches on the entity identity task. All the three probabilistic models outperform the discriminative approaches in this experiment, an indication of the effectiveness of the generative model.

Entity Type	Mod	InDoc $F_1(\%)$	InterDoc $F_1(\%)$	All R(%)	All P(%)	All $F_1(\%)$
All	B	86.0	68.8	58.5	85.5	**70.7**
	D	86.5	78.9	66.4	95.8	**79.8**
	I	96.3	85.0	79.0	94.1	**86.2**
	II	96.5	88.1	85.9	92.2	**89.0**
	III	96.5	87.9	84.4	93.6	**88.9**
P	B	82.4	59.0	48.5	86.3	64.7
	D	82.4	67.1	54.5	91.5	70.6
	I	96.2	84.8	80.6	94.8	87.4
	II	96.4	91.7	94.0	91.5	92.7
	III	96.4	88.9	89.8	91.3	90.5
L	B	88.8	63.0	54.8	75.0	64.1
	D	91.4	76.0	61.3	95.9	76.7
	I	92.9	78.9	70.9	89.1	79.5
	II	93.8	81.4	76.2	88.1	81.9
	III	93.8	82.8	76.0	91.2	83.3
O	B	95.3	82.8	72.6	96.4	83.7
	D	95.8	90.7	83.9	98.9	91.1
	I	98.8	91.8	86.5	98.5	92.3
	II	98.5	92.5	88.6	97.5	92.9
	III	98.8	93.0	88.5	98.6	93.4

Table 4.1: **Performance of different approaches over all test examples.** B, D, I, II and III denote the baseline model, the SoftTFIDF similarity model with clustering, and the three probabilistic models. *All,P,L,O* denote all entities, People, Locations and Organizations respectively. We distinguish between pairs of mentions that are inside the same document (*InDoc*, 10.5% of the pairs) or not (*InterDoc*).

We note that although Model III is more expressive and reasonable than model II, it does not always perform better. Indeed, the global dependency among entities in Model III achieves two-folded outcomes: it achieves better precision but, may degrade the recall. The following example, taken from the corpus, illustrates the advantage of this model.

Example 4.4.1 *"Sherman Williams" is mentioned along with the baseball team "Dallas Cowboys" in eight out of* 300 *documents, while "Jeff Williams" is mentioned along with "LA Dodgers"*

in two documents.

In all the models except Model III, "Jeff Williams" is judged to correspond to the same entity as "Sherman Williams" since they are quite similar and the prior probability of the latter is higher than the former. Only in Model III, due to the dependency between "Jeff Williams" and "Dodgers", the system identifies it as corresponding to a different entity than "Sherman Williams".

While this exhibits the better precision achieved by Model III, the recall may go down. The reason is that the global dependency among entities in Model III enforces restrictions over possible grouping of similar mentions; in addition, with a limited document set, estimating this global dependency cannot be done accurately, especially in the setting that entities themselves need to be found when learning the model. We expect that Model III will dominate Model II when we have enough data to estimate a more accurate global dependencies.

4.4.2 Further Analysis

To analyze the experimental results further, we evaluated separately two types of harder cases of the entity identity task: (1) mentions with *different* writings that refer to the same entity; and (2) mentions with *similar* writings that refer to different entities. Model II and III outperform other models in these two cases as well.

Figure 4.4 presents F_1 performance of different approaches in the first case. The best F_1 value is only 73.1%, indicating that appearance similarity and global dependency are not sufficient to solve this problem when the writings are very different. Figure 4.5 shows the performance of different approaches for disambiguating *similar* writings that correspond to different entities.

Both these cases exhibit the difficulty of the problem, and that our approach provides a significant improvement over the state-of-the-art similarity measure. It also shows that it is necessary to use contextual attributes of the names, which are not yet included in this evaluation.

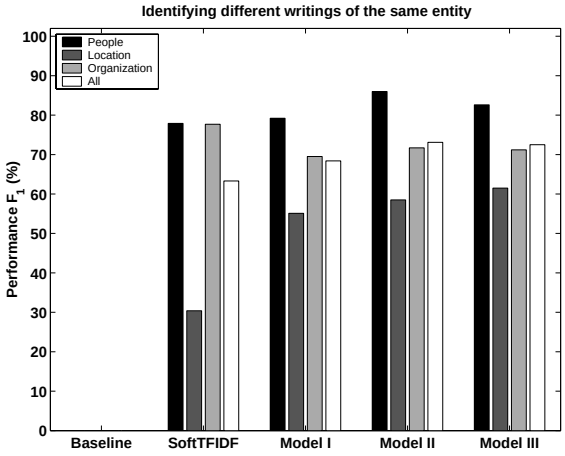

Figure 4.4: **Identifying different writings of the same entity (F_1).** We filter out identical writings and report only on cases of *different* writings of the same entity. The test set contains $46,376$ matching pairs (but in different writings) in the whole data set. The F_1 values of the Baseline algorithm are all zero in this experiment. Baseline, SoftTFIDF, Model I, II and III denote the baseline model, the SoftTFIDF similarity model with clustering, and the three probabilistic models, respectively. The results for each individual entity type and for all entity types are shown in different grey scales.

4.5 Comparison between the Discriminative and Generative Approaches

To further analyze the working mechanisms of the generative and discriminative models, in this section, we compare and evaluate them using the same data sets. Although the models can be trained according to different strategies as described later in detail, they are evaluated using the five standard test sets we constructed in the previous experiments in evaluating the discriminative models (see Section 4.4). The experiments focus on: (1) comparing the generative model with the pairwise classification model in various settings, and (2) combining the unsupervised and supervised models. Here we only test Model II of the three initializations of the generative model since it performs the best in the previous experiments.

We trained the generative model in an unsupervised way with all $8,000$ names. The somewhat surprising results are shown in Table 4.2. The generative model outperformed the supervised classifier for People and Organizations. That is, by incorporating a lot of unannotated data, the un-

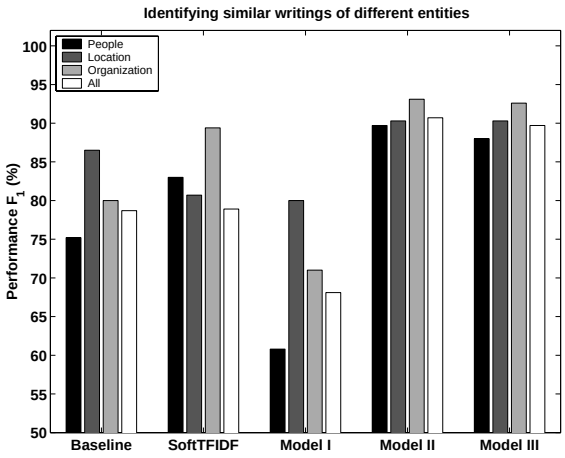

Figure 4.5: **Identifying similar writings of different entities (F_1).** The test set contains $39,837$ pairs of mentions that associated with different entities in the 300 documents and have at least one token in common. Baseline, SoftTFIDF, Model I, II and III denote the baseline model, the SoftTFIDF similarity model with clustering, and the three probabilistic models, respectively. The results for each individual entity type and for all entity types are shown in different grey scales.

$F_1(\%)$	Marlin	SoftTFIDF	LMR	Generative
Peop	88.3	89.0	90.5	**95.1**
Loc	77.3	90.5	92.5	**87.8**
Org	78.1	87.7	93.0	**96.7**

Table 4.2: **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.

supervised learning could do better. Please note that the performance of the generative model over these smaller test sets are better than that over the test using all $8,000$ names (shown in Table 4.1). This is due to different ratios of within-document examples and across-document examples in the test sets and performance over across-document examples is usually lower[7]. Table 4.3 shows the number of these examples in different test sets.

To understand the reasons behind the performance differences between the discriminative and generative approaches, and to compare them further, we addressed the following three issues:

Data: Our first intuition is that the outcome is caused by the fact that the discriminative ap-

[7]The number of across-document examples is $O(m^2)$ where m is the number of documents in the collection, while the number of within-document examples is only $O(m)$.

# of positive examples	600-name test sets	8,000-name test set
within-document	9517	13866
across-document	43346	116948

Table 4.3: **The distribution of within-document and across-document positive examples in different test sets.** Only numbers of positive examples are shown here. The number of positive examples in the 600-name test sets are the sum over 15 sets (five for each entity type).

proach has an inherent limitation. So we increase our training data to a comparable size to that of the generative model: We apply about $6,400$ annotated names for training. The results of the LMR classifier are improved somewhat after this (see Figure 4.6), but still incomparable with the generative model.

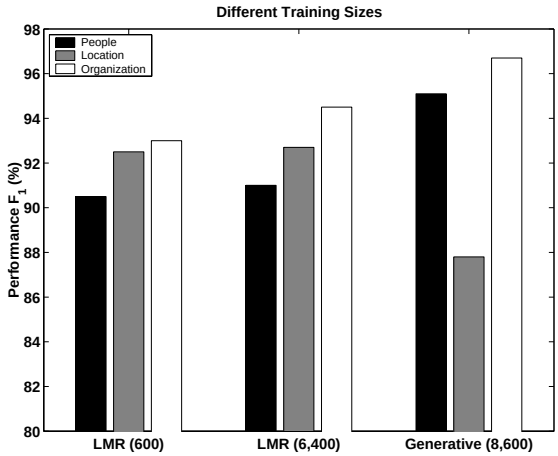

Figure 4.6: **More training data for the discriminative classifier.** Results are evaluated by the average F_1 values over the five test sets for each entity type. "LMR (600)" is the LMR classifier trained only on 600 annotated names and "LMR ($6,400$)" is the one trained on $6,400$ names. Results are averaged over the five test sets.

Learning protocol: A supervised learning approach is trained on a training corpus, and tested on a different one, necessarily, resulting in some degradation in performance. On the contrary, an unsupervised method learns directly on the target corpus. This difference, as we show, can be significant to performance. In a second experiment, in which we do not train the generative model on names it will see in the test set, results clearly degrade (Figure 4.7). Since we use maximum likelihood estimation of model parameters in the Truncated EM algorithm, the results indicate signs of overfitting in the generative model.

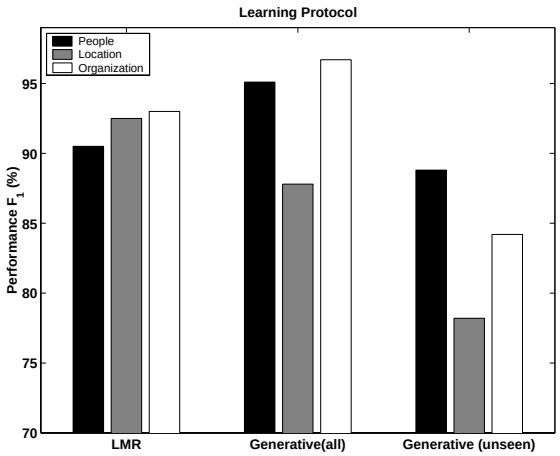

Figure 4.7: **Results of different learning protocols for the generative model.** The table shows the results of our supervised classifier (LMR) trained with 600 names, Generative (all) – the generative model trained with all the $8,000$ names and Generative (unseen) – the generative model trained with 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.

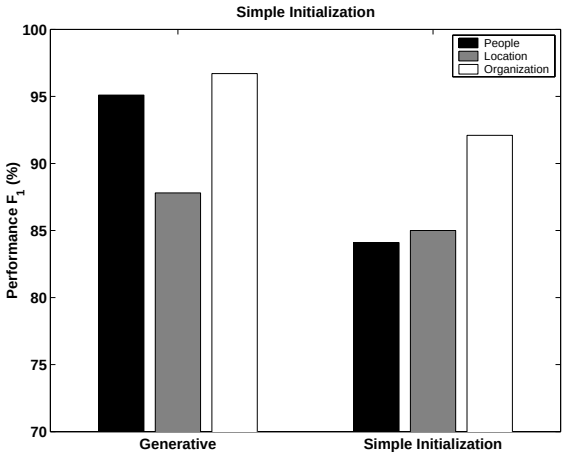

Figure 4.8: **Performance of simple initialization.** "Generative" – the generative model learned in a normal way. "Initial" – the parameters of the generative model initialized using some simple heuristics and used to cluster names. Results are evaluated by the average F_1 values over the five test sets for each entity type.

Structural assumptions: Our generative models benefit from the structural assumptions made in designing the model. We exhibit this by evaluating a fairly weak initialization of the model, and showing that, nevertheless, this results in a *cross-document entity identification* model with respectable results. Figure 4.8 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 models), the generative model can perform reasonably well.

All of the above factors have influenced the results in Table 4.2 more or less, although they do not seem to be dominant factors that cause the performance difference between the discriminative and generative approaches. In the next section, we will further analyze some inherent problems in learning the LMR classifier, which explains why more training data can not make the LMR classifier comparable to the generative model, and show that the generative models indeed have inherent advantages coming together with their structural assumptions.

4.5.1 A Further Explanation

Consider the following examples $< Kennnedy, Kennedy >$ and $< John, Kennedy >$. Suppose the two Kennedy's occur in different documents and they refer to different persons in different, we then have $h^*(Kennedy, Kennedy) = 0$. However, the two names have exactly the same appearance features and there is not way to get a classifier that can tell us they are different by only using these features. On the contrary, suppose "John" and "Kennedy" occur in different documents and they both refer to the person "John F. Kennedy", that is, $h^*(John, Kennedy) = 1$, because their appearance features are completely different, it is hard to train a pairwise classifier to tell that they actually co-refer. When we only want to train a local similarity function for clustering, these examples will be noise in training.

One advantage of the generative model is that it incorporates the information of document structures into the model. The approach, therefore, can distinguish between two levels of name ambiguity: the ambiguity within document and across document. Inside each document, most names belong to the same entity are similar to each other. On the cross-document level, only

representatives which are typically full names of entities are compared and clustered, where most of the cases of “John” and “Kennedy” as above can be avoided. Thus in inference, it will seldom meet the above hard cases. This advantage provides an explanation why the generative models perform better than the LMR classifier learned with supervision.

4.6 Conclusion

This chapter presents an unsupervised learning approach to several aspects of the cross-document entity identification problem. We developed a model that describes the natural generation process of a document and the process of how names are “sprinkled” into them, taking into account dependencies between entities across types and an “author” model.

Several relaxations of this model were developed and studied experimentally, and compared with the previously-developed discriminative model that does not take a global view. The experiments exhibit encouraging results and the advantages of our model.

There are several critical issues that our model can support, but were not included in this preliminary evaluation. Some of the issues that will be included in future steps are: (1) integration with more contextual information (like time and place) related to the target entities, both to support a better model and to allow temporal tracing of entities; (2) the scalability issues in applying the system to large corpora, and the development of an incremental approach of training the model; that is, when a new document is observed, coming, how to update existing model parameters ? and (3) integration of this work with other aspects of general coreference resolution (e.g., other terms like pronouns that refer to an entity) and named entity recognition (which we now take as given).

Chapter 5

Semantic Integration across Text and Databases

Many real-world applications increasingly involve a large amount of both structured data and text. The reason is two-folded: 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 still the *semantic heterogeneity* over concepts, 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 as we have shown before. 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 (Rahm & Do, 2000; Hernandez & Stolfo, 1995b; Sarawagi & Bhamidipaty, 2002; Cohen, Ravikumar, & Fienberg, 2003a). 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 chapter 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 matching), the well-known problem of deciding if two given relational records refer to the same real-world entity (e.g., (Hernandez & Stolfo, 1995b; Sarawagi & Bhamidipaty, 2002; Cohen, Ravikumar, & Fienberg, 2003a)). It is also more general than mention matching in text, as Section 5.2 will discuss.

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 thesis 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 extends the one in Chapter 4 to exploit characteristics of structured database records. It specifies how entity mentions are generated both within a database record or a text document.

Specifically, we make the following contributions:

- An architecture for mentions matching in data sets that involve both text and structured data. The generative 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.

[1]Matching Entities in Data Instances And TExt

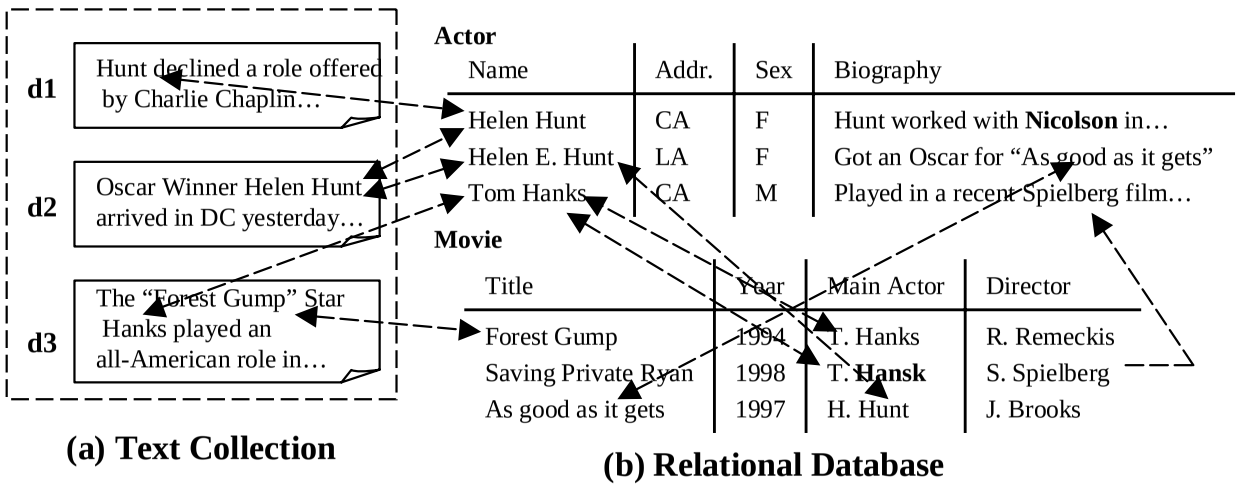

Figure 5.1: **A simplified data set for a movie application, which contains both text and structured data.** The arrows denote semantic matches that we want to establish among mentions of actors and movies.

- 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.

The chapter is organized as follows. The next section defines the mention matching problem. Section 5.2 reviews related work. Section 5.3-5.5 describe the MEDIATE system. Section 5.6 presents experiments and Section 5.7 concludes.

5.1 Problem Definition

We now describe the specific mention matching problem across text and databases, and reuse some of the concepts as we have defined for the related problem in text.

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 5.1 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 5.1, 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., (Agichtein & Ganti, 2004; Borkar, Deshmukh, & Sarawagi, 2001; Freitag, 1998)). 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 5.1). 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, we assume that mentions are already marked up in the data set, and focus on the problem of matching them.

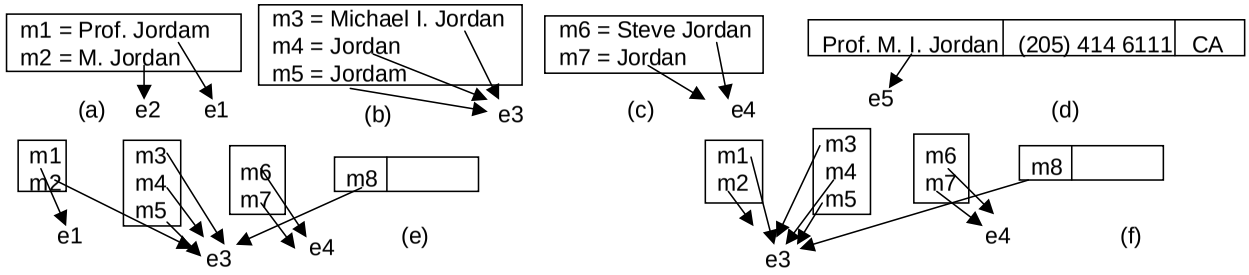

Figure 5.2: **An example of the running process.** The generative model is constructed iteratively by assigning mentions to entities, re-learning model parameters, then re-assigning the mentions.

The Mention Matching Problem: This problem is similar to entity identification problem in text. 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 5.1 shows the links that we want to establish in the above movie data set.

5.2 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 entity identification 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 5.1, 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 demonstrate empirically in Section 5.6 that solving mention matching also improves record linkage accuracy.

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

(Rahm & Bernstein, 2001). 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., (Tejada, Knoblock, & Minton, 2002; Cohen, 1998; McCallum, Nigam, & Ungar, 2000; Yih & Roth, 2002; Bilenko & Mooney, 2002; Ananthakrishna, Chaudhuri, & Ganti, 2002; Sarawagi & Bhamidipaty, 2002; Gravano, Ipeirotis, Koudas, & Srivastava, 2003; Hernandez & Stolfo, 1995b; Galhardas, Florescu, Shasha, & Simon, 2000; Raman & Hellerstein, 2001; Dasu & Johnson, 2003; Rahm & Do, 2000)). For record linkage, early solutions employ manually specified rules (Hernandez & Stolfo, 1995b), while subsequent works focus on learning matching rules from training data (Tejada, Knoblock, & Minton, 2002; Bilenko & Mooney, 2002; Sarawagi & Bhamidipaty, 2002), efficient techniques to match strings (Monge & Elkan, 1996b; Gravano, Ipeirotis, Koudas, & Srivastava, 2003), powerful methods to match entity names (Cohen, 1998; Gravano, Ipeirotis, Koudas, & Srivastava, 2003; Cohen, Ravikumar, & Fienberg, 2003a), scaling up to large number of tuples (Koudas, Marathe, & Srivastava, 2004; Ganti, Chaudhuri, & Motwani, 2005; Jin, Li, & Mehrotra, 2003; McCallum, Nigam, & Ungar, 2000; Cohen & Richman, 2002a), matching in online contexts (Chaudhuri, Ganjam, Ganti, & Motwani, 2003), personal information management (Dong, Halevy, Madhavan, & Nemes, 2005), matching XML data (Weis & Naumann, 2005), and exploiting links (Bhattacharya & Getoor, 2004).

Several recent works have also developed generative models to match mentions. The work (Pasula, Marthi, Milch, Russell, & Shpitser, 2003) addresses citation matching in structured contexts, a much narrower problem. It proposes a full-blown probabilistic relational model (Friedman, Getoor, Koller, & Pfeffer, 1999), 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 (Ravikumar & Cohen, 2004) for matching tuples is much more efficient, but does not capture and exploit the notion of real-world entities, as we do here.

Several recent works have employed another probabilistic framework called *conditional ran-*

dom fields (CRF) to match mentions (Parag & Domingos, 2004; Wellner, McCallum, Peng, & Hay, 2004). In particular, (Wellner, McCallum, Peng, & Hay, 2004) 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 (Pasula, Marthi, Milch, Russell, & Shpitser, 2003; Ananthakrishna, Chaudhuri, & Ganti, 2002; Bhattacharya & Getoor, 2004; Parag & Domingos, 2004; Wellner, McCallum, Peng, & Hay, 2004). (Ananthakrishna, Chaudhuri, & Ganti, 2002) 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 (Bhattacharya & Getoor, 2004; Parag & Domingos, 2004; Wellner, McCallum, Peng, & Hay, 2004) 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.

5.3 The MEDIATE Approach

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. Here we adopt the same set of notations we have used for entity identification in text in Section 4.1.

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-

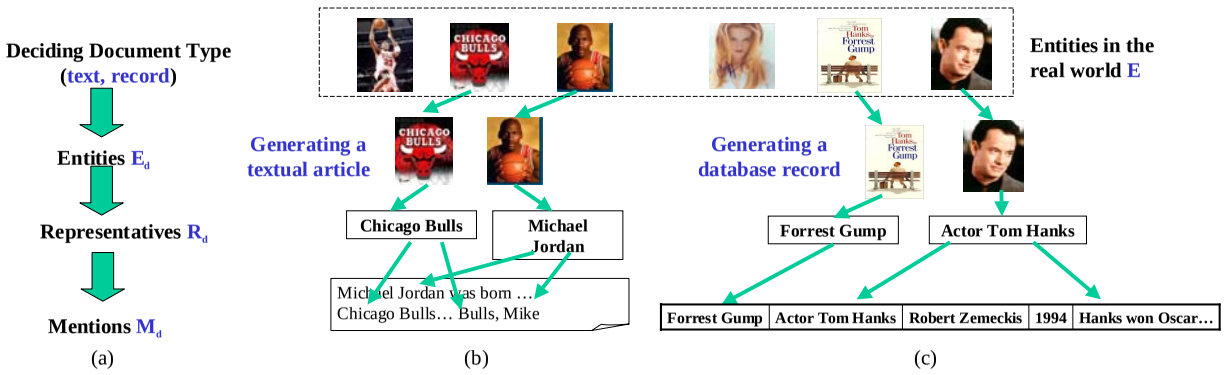

Figure 5.3: **Generating database records and text documents.**

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".

5.3.1 Constructing the ME Generative Model

Example 5.3.1 *Figure 5.3.a shows the document generation process, while Figures 5.3.b-c show specific examples of generating a text document and a relational record. For instance, Figure 5.3.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.*□

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

$$\begin{aligned} P(d) &= P(E_d, R_d, M_d) = P(E_d)P(R_d|E_d)P(M_d|R_d) \\ &\approx \prod_{i=1}^{|E_d|=l_d} [P(e_{di})P(r_{di}|e_{di})] \prod_{(r_{dj}, m_{dj})} P(m_{dj}|r_{dj}). \end{aligned} \tag{5.1}$$

Since the ME model is not much different from the generative model we used in entity identification in text, we learn the model in the same way as in Section 4.3. That is, if we have a set 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: $\theta^* \equiv argmax_\theta P(D_t|\theta)$.

5.3.2 An Example

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". This model is very similar to the model I developed in Section 4.2.3 for text Figures 5.2.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 5.2 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 5.2.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

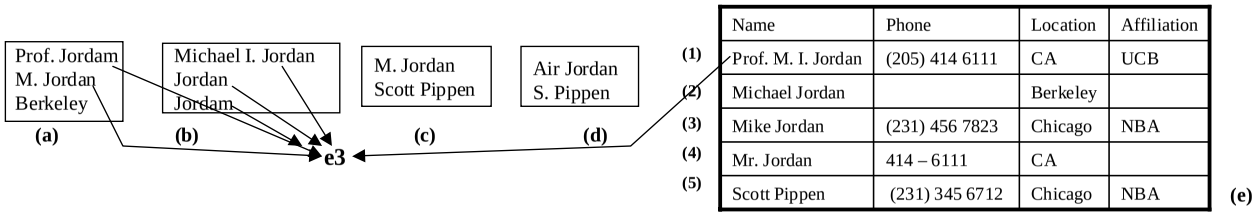

Figure 5.4: **Examples of exploiting external attributes (e.g., phone, location), co-occurrence of entities, and transferring knowledge across matches.**

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 5.2.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.

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.

5.4 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 5.4.

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 5.4). 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.

Interplay between Text & Databases The example in Figure 5.4 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 before 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.

5.4.1 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 Section 5.3) 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. The independence assumption clearly does not hold, but it reduces runtime complexity, and is shown empirically to work well (Section 5.6).

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 $d_k(v'_k, v_k) = d_k$ using an attribute-specific distance metric. As default metrics, for textual attributes, we convert the SoftTFIDF (Cohen, Ravikumar, & Fienberg, 2003a) 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):

$$f_k(v'_k|v_k) \equiv \frac{1}{\sqrt{\pi/2}\sigma_k} \cdot \exp\left(-d_k^2/\sigma_k^2\right) \tag{5.2}$$

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

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

5.4.2 Exploiting Entity Co-occurrence

Exploiting entity co-occurrence further improves matching accuracy 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 5.3.1), 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

$$P(d) \approx \prod_{i=1}^{|E_d|}[P(e_{di}|E_d^{i-1})P(r_{di}|e_{di})] \times \prod_{(r_{dj},m_{dj})} P(m_{dj}|r_{dj}). \tag{5.3}$$

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 (see Section 4.3.1), to compute $P(e_{di}|E_d^{i-1})$.

Integrating as an External Attribute: We also integrate entity co-occurrence 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 (Cohen, Ravikumar, & Fienberg, 2003a) 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

$$d_{\mathsf{con}}(r, e) \equiv \prod_{n_i \in \mathsf{con}(r)} [1 - max_{n'_j \in \mathsf{con}(e)} s(n_i, n'_j)] \tag{5.4}$$

The probability of the context of a representative being transformed from that of an entity is still computed by Equation 5.2. Note that we do not expand the representation of a mention since this co-occurring information does not benefit the mention level.

5.5 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.

As we have motivated above, 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: $d_{\mathsf{con}}(r, e) \equiv \sum_l w_l \cdot d_{c_l}(r, e)$, where $d_{c_l}(r, e)$ is defined as a distance between two sets of names, measured in the same way as in Equation 5.4. 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.

5.6 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.

5.6.1 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 5.5.

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 5.5.

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 (Hernandez & Stolfo, 1995b; Ananthakrishna, Chaudhuri, & Ganti, 2002), 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 $\rightarrow$ 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 5.5), and dropping others. We also perturbed the DBLP table by randomly removing middle names, and abbreviating first names.

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 5.5: **Characteristics of the data sets.**

Figure 5.5 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 5.6.2- 5.6.4. In Section 5.6.5, 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 (Cohen, Ravikumar, & Fienberg, 2003a) and shown empirically to be the best among several.

- *Clustering:* Many different clustering algorithms have been developed for record linkage (e.g., (McCallum, Nigam, & Ungar, 2000; Cohen & Richman, 2002a)), as well as mention matching in text (Li, Morie, & Roth, 2004a). We implemented a variation of these algorithms, using the SoftTF-IDF measure (Cohen, Ravikumar, & Fienberg, 2003a) 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 5.2), 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 (Hernandez & Stolfo, 1995b; Sarawagi & Bhamidipaty, 2002; Cohen, Ravikumar, & Fienberg, 2003a). 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. (Sarawagi & Bhamidipaty, 2002). 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 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 as we did in Section 2.2.2.

5.6.2 Overall Matching Accuracy

Table 5.1: **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 5.1 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 5.2 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 5.2: **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", "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.

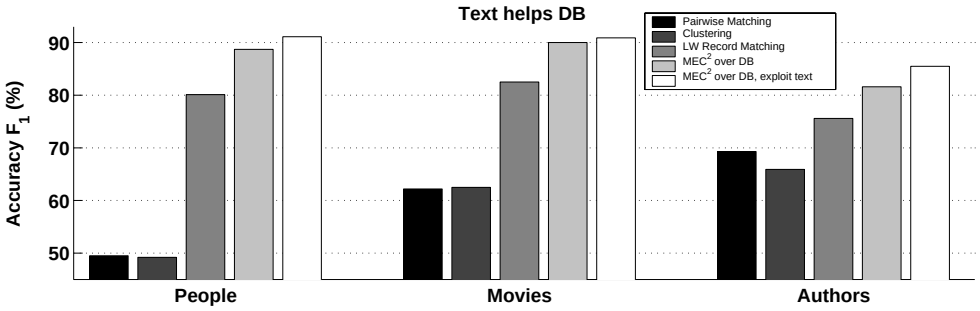

Figure 5.6: **achieves significantly higher accuracy than LW record linkage when applied to databases, and obtains even higher accuracy when exploiting text.**

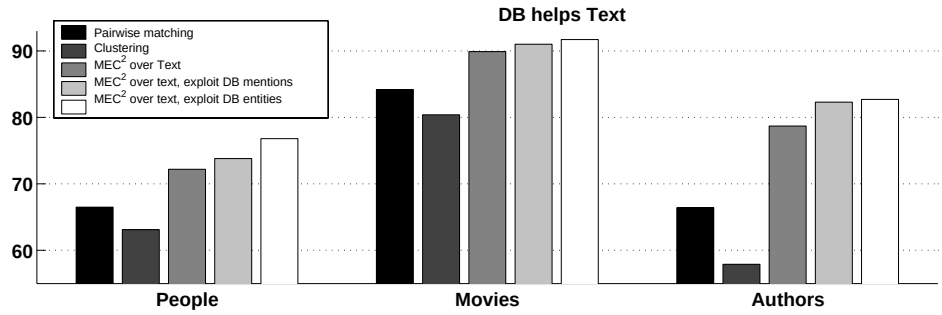

Figure 5.7: **can exploit databases to improve accuracy over text.**

5.6.3 Exploit Text to Improve Record Linkage

Figure 5.6 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 5.5) 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%.

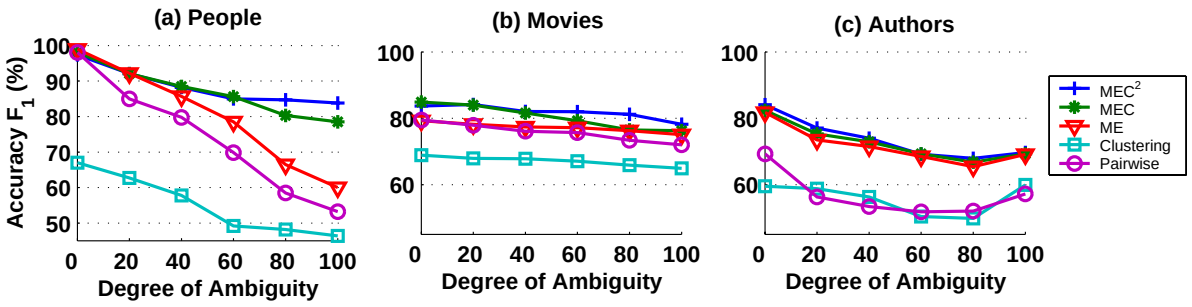

Figure 5.8: **The MEDIATE system is robust across a broad range of degrees of semantic ambiguity.**

5.6.4 Exploit DBs to Match Text Mentions

Figure 5.7 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 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%.

5.6.5 Sensitivity Analysis

Figure 5.8 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.

5.7 Conclusion

This chapter 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. MEDIATEis created based on the generative models described in Chapter 4, but allow several extensions of it in the context of a structured database. 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.

Chapter 6

More about Concept-Based Text Understanding and Mining

Entity disambiguation and identification is a critical step towards implementing concept-based text understanding mining. Based on the work of globally identifying real-world entities from a large collection of text, we aim at building a unified framework to support intelligent access to textual information, as shown in Figure 6.1. The hope is to provide a variety of inference and access functionalities for users and other text-related tasks such as information retrieval, information extraction and question answering.

In this framework, after entities and concepts are recognized from a large collection of text (e.g. the collection of all the online the web pages), indexing is created to link each of them to all of its occurrences in text. This is similar to the indexing mechanism implemented in most search engines, but indexing here takes entities as the basic unit rather than tokens. Moreover all the variations of the entity name have been identified from text, and are indexed together. In addition to indexing, meta-information about each entity, such as their occurrence frequency in all the text, other entities that are closely related to them, and other facts and events about them that can be extracted from the text, are put into the knowledge base. Users, including those end users who want to search for knowledge about entities as those of search engines, and the higher-level text understanding and mining systems, can access the indexed and integrated information in this knowledge base through multiple access functions. These functions involve direct concept query,

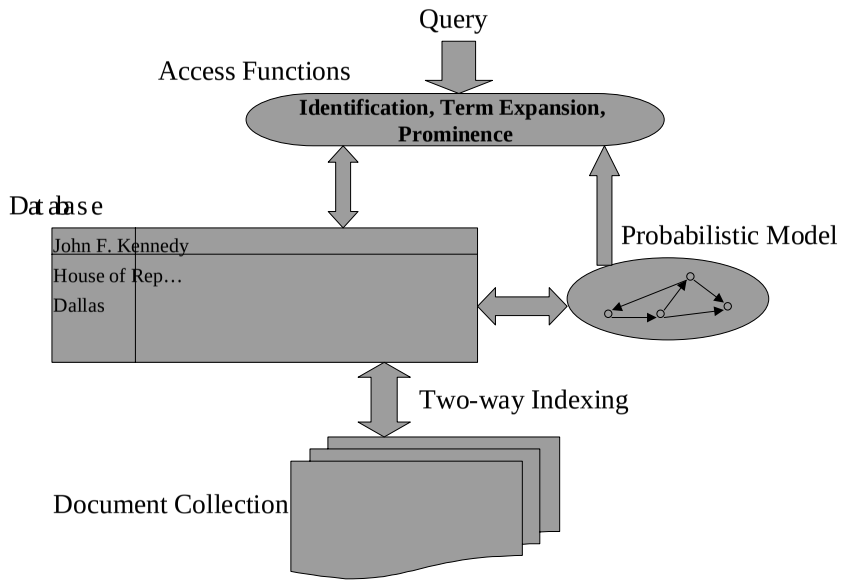

Figure 6.1: **A knowledge base for intelligent access to text**

as well as some inference modules based on some probabilistic model learned from the text in a way as described in Chapter 4, and updated when new text come in, e.g. daily news articles.

In this chapter, we first discuss in Section 6.1, using search engines as an example, what the potential functionalities this framework can support, and how text-related tasks can move from string- and mention-level processing to processing real-world concepts directly and benefit from it, without being bothered by the name ambiguity. After that in Section 6.2, we will discuss some future work in implementing this framework to support concept-based text understanding and mining , involving both scalability issues and a further task of coreference resolution – identifying entities from other types of references like pronouns. In the end, we will conclude this thesis in Section 6.3.

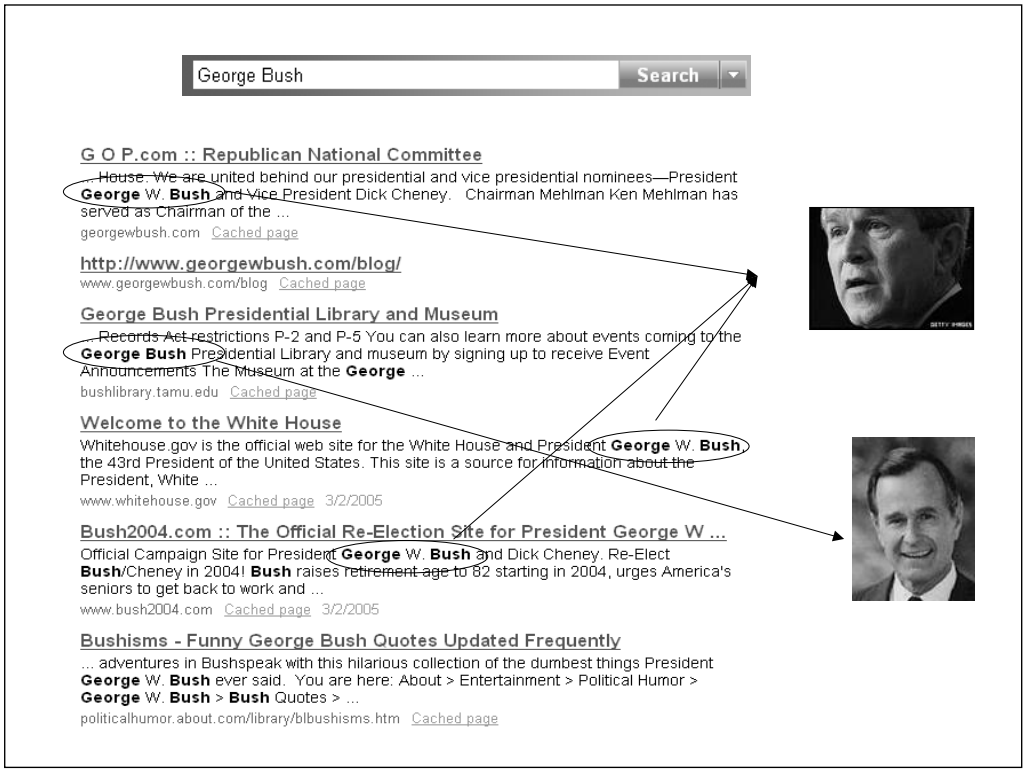

Figure 6.2: **Search by concept**

6.1 A Case Study with Search Engines

We use search engines, as an example of concept-based text understanding and mining tasks, to show how text-related tasks can benefit from entity identification and disambiguation. Let's first take a look at an example in Figure 6.2, to see what are the setbacks of the current search engine techniques.

Suppose a user is looking for information about the senior president George Bush and query a search engine with his name. A standard search engine that is based on keyword-matching, outputs a collection of "relevant" web pages to the user. All of the output pages contain both the query terms "George" and "Bush", but some of them are actually related to the son of the senior president, George W. Bush, which is not the one the user is looking for. That is, name ambiguity can bring noise to the search engines. More interestingly, since his son is the incumbent President, whose names are supposed to occur frequently in recent web pages, more pages about him are output. A

more significant setback of this keyword-matching scheme, is that when a web page only contains the name "Bush", even if it refers to the same person the user is looking for, the search engine will not treat it as a relevant page. Moreover, different web pages about the same person are scattered in the result, and it is up to the user to further identify the truly "relevant" information.

Assume entity identification and indexing have been implemented as a component of a search engine. After all the occurrences of entities, such as people, locations, and companies, are identified and disambiguated in the web pages, a search engine could benefit from it in at least five aspects: (1) expanding queries of names; (2) ranking prominence; (3) creating more accurate relevance ranking; (4) clustering relevant documents based on entities; and (5) providing more context to guide iterative search.

They are further discussed in the rest of this section, and some of them formalized as inference tasks based on the generative model described in Chapter 4. In the preliminary experiments, we evaluate our generative model on these tasks related to the cross-document entity identification problem, but present results only for Model II as described in Section 4.2.3.

6.1.1 Name Expansion

The problem of *Name Expansion* in a search engine is that, given a name of an entity (say, in a question), find other likely names of the same entity. An inference task can be defined to address it, based on the probabilistic model as described in Chapter 4. That is, given a mention m_q in a query q, decide whether mention m in the document collection D is a 'legal' expansion of m_q:

$$m_q \rightarrow m \text{ iff } e^*_{m_q} = argmax_{e \in E} P(E_q, R_q, M_q) \;\&\; m \in mentions(e^*)$$

We assume here that we already know the possible mentions of e^* after learning the models in D.

In the following preliminary experiments, given a mention m in a query (for example, in an q), we find the most likely entity $e \in E$ for m using our inference algorithm. All unique mentions of the entity in the documents are output as the expansions of m. The accuracy of *Name Expansion*

for one mention in a query is defined as the percentage of correct expansions among all expansions output for a query. The average accuracy of Name Expansion of Model II is shown in Table 6.1 (averaged over 30 queries for each of the three entity types). Here is an example of a query:

Query: Who is *Gore* ?

Expansions: Vice President Al Gore, Al Gore, Gore.

Entity Type	People	Location	Organization
Accuracy(%)	90.6	100	100

Table 6.1: **Accuracy of name expansion.** The accuracy of Name Expansion for one mention in a query is defined as the percentage of correct expansions among all expansions output for a query. Accuracy is averaged over 30 randomly chosen queries for each entity type.

6.1.2 Prominence

The problem *Prominence* is that: given a question "What is Bush's foreign policy?", and given that any large collection of documents may contain several Bush's, there is a need to identify the most prominent, or relevant "Bush", based on a ranking of the prominence of entities, perhaps taking into account also some contextual information. The inference task for this problem, based on the probabilistic model as described in Chapter 4, can be formalized as: given a name $n \in W$, the most prominent entity for n is given by:

$$e^* = argmax_{e \in E} P(e)P(n|e).$$

$P(e)$ is given by the prior distribution P_E and $P(n|e)$ is given by the appearance model.

We refer to Example 4.2.1 and use it to exemplify qualitively how our system supports prominence ranking. The following examples show the ranking of entities with regard to the value of $P(e) \cdot P(m|e)$ using Model II, given a query name m.

Input: George Bush

1. George Bush 2. George W. Bush

Input: Bush

1. George W. Bush 2. George Bush 3. Steve Bush

6.1.3 Other Applications

Creating more accurate relevance ranking. One of the most important statistics used in ranking web pages is term frequency – counting how many times a query term occurring in a web page to be ranked. Since names are also split into individual query terms in the current search engine techniques, and names of the same entity could be very ambiguous in text as we have claimed, the statistics is not very accurate, with regard to the importance of a name in ranking the page. This problem has two-folded influence in reality. On one hand, if a web page only contains the term "Bush" rather than "George", the term frequency for "George" will be zero, resulting in a very low relevance for this page. On the other hand, when multiple entities share the same name, term frequency over each token of the name will mistakenly accumulate over all these different entities.

One solution to these setbacks is to take entity-based frequency statistics, rather than token-base frequency statistics. For user queries with a name, the real-world entity behind this name is first identified, given the other context of the query. The frequency statistics is taken based on the identified entity according to how many times this entity is mentioned in it, no matter which name of it is used. The frequency of an entity in a collection of pages is computed correspondingly, as how many times its names occur in all these pages.

Clustering relevant documents based on entities. The layout of the search results could also be reorganized, base on identified entities. When there are multiple possible entities for a query, (e.g. searching for "Bush"). A search engine could first returns a list of candidate Bush's (such as "George Bush", "David Bush") as the first-tier outcome. A short biography is provided for each person and can be used to help the user decide the one he is looking for. Moreover, web pages about the different persons can be split apart into different groups. The user can choose a entity here, and then go to its relevant pages for more relevant information. Some further categorization and summarization of the web pages about the same entity, can be performed to avoid redundant

output.

Providing more context to guide iterative search. After entities are identified in text, further knowledge can be extracted for each entity based on straightforward statistics over the entities. One type of the most important knowledge for an entity is the other "related" entities, that frequently occur with one entity in different texts. "George W. Bush", as an instance, can be easily identified relating to the following entities, such as "White House", "Dick Cheney", "Iraq". The related entities can provide a user some context of the target entity that he is searching for. He may then follow the relation to find information about other entities, and get the ultimate information he is interested with, in an iterative setting.

6.2 Future Work

For implementing the framework to support intelligent access to textual information as in Figure 6.1, there are still many issues to be addressed in the future.

6.2.1 the Scalability Problem

The practical system of concept disambiguation and tracing of ambiguous names, that can work in the domain of news articles and web pages, should be capable of efficiently processing of a great number of documents, even millions of documents, with reasonable computing resources. However, the current implementation of the discriminative and generative approaches as described in Chapter 2 – Chapter 4, have about $\Omega(N^2)$ time complexity and $\Omega(N)$ space complexity, where N is the number of documents. In the 300 documents that we have experimented with, there are $8,000$ names which correspond to $2,000$ entities. It takes about hours to train the pairwise classifier and one or two hours to train our generative model (mode II). Therefore it is impossible form them to handle a much larger document collection.

There are two possible solutions to this problem: (1) hierarchically clustering documents; and

(2) incrementally training the generative models and building the knowledge base.

Our current generative models are hierarchical models in some sense. When training the model parameters, the observed mentions are clustered into groups within each document, and a representative is selected for each group. Only representatives participate in the global clustering on the cross-document level. This scheme avoids direct clustering of all mentions and computing pairwise similarities for all pairs of mentions. The same idea can be extended further to include more levels in the training and inference process of the generative models because documents are naturally clustered corresponding to different dates, sources and so on, mentions can be clustered step by step according to the common properties they or the documents have.

The second solution is to train the generative models and to build the knowledge base incrementally. The parameters of the generative models and the content in the knowledge depend on a set of statistics over entities, entities dependencies and features in the appearance models, which could be updated in an incremental way. For example, if the frequency of each entity occurring in processed documents has been maintained, when new documents come in, entities are identified using the inference algorithm with the existing model parameters. If necessary, new entities will be created. Then the frequency can be easily updated then by incorporating statistics over the new documents. After some time, all the parameters in the generative models can be re-estimated based on the new statistics.

An ideal implementation of the system should integrate both of the above strategies in a reasonable way.

6.2.2 Coreference Resolution

In addition to proper names, coreference resolution cares about other types of reference ambiguity to real-world concepts as well, such as nominal and pronominal references. Consider the following simple story (See Figure 6.3) and some related comprehension questions (Hirschman, Light, Breck, & Burger, 1999):

What is involved in being able to understand this story and to answer the following questions ?

(ENGLAND, June, 1989) - ***Christopher Robin*** *is alive and well.* ***He*** *lives in England.* ***He*** *is the same person that you read about in the book, Winnie the Pooh. As a boy,* ***Chris*** *lived in a pretty home called Cotchfield Farm. When* ***Chris*** *was three years old,* ***his father*** *wrote a poem about* ***him****. The poem was printed in a magazine for others to read.* ***Mr. Robin*** *then wrote a book.* ***He*** *made up a fairy tale land where* ***Chris*** *lived.* ***His*** *friends were animals. There was a bear called Winnie the Pooh. There was also an owl and a young pig, called a piglet. All the animals were stuffed toys that Chris owned.* ***Mr. Robin*** *made them come to life with* ***his*** *words. The places in the story were all near Cotchfield Farm. Winnie the Pooh was written in 1925. Children still love to read about* ***Christopher Robin*** *and* ***his*** *animal friends. Most people don't know* ***he*** *is a real person who is grown now.* ***He*** *has written two books of* ***his*** *own. They tell what it is like to be famous.*

Question:
1. Who is Christopher Robin?
2. When was Winnie the Pooh written?
3. What did Mr. Robin do when Chris was three years old?
4. Where did young Chris live?
5. Why did Chris write two books of his own?

Figure 6.3: **An example of coreference ambiguity in text.** Noun phrases that refer to "Christopher Robin" or his father are in bold.

Clearly, there are many small local decisions need to make. We need to recognize that there are two Chris's here. Mentions of "Christopher Robin" and his father interweave in the above story and occur in different forms. "Christopher Robin" is referred as $\{ChristopherRobin, he, his, Chris\}$ while his father is referred as $\{Mr.Robin, he, hisfather, his\}$. Coreference resolution is crucial here to automatically identify all types of occurrences of concepts, not just proper names, and relations between them.

In solving the coreference ambiguity of proper names, we have concentrated more on the appearance similarity of different names and the co-occurring dependency among entities. In coreference resolution, however, the usage of the nominal and pronominal references depend more on the lexical semantics of noun phrases, syntactic structures of a sentence and contextual information around a reference. For this reason, modelling the problem and seeking neat solution is much harder. An example of the more complex coreference ambiguity in the above article is the use of pronouns: "he" and "His" are used many times in the story and they can both refer to either of two Chris's in different contexts.

Most works in this area only focus on noun phrase coreference within documents. Many machine learning approaches (Carbonell & Brown, 1988; Dagan & Itai, 1990; Aone & Bennett, 1995; McCarthy & Lehnert, 1995; Ge, Hale, & Charniak, 1998; Cardie & Wagstaff, 1999; Florian, Hassan, Ittycheriah, Jing, Kambhatla, Luo, Nicolov, & Roukos, 2004) have been applied in determining whether a pair of NPs refers to the same entity based on local context of them in a document. The contextual information of a noun phrase is converted into a set of features. For example, (Soon, Ng, & Lim, 2001) applies decision tree induction based on 12 features types to two standard coreference data sets (MUC-6, 1995; MUC-7, 1999).

The influence of these works is two-folded. On one hand, their results have shown the importance of features in this task. By using an exhaustive set of lexical, grammatical, semantic and positional features, their system can achieve 70.4% and 63.4% F-measure on MUC-6 and MUC-7 corpus respectively, which is a significant progress from the 64.3% and 61.2% reported in (Soon, Ng, & Lim, 2001). On the other hand, the result is also an indication of the hardness of this problem. By applying most of currently known features — many of which are even hand-selected, the performance is still far from satisfactory. This hardness motivates us to turn to better modelling of the problem in the future, instead of focusing on feature engineering.

6.3 Conclusions of the Thesis

Our major conclusion in this thesis is that: semantic understanding of text and intelligent access to textual information require *concept-based text understanding and mining*, that is, a framework of organizing, indexing, accessing textual information centered around real-world concepts, and a mechanism of analyzing and integrating segregated information. This framework consists of several steps: (1) recognizing occurrences of real-world concepts and entities from text; (2) identifying real-world concepts from their ambiguous occurrences in text; (3) integrating and organizing textual information based on concepts; and (4) extract their properties, relations and other facts or knowledge based on this integrated organization.

After all the occurrence of a real-world entity has been identified, information related to this entity which are previously scattered in different texts and different context of the same text, can be indexed and integrated together. A lot of text-related applications such as Information Retrieval, Information Extraction Question Answering, Text Summarization and Reading Comprehension, can be improved by directly working on the concept-level, rather than being bothered by ambiguous names and different occurrences of the same entity.

In this thesis, we describe our effort in one of the above fundamental steps – disambiguation and identification of entities (people, locations, organizations and so on) from their ambiguous writings of names, in the across-document setting of text. While semantic integration of structured information has been widely studied, little attention has been paid to a similar problem in unstructured and semi-structured data. This paper also describes one of the first efforts towards sematic integration in unstructured textual data, providing a promising perspective on the integration of structured databases with unstructured or semi-structured information. We propose multiple machine learning techniques to address the entity identification and semantic integration problem. It has been shown that as more information can be exploited, the learning techniques developed accordingly, can continuously improve the identification accuracy.

Our first solution (described in Chapter 2) is a discriminative approach for studying the influence of appearance similarity between names in entity identification. This approach models the problem as deciding whether any two names mentioned in a collection of documents represent the same entity. This is a standard pairwise classification task, under a supervised learning protocol; our main contribution in this part is to show how relational – string and token-level features – and structural features, representing transformations between names, can significantly improve the performance of this classifier. We also show that the appearance similarity between names are critical information in entity identification, and the classifier based on it, without any help from contextual information, can already achieve decent performance.

Our second approach in Chapter 3, is a new clustering framework, that can make global optimization by making decisions over a set of names together in entity identification. The proposed

supervised discriminative clustering framework (SDC) targets learning a partition function, parameterized by any chosen clustering algorithm, to minimize the clustering distortion from given supervision. Our experiments on entity identification task show that SDC which trains a similar metric for a chosen clustering can significantly outperforms the pairwise classification approach, and existing clustering approaches, and other metric learning approaches where clustering is disjoint from the metric learning procedure. This new clustering framework is very promising to be applied a broad of problems in natural language processing, and data mining domains.

In Chapter 4, we develop a global probabilistic model to exploit more contextual information for *Entity Identification*, at the heart of which is a view on how documents are generated and how names (of different entity types) are "sprinkled" into them. This unsupervised approach can outperform the supervised pairwise classifier in the experiments, an indication of the advantages of more contextual information in this task, such as concurring entities, and the notion of documents.

In addition to developing more advanced learning techniques that can effectively exploiting more information, we also extend our global probabilistic model to address another related problem – semantic integration between text and databases in Chapter 5. This is a very significant problem since there are many important applications require integration of structured databases with a greater amount of unstructured text, which can provide both efficient access to textual information, and expansion to databases with more related and integrated textual information.

Based on the work of globally identifying real-world entities from a large collection of documents (for example, everyday news articles or the whole set of online web pages), our ultimate goal is to design and implement a unified framework for intelligent access of textual information. For this purpose, in Chapter 6 we study several applications of entity identification in a text-mining task like search engine, and show how text-related tasks can benefit significantly from concept-level understanding and mining.

Bibliography

Agichtein, E., & Ganti, V. (2004). Mining reference tables for automatic text segmentation.

Ananthakrishna, R., Chaudhuri, S., & Ganti, V. (2002). Eliminating fuzzy duplicates in data warehouses.

Aone, C., & Bennett, S. W. (1995). Evaluating automated and manual acquisition of anaphora resolution strategies.

Bach, F. R., & Jordan, M. I. (2003). Learning spectral clustering.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison Wesley Longman.

Bagga, A., & Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In *the 17th international conference on Computational linguistics* (pp. 79–85). Association for Computational Linguistics.

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2003). Learning distance functions using equivalence relations.

Bhattacharya, I., & Getoor, L. (2004). Iterative record linkage for cleaning and integration.

Bickel, P. J., & Doksum, K. A. (1977). *Mathematical statistics: Basic ideas and selected topics*. San Francisco: Holden-Day.

Bilenko, M., Basu, S., & Mooney, R. (2004). Integrating constraints and metric learning in semi-supervised clustering.

Bilenko, M., & Mooney, R. (2002, February). *Learning to combine trained distance metrics for duplicate detection in databases* (Technical Report Technical Report AI 02-296). Austin, TX: Artificial Intelligence Laboratory, University of Texas at Austin.

Bilenko, M., & Mooney, R. (2003). Adaptive duplicate detection using learnable string similarity measures.

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., & Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*.

Borkar, V., Deshmukh, K., & Sarawagi, S. (2001). Automatic text segmentation for extracting structured records.

Borovkov, A. A. (1984). *Mathematical statistics*. Mir,Moscow.

Bradley, P. S., Fayyad, U. M., & Reina, C. (1998). Scaling clustering algorithms to large databases.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics, 21*(4), 543–565.

Brill, E. (1997). Unsupervised learning of disambiguation rules for part of speech tagging. Kluwer Academic Press.

Brown, P., deSouza R. Mercer, P., Pietra, V., & Lai, J. (1992). Class-based n-gram models of natural language. *Computational Linguistics, 18*(4).

Califf, M., & Mooney, R. (1999). Relational learning of pattern-match rules for information extraction.

Carbonell, J. G., & Brown, R. D. (1988). Anaphora resolution: A multi strategy approach.

Cardie, C., & Wagstaff, K. (1999). Noun phrase coreference as clustering. In *Proceedings of The Conference on Empirical Methods in Natural Language Processing* (pp. 82–89).

Carlson, A., Cumby, C., Rosen, J., & Roth, D. (1999, May). *The SNoW learning architecture* (Technical Report UIUCDCS-R-99-2101). UIUC Computer Science Department.

Charniak, E. (1993). *Statistical language learning*. MIT Press.

Charniak, E. (1997). Statistical techniques for natural language parsing. *The AI Magazine*.

Charniak, E. (2000). A maximum entropy-inspired parser. In *Proceedings of North American chapter of the Association for Computational Linguistics annual meeting* (pp. 132–139).

Chaudhuri, S., Ganjam, K., Ganti, V., & Motwani, R. (2003). Robust and efficient fuzzy match for online data cleaning.

Chieu, H., & Ng, H. (2002). A maximum entropy approach to information extraction from semi-structure and free text. In *the Eighteenth National Conference on Artificial Intelligence (AAAI 2002)* (pp. 786–791).

Chu, S. C., Roddick, J. F., & Pan, J. S. (2001). A comparative study and extensions to k-medoids algorithms.

Cohen, W. (1998). Integration of heterogeneous databases without common domains using queries based onb textual similarity.

Cohen, W., Ravikumar, P., & Fienberg, S. (2003b). A comparison of string metrics for matching names and records.

Cohen, W., Ravikumar, P., & Fienberg, S. (2003a). A comparison of string metrics for name-matching tasks.

Cohen, W., & Richman, J. (2002a). Learning to match and cluster entity names.

Cohen, W., & Richman, J. (2002b). Learning to match and cluster large high-dimensional data sets for data integration.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing.

Collins, M. (1999). Head-driven statistical models for natural language parsing. *PhD thesis, University of Pennsylvania*.

Collins, M., & Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and voted perceptron.

Collins, M., & Singer, Y. (1999, June). Unsupervised models for name entity classification.

Dagan, I., & Itai, A. (1990). Automatic acquisition of constraints for the resolution of anaphora references and syntactic ambiguities. In *Proceedings of International Conference on Computational Linguistics*, Volume 3 (pp. 330–332).

Dagan, I., Karov, Y., & Roth, D. (August 1997). Mistake-driven learning in text categorization. In *EMNLP-97, The Second Conference on Empirical Methods in Natural Language Processing* (pp. 55–63).

Dagan, I., Lee, L., & Pereira, F. (1999). Similarity-based models of word cooccurrence probabilities. *Machine Learning*, *34*(1-3), 43–69.

Dasu, T., & Johnson, T. (2003). *Exploratory data mining and data cleaning*. John Wiley and Sons.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *39(B)*, 1–38.

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

Dong, X., Halevy, A., Madhavan, J., & Nemes, S. (2005). Reference reconciliation in complex information spaces.

Durban, R., Eddy, S. R., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis - probabilistic models of proteins and nucleic acids.* Cambridge University Press.

Even-Zohar, Y., & Roth, D. (2001). A sequential model for multi-class classification. In *EMNLP-2001, the SIGDAT Conference on Empirical Methods in Natural Language Processing* (pp. 10–19).

Florian, R., Hassan, H., Ittycheriah, A., Jing, H., Kambhatla, N., Luo, X., Nicolov, N., & Roukos, S. (2004). A statistical model for multilingual entity detection and tracking.

Freitag, D. (1998). Multistrategy learning for information extraction.

Freitag, D. (2000). Machine learning for information extraction in informal domains. *Machine Learning Journal*, *39*(2/3), 169–202.

Freund, Y., & Schapire, R. (1998). Large margin classification using the Perceptron algorithm.

Frey, B. J., & Jojic, N. (2003). Transformation-invariant clustering using the EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(1).

Friedman, N. (1998). The Bayesian structural EM algorithm.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models.

Fuhr, N. (1992). Probabilistic models in information retrieval. *The Computer Journal*, *35*(3), 243–255.

Fuhr, N. (2001, May 31 – June 1). Language models and uncertain inference in information retrieval. Extended abstract.

Galhardas, H., Florescu, D., Shasha, D., & Simon, E. (2000). An extensible framework for data cleaning.

Ganti, V., Chaudhuri, S., & Motwani, R. (2005). Robust identification of fuzzy duplicates.

Ge, N., Hale, J., & Charniak, E. (1998). A statistical approach to anaphora resolution. In *the Sixth Workshop on Very Large Corpora (COLING-ACL 98)* (pp. 161–170).

George, K. (2003). *Cluto: A clustering toolkit* (Technical Report). Dept of Computer Science, University of Minnesota.

Golding, A. R., & Roth, D. (1999). A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, *34*(1-3), 107–130. Special Issue on Machine Learning and Natural Language.

Gooi, C., & Allan, J. (2004). Cross-document coreference on a large scale corpus.

Gravano, L., Ipeirotis, P., Koudas, N., & Srivastava, D. (2003). Text join for data cleansing and integration in an rdbms.

Hernandez, M., & Stolfo, S. (1995a). The merge/purge problem for large databases.

Hernandez, M., & Stolfo, S. (1995b). The merge/purge problem for large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 127–138).

Hindle, D. (1990). Noun classification from predicate-argument structures. In *Proceedings of the Annual Meeting of The Association for Computational Linguistics* (pp. 268–275).

Hirschman, L., Light, M., Breck, E., & Burger, J. (1999). Deep read: A reading comprehension system.

Hovy, E., Gerber, L., Hermjakob, U., Lin, C., & Ravichandran, D. (2001). Toward semantics-based answer pinpointing.

J. Hartigan, M. W. (1979). A k-means clustering algorithm. *Applied Statistics*, *28*(1).

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, *31*(3).

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, *84*, 414C420.

Jaro, M. A. (1995). Probabilistic linkage of large public health data files. *Statistics in Medicine*, *14*, 491C498.

Jin, L., Li, C., & Mehrotra, S. (2003). Efficient record linkage in large data sets.

Kamvar, S., Klein, D., & Manning, C. (2002). Interpreting and extending classical agglomerative clustering algorithms using a model-based approach.

Kehler, A. (2002). *Coherence, reference, and the theory of grammar*. CSLI Publications.

Khardon, R., Roth, D., & Valiant, L. G. (1999). Relational learning for NLP using linear threshold elements. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 911–917).

Kobayashi, M., & Takeda, K. (2000). Information retrieval on the web. *ACM Computing Surveys*, *32*(2), 144–173.

Koudas, N., Marathe, A., & Srivastava, D. (2004). Flexible string matching against large databases in practice.

Kupiec, J. (1992). Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, *6*, 225–242.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lafferty, J., & Zhai, C. (2001, Sept). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'2001* (pp. 111–119).

Lafferty, J., & Zhai, C. (2002). Probabilistic relevance models based on document and query generation.

Lee, L. (1997). *Similarity-based approaches to natural language processing*. Doctoral dissertation, Harvard University, Cambridge, MA.

Lee, L. (1999). Measure of distributional similarity.

Li, X., Morie, P., & Roth, D. (2004a). Identification and tracing of ambiguous names: Discriminative and generative approaches. In *Proceedings of The National Conference on Artificial Intelligence* (pp. 419–424).

Li, X., Morie, P., & Roth, D. (2004b). Robust reading: Identification and tracing of ambiguous names. In *Proceedings of Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting* (pp. 17–24).

Li, X., & Roth, D. (2001). Exploring evidence for shallow parsing.

Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of International Conference on Computational Linguistics* (pp. 556–562).

Light, M., Mann, G., Riloff, E., & Breck, E. (2001). Analyses for Elucidating Current Question Answering Technology. *Journal for Natural Language Engineering*. forthcoming.

Littlestone, N. (1989, March). *Mistake bounds and logarithmic linear-threshold learning algorithms*. Doctoral dissertation, U. C. Santa Cruz.

Mann, G., & Yarowsky, D. (2003). Unsupervised personal name disambiguation.

McCallum, A., Nigam, K., & Ungar, L. (2000). Efficient clustering of high-dimensional data sets with application to reference matching.

McCallum, A., & Wellner, B. (2003). Toward conditional models of identity uncertainty with application to proper noun coreference.

McCarthy, J., & Lehnert, W. (1995). Using decision trees for coreference resolution. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 1050–1055).

McLachlan, G., & Krishnan, T. (1997). *The EM algorithm and extensions*. Wiley.

Mitchell, T. M. (1997). *Machine learning*. New York, NY: McGraw-Hill.

Mochihashi, D., Kikui, G., & Kita, K. (2004). Learning nonstructural distance metric by minimum cluster distortions.

Moldovan, D., Harabagiu, S., Girju, R., Morarescu, P., Lacatusu, F., Novischi, A., Badulescu, A., & Bolohan, O. (2002). Lcc tools for question answering. In Voorhees, E. (Ed.), *the 11th Text Retrieval Conference, NIST* (pp. 144–154).

Moldovan, D., Pasca, M., Harabagiu, S., & Surdeanu, M. (2002). Performance issues and error analysis in an open-domain question answering system. In *the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 33–40).

Monge, A., & Elkan, C. (1996a). The field-matching problem: algorithm and applications.

Monge, A., & Elkan, C. (1996b). The field matching problem: Algorithms and applications.

MUC-6 (1995). *Proceedings of the sixth message understanding conference (muc-6)*. Morgan Kaufmann, San Francisco, CA.

MUC-7 (1999). *the seventh message understanding conference (muc-7)*. Morgan Kaufmann, San Francisco, CA.

Munoz, M., Punyakanok, V., Roth, D., & Zimak, D. (1999, June). A learning approach to shallow parsing.

Ng, V., & Cardie, C. (2003). Improving machine learning approaches to coreference resolution.

Pantel, P., & Lin, D. (2002). Discovering word senses from text.

Parag, & Domingos, P. (2004). Multi-relational record linkage.

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2002). Identity uncertainty and citation matching.

Pasula, H., Marthi, B., Milch, B., Russell, S., & Shpitser, I. (2003). Identity uncertainty and citation matching.

Punyakanok, V., & Roth, D. (2001). The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems* (pp. 995–1001). MIT Press.

Rahm, E., & Bernstein, P. (2001). On matching schemas automatically. *VLDB Journal*, *10*(4).

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

Raman, V., & Hellerstein, J. (2001). Potter's wheel: An interactive data cleaning system. In *The VLDB Journal* (pp. 381–390).

Ravikumar, P., & Cohen, W. (2004). A hierarchical graphical model for record linkage.

Ristad, E. S., & Yianilos, P. N. (1998). Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(5).

Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of The National Conference on Artificial Intelligence* (pp. 806–813).

Roth, D. (1999). Learning in natural language. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 898–904).

Roth, D., Cumby, C., Li, X., Morie, P., Nagarajan, R., Rizzolo, N., Small, K., & Yih, W. (2002). Question answering via enhanced understanding of questions. In *Proceedings of Text REtrieval Conference* (pp. 592–601).

Roth, D., Yang, M.-H., & Ahuja, N. (2000). Learning to recognize objects. In *CVPR'00, The IEEE Conference on Computer Vision and Pattern Recognition* (pp. 724–731). Acceptance Rate: 200/466 (43%).

Roth, D., Yang, M.-H., & Ahuja, N. (2002). Learning to recognize objects. *Neural Computation, 14*(5), 1071–1104.

Roth, D., & Yih, W. (2001). Relational learning via propositional algorithms: An information extraction case study. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 1257–1263).

Salton, G. (1988). Syntactic approaches to automatic book indexing. In *Proceedings of the Annual Meeting of The Association for Computational Linguistics* (pp. 204–210).

Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval*. McGraw-Hill.

Sang, E., & Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition.

Sarawagi, S., & Bhamidipaty, A. (2002). Interactive deduplication using active learning.

Sarawagi, S., & Cohen, W. W. (2004). Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods.

Schultz, M., & Joachims, T. (2004). Learning a distance metric from relative comparisons.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8).

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

Tejada, S., Knoblock, C., & Minton, S. (2002). Learning domain-independent string transformation weights for high accuracy object identification.

Tsuda, K., Akaho, S., & Asai, K. (2003). The EM algorithm for kernel matrix completion with auxiliary data. *Journal of Machine Learning Research, 4*(May).

van Rijsbergen, C. J. (1979). *Information retrieval*. Butterworths.

Vilalta, R., & Rish, I. (2003). A decomposition of classes via clustering to explain and improve naive bayes.

Voorhees, E. (2002). Overview of the TREC-2002 question answering track. In *Proceedings of Text REtrieval Conference* (pp. 115–123).

Weeds, J., Weir, D., & McCarthy, D. (2004). Characterising measures of lexical distributional similarity.

Weis, M., & Naumann, F. (2005). Dogmatix tracks down duplicates in xml.

Wellner, B., McCallum, A., Peng, F., & Hay, M. (2004). An integrated, conditional model of information extraction and coreference with application to citation matching.

Winkler, W. E. (1999). The state of record linkage and current research problems. *Statistics of Income Division, Internal Revenue Service Publication R99/04*.

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2002). Distance metric learning, with application to clustering with side-information.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval, 1*(1/2), 69–90.

Yih, W., & Roth, D. (2002). Probabilistic reasoning for entity and relation recognition.

Zhang, T., & Johnson, D. (2003). A robust risk minimization based named entity recognition system. In Daelemans, W., & Osborne, M. (Eds.), *CoNLL-2003* (pp. 204–207). Edmonton, Canada.

Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval, 4*, 5–31.

Publishing Notes

Most of the content in this dissertation is based on the following papers I have published during Ph.D. study.

Journal Papers

1. Xin Li, Dan Roth. *Learning Question Classifiers: The Role of Semantic Information.* To appear in Journal of Natural Language Engineering, volume 11(4), Dec. 2005.

2. Xin Li, Paul Morie, Dan Roth. *Semantic Integration in Text: From Ambiguous Names to Identifiable Entities.* In AI Magazine, Vol. 26(1), 2005. Pages 45-58. Invited paper.

Conference Papers

3. Xin Li, Dan Roth. *Discriminative Training of Clustering Functions: Theory and Experiments with Entity Identification.* In Proceedings the 9th Conference on Computational Natural Language Learning (CoNLL 2005, acceptance rate: 27%).

4. Warren Shen, Xin Li and Anhai Doan. *Constraint-Based Entity Matching.* In Proceedings of the 21th National Conference on Artificial Intelligence (AAAI 2005, acceptance rate: 18%).

5. Xin Li, Paul Morie, Dan Roth. *Toward Robust Reading: Discriminative and Generative Approaches.* In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2004, acceptance rate: 27%).

6. Xin Li, Paul Morie, Dan Roth. *Robust Reading: Identification and Tracing of Ambiguous Names.* In Proceedings of the 3rd Annual Conference of North American Chapter of Association

of Computational Linguistics (HLT-NAACL 2004, acceptance rate: 26%).

7. Xin Li, Dan Roth, Kevin Small. *The Role of Semantic Information in Learning Question Classifiers.* In Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP 2004, acceptance rate: 31%).

8. Xin Li, Dan Roth, Yuancheng Tu. *PhraseNet: Towads Context Sensitive Lexical Semantics.* In Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL 2003).

9. Xin Li, Dan Roth. *Learning Question Classifiers.* In Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002, acceptance rate: 39%).

10. Xin Li, Dan Roth. *Exploring Evidence for Shallow Parsing.* In Proceedings of the 5th Conference on Computational Natural Language Learning (CoNLL 2001, acceptance rate: 30%).

Workshop and Technical Reports

11. Xin Li, Dan Roth. *Discriminative Training of Clustering Functions: Theory and Experiments with Entity Identification.* The Second Midwest Computational Linguistics Colloquium (MCLC 2005).

12. Xin Li, Dan Roth. *Supervised Discriminative Clustering.* In "Learning with Structured Outputs" Workshop, Eighteenth Annual Conference on Neural Information Processing Systems (NIPS 2004).

13. Xin Li, Dan Roth. *A Unified Framework to Integrate Supervision and Metric Learning into Clustering.* UIUC Technical Report UIUCDCS-R-2004-2488, Dec. 2004.

14. Xin Li, Paul Morie, Dan Roth. *Robust Reading of Ambiguous Writing.* UIUC Technical Report UIUCDCS-R-2003-2371, Dec. 2003.

15. Dan Roth, Chad Cumby, Xin Li, Paul Morie, Ramya Nagarajan, Nick Rizzolo, Kevin Small, Wen-tau Yih. *Question Answering via Enhanced Understanding of Questions.* In Proceedings of the 11th Text Retrieval Conference (TREC 2002).

16. D. Roth, G.K. Kao, Xin Li, R. Nagarajan, V. Punyakanok, N. Rizzolo, W-t Yih, C. Oves-

dotter Alm, L. Gerard Moran. *Learning Components for A Question-Answering System.* In Proceedings of the 10th Text Retrieval Conference (TREC 2001).

Vita

Xin Li was born in Wuhan, China – where the grand Yangtsze River runs across, on May 26, 1975. He grew up by the river and stayed in this city, until he got his Bachelor degree in Computer Science from Wuhan University in 1997. After that, he moved to Beijing, China, to pursue his master study in Software Engineering in Peking University. Again in 2000, he was relocated to Champaign, Illinois and started his doctoral research on artificial intelligence and natural language processing. Following five years of hard working in this area, he will receive his Ph.D and continue his research work in Yahoo! Search.