Foundations and Trends$^{®}$ in
Databases
Vol. 1, No. 3 (2007) 261–377
© 2008 S. Sarawagi
DOI: 10.1561/1500000003

Information Extraction

Sunita Sarawagi

Indian Institute of Technology, CSE, Mumbai 400076, India,
sunita@iitb.ac.in

Abstract

The automatic extraction of information from unstructured sources has opened up new avenues for querying, organizing, and analyzing data by drawing upon the clean semantics of structured databases and the abundance of unstructured data. The field of information extraction has its genesis in the natural language processing community where the primary impetus came from competitions centered around the recognition of named entities like people names and organization from news articles. As society became more data oriented with easy online access to both structured and unstructured data, new applications of structure extraction came around. Now, there is interest in converting our personal desktops to structured databases, the knowledge in scientific publications to structured records, and harnessing the Internet for structured fact finding queries. Consequently, there are many different communities of researchers bringing in techniques from machine learning, databases, information retrieval, and computational linguistics for various aspects of the information extraction problem.

This review is a survey of information extraction research of over two decades from these diverse communities. We create a taxonomy of the field along various dimensions derived from the nature of the extraction task, the techniques used for extraction, the variety of input resources exploited, and the type of output produced. We elaborate on rule-based and statistical methods for entity and relationship extraction. In each case we highlight the different kinds of models for capturing the diversity of clues driving the recognition process and the algorithms for training and efficiently deploying the models. We survey techniques for optimizing the various steps in an information extraction pipeline, adapting to dynamic data, integrating with existing entities and handling uncertainty in the extraction process.

1 Introduction

Information Extraction refers to the automatic extraction of structured information such as entities, relationships between entities, and attributes describing entities from unstructured sources. This enables much richer forms of queries on the abundant unstructured sources than possible with keyword searches alone. When structured and unstructured data co-exist, information extraction makes it possible to integrate the two types of sources and pose queries spanning them.

The extraction of structure from noisy, unstructured sources is a challenging task, that has engaged a veritable community of researchers for over two decades now. With roots in the Natural Language Processing (NLP) community, the topic of structure extraction now engages many different communities spanning machine learning, information retrieval, database, web, and document analysis. Early extraction tasks were concentrated around the identification of named entities, like people and company names and relationship among them from natural language text. The scope of this research was strongly influenced by two competitions, the Message Understanding Conference (MUC) [57, 100, 198] and Automatic Content Extraction (ACE) [1, 159] program. The advent of the Internet considerably increased the extent and diversity of applications depending on various forms of information extraction. Applications such as comparison shopping, and other automatic portal creation applications, lead to a frenzy of research and commercial activity on the topic. As society became more data oriented with easy online access to both structured and unstructured data, new applications of structure extraction came around.

To address the needs of these diverse applications, the techniques of structure extraction have evolved considerably over the last two decades. Early systems were rule-based with manually coded rules [10, 127, 181]. As manual coding of rules became tedious, algorithms for automatically learning rules from examples were developed [7, 43, 60, 195]. As extraction systems were targeted on more noisy unstructured sources, rules were found to be too brittle. Then came the age of statistical learning, where in parallel two kinds of techniques were deployed: generative models based on Hidden Markov Models [3, 20, 25, 189] and conditional models based on maximum entropy [26, 118, 135, 143, 177]. Both were superseded by global conditional models, popularly called Conditional Random Fields [125]. As the scope of extraction systems widened to require a more holistic analysis of a document's structure, techniques from grammar construction [191, 213] were developed. In spite of this journey of varied techniques, there is no clear winner. Rule-based methods [72, 113, 141, 190] and statistical methods [32, 72, 146, 220] continue to be used in parallel depending on the nature of the extraction task. There also exist hybrid models [42, 59, 70, 89, 140, 173] that attempt to reap the benefits of both statistical and rule-based methods.

1.1 Applications

Structure extraction is useful in a diverse set of applications. We list a representative subset of these, categorized along whether the applications are enterprise, personal, scientific, or Web-oriented.

1.1.1 Enterprise Applications

News Tracking: A classical application of information extraction, which has spurred a lot of the early research in the NLP community, is automatically tracking specific event types from news sources.

The popular MUC [57, 100, 198] and ACE [1] competitions are based on the extraction of structured entities like people and company names, and relations such as "is-CEO-of" between them. Other popular tasks are: tracking disease outbreaks [99], and terrorist events from news sources. Consequently there are several research publications [71, 98, 209] and many research prototypes [10, 73, 99, 181] that target extraction of named entities and their relationship from news articles. Two recent applications of information extraction on news articles are: the automatic creation of multimedia news by integrating video and pictures of entities and events annotated in the news articles,[1] and hyperlinking news articles to background information on people, locations, and companies.[2]

Customer Care: Any customer-oriented enterprise collects many forms of unstructured data from customer interaction; for effective management these have to be closely integrated with the enterprise's own structured databases and business ontologies. This has given rise to many interesting extraction problems such as the identification of product names and product attributes from customer emails, linking of customer emails to a specific transaction in a sales database [19, 44], the extraction of merchant name and addresses from sales invoices [226], the extraction of repair records from insurance claim forms [168], the extraction of customer moods from phone conversation transcripts [112], and the extraction of product attribute value pairs from textual product descriptions [97].

Data Cleaning: An essential step in all data warehouse cleaning processes is converting addresses that are stored as flat strings into their structured forms such as road name, city, and state. Large customer-oriented organizations like banks, telephone companies, and universities store millions of addresses. In the original form, these addresses have little explicit structure. Often for the same person, there are different address records stored in different databases. During warehouse construction, it is necessary to put all these addresses in a standard canonical format where all the different fields are identified and duplicates

[1] http://spotlight.reuters.com/.
[2] http://www.linkedfacts.com.

removed. An address record broken into its structured fields not only enables better querying, it also provides a more robust way of doing deduplication and householding — a process that identifies all addresses belonging to the same household [3, 8, 25, 187].

Classified Ads: Classified ads and other listings such as restaurant lists is another domain with implicit structure that when exposed can be invaluable for querying. Many researchers have specifically targeted such record-oriented data in their extraction research [150, 156, 157, 195].

1.1.2 Personal Information Management

Personal information management (PIM) systems seek to organize personal data like documents, emails, projects and people in a structured inter-linked format [41, 46, 74]. The success of such systems will depend on being able to automatically extract structure from existing predominantly file-based unstructured sources. Thus, for example we should be able to automatically extract from a PowerPoint file, the author of a talk and link the person to the presenter of a talk announced in an email. Emails, in particular, have served as testbeds for many extraction tasks such as locating mentions of people names and phone numbers [113, 152], and inferring request types in service centers [63].

1.1.3 Scientific Applications

The recent rise of the field of bio-informatics has broadened the scope of earlier extractions from named entities, to biological objects such as proteins and genes. A central problem is extracting from paper repositories such as Pubmed, protein names, and their interaction [22, 32, 166]. Since the form of entities like Gene and Protein names is very different from classical named entities like people and companies, this task has helped to broaden the techniques used for extraction.

1.1.4 Web Oriented Applications

Citation Databases: Many citation databases on the web have been created through elaborate structure extraction steps from sources ranging from conference web sites to individual home pages. Popular amongst these are Citeseer [126], Google Scholar[3] and Cora [144]. The creation of such databases requires structure extraction at many different levels starting from navigating web sites for locating pages containing publication records, extracting individual publication records from a HTML page, extracting title, authors, and references from paper PDFs, and segmenting citation strings into individual authors, title, venue, and year fields. The resulting structured database provides significant value added in terms of allowing forward references, and aggregate statistics such as author-level citation counts.

Opinion Databases: There are innumerable web sites storing unmoderated opinions about a range of topics, including products, books, movies, people, and music. Many of the opinions are in free text form hidden behind Blogs, newsgroup posts, review sites, and so on. The value of these reviews can be greatly enhanced if organized along structured fields. For example, for products it might be useful to find out for each feature of the product, the prevalent polarity of opinion [131, 167]. See [160] for a recent survey.

Community Websites: Another example of the creation of structured databases from web documents is community web sites such as DBLife [78] and Rexa[4] that tracks information about researchers, conferences, talks, projects, and events relevant to a specific community. The creation of such structured databases requires many extraction steps: locating talk announcements from department pages, extracting names of speakers and titles from them [189], extracting structured records about a conference from a website [111], and so on.

Comparison Shopping: There is much interest in creating comparison shopping web sites that automatically crawl merchant web sites to find products and their prices which can then be used for comparison shopping [87]. As web technologies evolved, most large merchant web sites started getting hidden behind forms and scripting languages. Consequently, the focus has shifted to crawling and extracting information

[3] http://www.scholar.google.com.
[4] http://rexa.info.

from form-based web sites [104]. The extraction of information from form-based web sites is an active research area not covered in this survey.

Ad Placement on Webpages: Suppose a web site wants to place advertisements of a product next to the text that both mentions the product and expresses a positive opinion about it. Both of these subtasks: extracting mentions of products and the type of opinion expressed on the product are examples of information extraction tasks that can facilitate the burgeoning Internet ad placement industry [29].

Structured Web Searches: Finally, a grand challenge problem for information extraction is allowing structured search queries involving entities and their relationships on the World Wide Web. Keyword searches are adequate for getting information about entities, which are typically nouns or noun phrases. They fail on queries that are looking for relationships between entities [45]. For example, if one wants to retrieve documents containing text of the form "Company X acquired Company Y", then keywords alone are extremely inadequate. The only obvious keyword is "acquired", and one has to work hard to introduce related words like "Corp" etc. to get the required documents. Research prototypes for answering such kinds of queries are only starting to appear [39, 196, 197].

1.2 Organization of the Survey

Given the broad scope of the topic, the diversity of communities involved and the long history, compiling an exhaustive survey on structure extraction is a daunting task. Fortunately, there are many short surveys on information extraction from different communities that can be used to supplement what is missed here [71, 98, 104, 139, 142, 153, 154, 178, 209, 212].

We provide a taxonomy of the field by categorizing along different dimensions and alongside scope out what is covered in this survey. We layout the field of information extraction along the following five dimensions.

(1) The type of structure extracted (entities, relationships, lists, tables, attributes, etc.).
(2) The type of unstructured source (short strings or documents, templatized or open-ended).
(3) The type of input resources available for extraction (structured databases, labeled unstructured data, linguistic tags, etc.).
(4) The method used for extraction (rule-based or statistical, manually coded or trained from examples).
(5) The output of extraction (annotated unstructured text, or a database).

These are discussed in Sections 1.3 through 1.7.

1.3 Types of Structure Extracted

We categorize the type of structure extracted from an unstructured source into four types: entities, relationships between entities, adjectives describing entities, and higher-order structures such as tables and lists.

1.3.1 Entities

Entities are typically noun phrases and comprise of one to a few tokens in the unstructured text. The most popular form of entities is *named entities* like names of persons, locations, and companies as popularized in the MUC [57, 100], ACE [1, 159], and CoNLL [206] competitions. Named entity recognition was first introduced in the sixth MUC [100] and consisted of three subtasks: proper names and acronyms of persons, locations, and organizations (ENAMEX), absolute temporal terms (TIMEX) and monetary and other numeric expressions (NUMEX). Now the term entities is expanded to also include generics like disease names, protein names, paper titles, and journal names. The ACE competition for entity relationship extraction from natural language text lists more than 100 different entity types.

Figures 1.1 and 1.2 present examples of entity extractions: Figure 1.1 shows the classical IE task of extracting person, organization,

According to Robert Callahan, president of Eastern's flight attendants union, the past practice of Eastern's parent, Houston-based Texas Air Corp., has involved ultimatums to unions to accept the carrier's terms
The unstructured source
According to ⟨Per⟩ **Robert Callahan** ⟨**/Per**⟩, president of ⟨Org⟩ **Eastern's** ⟨**/Org**⟩ igh t attendants union, the past practice of ⟨Org⟩ **Eastern's** ⟨**/Org**⟩ parent, ⟨Loc⟩ **Houston** ⟨**/Loc**⟩-based ⟨Org⟩ **Texas Air Corp.** ⟨**/Org**⟩, has involved ultimatums to unions to accept the carrier's terms
Annotated entities
Robert Callahan ⟨Employee_Of⟩ Eastern's Texas Air Corp. ⟨Located_In⟩ Houston
Relationships

Fig. 1.1 Traditionally named entity and relationship extraction from plain text (in this case a news article). The extracted entities are bold-faced with the entity type surrounding it.

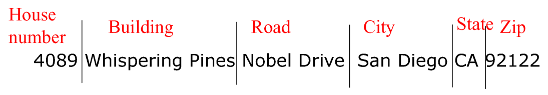

Fig. 1.2 Text segmentation as an example of entity extraction from address records.

and location entities from news articles; Figure 1.2 shows an example where entity extraction can be treated as a problem of segmenting a text record into structured entities. In this case an address string is segmented so as to identify six structured entities. More examples of segmentation of addresses coming from diverse geographical locations appear in Table 1.1.

We cover techniques for entity extraction in Sections 2 and 3.

1.3.2 Relationships

Relationships are defined over two or more entities related in a predefined way. Examples are "is employee of" relationship between a person and an organization, "is acquired by" relationship between pairs of companies, "location of outbreak" relationship between a disease

Table 1.1 Sample addresses from different countries. The first line shows the unformatted address and the second line shows the address broken into its elements.

#	Address text [Segmented address]
0	M. J. Muller, 71, route de Longwy L-4750 PETANGE [recipient: M. J. Muller] [House#: 71] [Street: route de Longwy] [Zip: L-4750] [city:PETANGE]
1	Viale Europa, 22 00144-ROMA RM [Street: Viale Europa] [House#: 22] [City: ROMA] [Province: RM] [Zip: 00144-]
2	7D-Brijdham Bangur Nagar Goregaon (W) Bombay 400 090 [House#: 7D-] [Building: Brijdham] [Colony: Bangur Nagar] [Area: Goregaon (W)] [City: Bombay] [Zip: 400 090]
3	18100 New Hamshire Ave. Silver Spring, MD 20861 [House#: 18100], [Street: New Hamshire Ave.], [City: Silver Spring,], [State: MD], [Zip: 20861]

and a location, and "is price of" relationship between a product name and a currency amount on a web-page. Figure 1.1 shows instances of the extraction of two relationships from a news article. The extraction of relationships differs from the extraction of entities in one significant way. Whereas entities refer to a sequence of words in the source and can be expressed as annotations on the source, relationships are not annotations on a subset of words. Instead they express the associations between two separate text snippets representing the entities.

The extraction of multi-way relationships is often referred to as record extraction. A popular subtype of record extraction is event extraction. For example, for an event such as a disease outbreak we extract a multi-way relationship involving the "disease name", "location of the outbreak", "number of people affected", "number of people killed", and "date of outbreak." Some record extraction tasks are trivial because the unstructured string implies a fixed set of relationships. For example, for addresses, the relation "is located in" is implied between an extracted street name and city name.

In Section 4, we cover techniques for relationship extraction concentrating mostly on binary relationships.

Another form of multi-way relationship popular in the natural language community is Semantic Role Labeling [124], where given a predicate in a sentence, the goal is to identify various semantic arguments of the predicate. For example, given a predicate *accept* in the sentence “He accepted the manuscript from his dying father with trembling hands” the extraction task is to find the role-sets of the predicate consisting of the “acceptor”, “thing accepted”, and “accepted-from”. We will not cover semantic role labeling in this survey, and refer the reader to [124] to know more about this topic.

1.3.3 Adjectives Describing Entities

In many applications we need to associate a given entity with the value of an adjective describing the entity. The value of this adjective typically needs to be derived by combining soft clues spread over many different words around the entity. For example, given an entity type, say restaurants, or music bands, we need to extract parts of a Blog or web-page that presents a critique of entities of such type. Then, we would like to infer if the critique is positive or negative. This is also called opinion extraction and is now a topic of active research interest in many different communities. We will not cover this topic in this survey but instead refer the reader to [160] for a current and exhaustive survey.

1.3.4 Structures such as Lists, Tables, and Ontologies

The scope of extraction systems has now expanded to include the extraction of not such atomic entities and flat records but also richer structures such as tables, lists, and trees from various types of documents. For example, [109, 134, 164] addresses the identification of tables from documents, [62, 85, 156] considers the extraction of elements of a list, and [130] considers the extraction of ontologies. We will not be able to cover this topic in the survey to contain its scope and volume. On the topic of table extraction there is an extensive research literature spanning many different communities, including the document analysis [84, 109, 134, 222], information retrieval [164], web [62, 96], database [36, 165], and machine learning [164, 216] communities. A survey can be found in [84].

1.4 Types of Unstructured Sources

We classify the type of unstructured source along two dimensions: the basic unit of granularity on which an extractor is run, and the heterogeneity in style and format across unstructured documents.

1.4.1 Granularity of Extraction

Record or Sentences: The most popular form of extraction is from small text snippets that are either unstructured records like addresses, citations and classified ads [3, 25, 151, 163, 195] or sentences extracted from a natural language paragraph [1, 26, 57, 100, 159, 206]. In the case of unstructured records, the data can be treated as a set of structured fields concatenated together, possibly with a limited reordering of the fields. Thus, each word is a part of such structured field and during extraction we just need to segment the text at the entity boundaries. In contrast, in sentences there are many words that do not form part of any entity of interest.

Paragraphs and Documents: Many other extraction tasks make it necessary to consider the context of multiple sentences or an entire document for meaningful extractions. Popular examples include extractions of events from news articles [57, 100], extraction of part number and problem description from emails in help centers, extraction of a structured resume from a word file, extraction of title, location and timing of a talk from talk announcements [189] and the extraction of paper headers and citations from a scientific publication [163].

The techniques proposed in this survey mostly assume the first kind of source. Typically, for extracting information from longer units the main challenge is designing efficient techniques for filtering only the relevant portion of a long document. Currently, this is handled through hand-coded heuristics, so there is nothing specifically to cover in a survey on the handling of longer units.

1.4.2 Heterogeneity of Unstructured Sources

An important concern that has a huge impact on the complexity and accuracy of an extractor is how much homogeneity is there in the format and style of the unstructured documents. We categorize them as:

Machine Generated Pages: On the easy end of the spectrum we have highly templatized machine generated pages. A popular source in this space is HTML documents dynamically generated via database backed sites. The extractors for such documents are popularly known as wrappers. These have been extensively studied in many communities [11, 184, 16, 17, 67, 103, 106, 123, 133, 149, 156], where the main challenge is how to automatically figure out the layout of a page with little or no human input by exploiting mostly the regularity of HTML tags present in the page. In this survey we will not be able to do justice to the extensive literature on web wrapper development.

Partially Structured Domain Specific Sources: The most studied setting for information extraction is where the input source is from within a well-defined scope, say news articles [1, 57, 100, 159, 206], or classified ads [151, 195], or citations [25, 163], or resumes. In all these examples, there is an informal style that is roughly followed so that it is possible to develop a decent extraction model given enough labeled data, but there is lot more variety from one input to another than in machine generated pages. Most of the techniques in this survey are for such input sources.

Open Ended Sources: Recently [14, 37, 86, 192], there is interest in extracting instances of relationships and entities from open domains such as the web where there is little that can be expected in terms of homogeneity or consistency. In such situations, one important factor is to exploit the redundancy of the extracted information across many different sources. We discuss extractions from such sources in the context of relationship extraction in Section 4.2.

1.5 Input Resources for Extraction

The basic specification of an extraction task includes just the types of structures to be extracted and the unstructured sources from which it should be extracted. In practice, there are several additional input resources that are available to aid the extraction.

1.5.1 Structured Databases

Existing structured databases of known entities and relationships are a valuable resource to improve extraction accuracy. Typically, there are several such databases available during extraction. In many applications unstructured data needs to be integrated with structured databases on an ongoing basis so that at the time of extraction a large database is available. Consider the example of portals like DBLife, Citeseer, and Google Scholar. In addition to their own operational database of extracted publications, they can also exploit external databases such as the ACM digital library or DBLP. Other examples include the use of a sales transactions database and product database for extracting fields like customer id and product name in a customer email; the use of a contact database to extract authoring information from files in a personal information management system; the use of a postal database to identify entities in address records.

1.5.2 Labeled Unstructured Text

Many extraction systems are seeded via labeled unstructured text. The collection of labeled unstructured text requires tedious labeling effort. However, this effort is not totally avoidable because even when an extraction system is manually coded, a ground truth is necessary for evaluating its accuracy. A labeled unstructured source is significantly more valuable than a structured database because it provides contextual information about an entity and also because the form in which an entity appears in the unstructured data is often a very noisy form of its occurrence in the database.

We will discuss how labeled data is used for learning entity extraction models in Sections 2.3 and 3.4 and for relationship extraction in Section 4.1. In Section 4.2, we show how to learn a model using only a structured database and a large corpus of unlabeled corpus. We discuss how structured databases are used in conjunction with labeled data in Sections 2 and 3.

1.5.3 Preprocessing Libraries for Unstructured Text

Many extraction systems crucially depend on preprocessing libraries that enrich it with linguistic or layout information that serve as valuable anchors for structure recognition.

Natural Language Text: Natural language documents are often analyzed by a deep pipeline of preprocessing libraries, including,

- *Sentence analyzer and tokenizer* that identifies the boundaries of sentences in a document and decomposes each sentence into tokens. Tokens are obtained by splitting a sentence along a predefined set of delimiters like spaces, commas, and dots. A token is typically a word or a digit, or a punctuation.
- *Part of speech tagger* that assigns to each word a grammatical category coming from a fixed set. The set of tags includes the conventional part of speech such as noun, verb, adjective, adverb, article, conjunct, and pronoun; but is often considerably more detailed to capture many subtypes of the basic types. Examples of well-known tag sets are the Brown tag set which has 179 total tags, and the Penn treebank tag set that has 45 tags [137]. An example of POS tags attached to a sentence appears below:

  The/DT University/NNP of/IN Helsinki/NNP hosts/VBZ ICML/NNP this/DT year/NN

- *Parser* that groups words in a sentence into prominent phrase types such as noun phrases, prepositional phrases, and verb phrases. A context free grammar is typically used to identify the structure of a sentence in terms of its constituent phrase types. The output of parsing is a parse tree that groups words into syntactic phrases. An example of a parse tree appears in Figure 4.1. Parse trees are useful in entity extraction because typically named entities are noun phrases. In relationship extraction they are useful because they provide valuable linkages between verbs and their arguments as we will see in Section 4.1.

- *Dependency analyzer* that identifies the words in a sentence that form arguments of other words in the sentence. For example, in the sentence “Apple is located in Cupertino”, the word “Apple” and “Cupertino” are dependent on the word “located”. In particular, they respectively form the subject and object argument of the word “located”. The output of a dependency analyzer is a graph where the nodes are the words and the directed edges are used to connect a word to words that depend on it. An example of a dependency graph appears in Figure 4.2. The edges could be typed to indicate the type of dependency, but even untyped edges are useful for relationship extraction as we will see in Section 4.

Many of the above preprocessing steps are expensive. The shift is now for selective preprocessing of only parts of the text. Many shallow extractions are possible without subjecting a sentence to the full preprocessing pipeline. Also, some of these preprocessing steps, example parsing, are often erroneous. The extraction system needs to be robust to errors in the preprocessing steps to avoid cascading of errors. This problem is particularly severe on ill-formed sentences of the kind found in emails and speech transcripts.

GATE [72] and UIMA [91] are two examples of frameworks that provide support for such preprocessing pipelines. Many NLP libraries are also freely available for download such as IBM’s Languageware,[5] libraries from the Stanford NLP group,[6] and several others listed under the OpenNLP effort.[7]

Formatted Text: For formatted text such as a pdf document and a web-page, there is often a need for understanding the overall structure and layout of the source before entity extraction. Two popular preprocessing steps on formatted documents are, extracting items in a list-like environment and creating hierarchies of rectangular regions comprising logical units of content. Much work exists in this area in the document

[5] http://www.alphaworks.ibm.com/tech/lrw.
[6] http://nlp.stanford.edu/software/.
[7] http://opennlp.sourceforge.net/.

analysis community [139] and elsewhere [40, 85, 157, 191]. We will not discuss these in this survey.

1.6 Methods of Extraction

We categorize the method used for information extraction along two dimensions: hand-coded or learning-based and rule-based or statistical.

1.6.1 Hand-coded or Learning-based

A hand-coded system requires human experts to define rules or regular expressions or program snippets for performing the extraction. That person needs to be a domain expert and a programmer, and possess descent linguistic understanding to be able to develop robust extraction rules. In contrast, learning-based systems require manually labeled unstructured examples to train machine learning models of extraction. Even in the learning-based systems, domain expertise is needed in identifying and labeling examples that will be representative of the actual deployment setting. It is also necessary to possess an understanding of machine learning to be able to choose between various model alternatives and also to define features that will be robust on unseen data. The nature of the extraction task and the amount of noise in the unstructured data should be used to decide between a hand-coded and a learning-based system. An interesting commentary that quantitatively and qualitatively compares the two sides can be found in [127].

1.6.2 Rule-based or Statistical

Rule-based extraction methods are driven by hard predicates, whereas statistical methods make decisions based on a weighted sum of predicate firings. Rule-based methods are easier to interpret and develop, whereas statistical methods are more robust to noise in the unstructured data. Therefore, rule-based systems are more useful in closed domains where human involvement is both essential and available. In open-ended domains like fact extraction from speech transcripts, or opinion extraction from Blogs, the soft logic of statistical methods is more appropriate. We will present both rule-based techniques for entity extraction in Section 2 and statistical techniques for entity and relationship extraction in Sections 3 and 4, respectively.

1.7 Output of Extraction Systems

There are two primary modes in which an extraction system is deployed. First, where the goal is to identify all mentions of the structured information in the unstructured text. Second, where the goal is to populate a database of structured entities. In this case, the end user does not care about the unstructured text after the structured entities are extracted from it. The core extraction techniques remain the same irrespective of the form of the output. Therefore, in the rest of the survey we will assume the first form of output. Only for a few types of open ended extractions where redundancy is used to improve the reliability of extractions stored in a database is the distinction important. We briefly cover this scenario in Sections 4.2 and 5.4.3.

1.8 Challenges

Large scale deployments of information extraction models raises many challenges of accuracy, performance, maintainability, and usability that we elaborate on next.

1.8.1 Accuracy

The foremost challenge facing the research community, in spite of more than two decades of research in the field, is designing models that achieve high accuracy of extraction. We list some of the factors that contribute to the difficulty of achieving high accuracy in extraction tasks.

Diversity of Clues: The inherent complexity of the recognition task makes it crucial to combine evidence from a diverse set of clues, each of which could individually be very weak. Even the simplest and the most well-explored of tasks, Named Entity recognition, depends on a myriad set of clues including orthographic property of the words, their part of speech, similarity with an existing database of entities, presence of specific signature words and so on. Optimally combining these different modalities of clues presents a nontrivial modeling challenge. This is evidenced by the huge research literature for this task alone over the past two decades. We will encounter many of these in the next three sections of the survey. However, the problem is far from solved for all the different types of extraction tasks that we mentioned in Section 1.3.

Difficulty of Detecting Missed Extractions: The accuracy of extraction comprises of two components: precision, that measures the percent of extracted entries that are correct, and recall, that measures the percent of actual entities that were extracted correctly. In many cases, precision is high because it is easy to manually detect mistakes in extractions and then tune the models until those mistakes disappear. The bigger challenge is achieving high recall, because without extensive labeled data it is not even possible to detect what was missed in the large mass of unstructured information.

Increased Complexity of the Structures Extracted: New tasks requiring the extraction of increasingly complex kinds of entities keep getting defined. Of the recent additions, it is not entirely clear how to extract longer entities such as the parts within running text of a Blog where a restaurant is mentioned and critiqued. One of the challenges in such tasks is that the boundary of the entity is not clearly defined.

1.8.2 Running Time

Real-life deployment of extraction techniques in the context of an operational system raises many practical performance challenges. These arise at many different levels. First, we need mechanisms to efficiently filter the right subset of documents that are likely to contain the structured information of interest. Second, we need to find means of efficiently zooming into the (typically small) portion of the document that contains the relevant information. Finally, we need to worry about the many expensive processing steps that the selected portion might need to go through. For example, while existing database of structured entries are invaluable for information extraction, they also raise performance challenges. The order in which we search for parts of a compound entity or relationship can have a big influence on running time. These and other performance issues are discussed in Section 5.1.

1.8.3 Other Systems Issues

Dynamically Changing Sources: Extraction models take time and effort to build and tune to specific unstructured sources. When these sources change, a challenge to any system that operates continuously on that source is detecting the change and adapting the model automatically to the change. We elaborate on this topic in Section 5.2.

Data Integration: Although in this survey we will concentrate primarily on information extraction, extraction goes hand in hand with the integration of the extracted information with pre-existing datasets and with information already extracted. Many researchers have also attempted to jointly solve the extraction and integration problem with the hope that it will provide higher accuracy than performing each of these steps directly. We elaborate further in Section 5.3.

Extraction Errors: It is impossible to guarantee perfect extraction accuracy in real-life deployment settings even with the latest extraction tools. The problem is more severe when the sources are extremely heterogeneous, making it impossible to hand tune any extraction tool to perfection. One method of surmounting the problem of extraction errors is to require that each extracted entity be attached with confidence scores that correlate with the probability that the extracted entities are correct. Normally, even this is a hard goal to achieve. Another challenging issue is how to represent such results in a database that captures the imprecision of extraction, while being easy to store and query. In Section 5.4, we review techniques for managing errors that arise in the extraction process.

Section Layout

The rest of the survey is organized as follows. In Section 2, we cover rule-based techniques for entity extraction. In Section 3, we present an overview of statistical methods for entity extraction. In Section 4, we cover statistical and rule-based techniques for relationship extraction. In Section 5, we discuss work on handling various performance and systems issues associated with creating an operational extraction system.

2

Entity Extraction: Rule-based Methods

Many real-life extraction tasks can be conveniently handled through a collection of rules, which are either hand-coded or learnt from examples. Early information extraction systems were all rule-based [10, 72, 141, 181] and they continue to be researched and engineered [60, 113, 154, 190, 209] to meet the challenges of real world extraction systems. Rules are particularly useful when the task is controlled and well-behaved like the extraction of phone numbers and zip codes from emails, or when creating wrappers for machine generated web-pages. Also, rule-based systems are faster and more easily amenable to optimizations [179, 190].

A typical rule-based system consists of two parts: a collection of rules, and a set of policies to control the firings of multiple rules. In Section 2.1, we present the basic form of rules and in Section 2.2, we present rule-consolidation policies. Rules are either manually coded, or learnt from example labeled sources. In Section 2.3, we will present algorithms for learning rules.

2.1 Form and Representation of Rules

Rule-based systems have a long history of usage and many different rule representation formats have evolved over the years. These include the Common Pattern Specification Language (CSPL) [10] and its derivatives like JAPE [72], pattern items and lists as in Rapier [43], regular expressions as in WHISK [195], SQL expressions as in Avatar [113, 179], and Datalog expressions as in DBLife [190]. We describe rules in a generic manner that captures the core functionality of most of these languages.

A basic rule is of the form: "Contextual Pattern $\rightarrow$ Action". A Contextual Pattern consists of one or more labeled patterns capturing various properties of one or more entities and the context in which they appear in the text. A labeled pattern consists of a pattern that is roughly a regular expression defined over features of tokens in the text and an optional label. The features can be just about any property of the token or the context or the document in which the token appears. We list examples of typical features in Section 2.1.1. The optional label is used to refer to the matching tokens in the rule action.

The action part of the rule is used to denote various kinds of tagging actions: assigning an entity label to a sequence of tokens, inserting the start or the end of an entity tag at a position, or assigning multiple entity tags. We elaborate on these in Sections 2.1.2, 2.1.3, and 2.1.4, respectively.

Most rule-based systems are cascaded; rules are applied in multiple phases where each phase associates an input document with an annotation that serves as input features to the next phase. For example, an extractor for contact addresses of people is created out of two phases of rule annotators: the first phase labels tokens with entity labels like people names, geographic locations like road names, city names, and email addresses. The second phase locates address blocks with the output of the first phase as additional features.

2.1.1 Features of Tokens

A token in a sentence is typically associated with a bag of features obtained via one or more of the following criteria:

- The string representing the token.
- Orthography type of the token that can take values of the

form capitalized word, smallcase word, mixed case word, number, special symbol, space, punctuation, and so on.
- The Part of speech of the token.
- The list of dictionaries in which the token appears. Often this can be further refined to indicate if the token matches the start, end, or middle word of a dictionary. For example, a token like "New" that matches the first word of a dictionary of city names will be associated with a feature, "Dictionary-Lookup = start of city."
- Annotations attached by earlier processing steps.

2.1.2 Rules to Identify a Single Entity

Rules for recognizing a single full entity consists of three types of patterns:

- An optional pattern capturing the context before the start of an entity.
- A pattern matching the tokens in the entity.
- An optional pattern for capturing the context after the end of the entity.

An example of a pattern for identifying person names of the form "Dr. Yair Weiss" consisting of a title token as listed in a dictionary of titles (containing entries like: "Prof", "Dr", "Mr"), a dot, and two capitalized words is

({DictionaryLookup = Titles} {String = "."} {Orthography type = capitalized word}{2}) → Person Names.

Each condition within the curly braces is a condition on a token followed with an optional number indicating the repetition count of tokens.

An example of a rule for marking all numbers following words "by" and "in" as the Year entity is

({String="by"|String="in"}) ({Orthography type = Number }):y → Year=:y.

There are two patterns in this rule: the first one for capturing the context of the occurrence of the Year entity and the second one for capturing the properties of tokens forming the "Year" field.

Another example for finding company names of the form "The XYZ Corp." or "ABC Ltd." is given by

> ({String="The"}? {Orthography type = All capitalized} {Orthography type = Capitalized word, DictionaryType = Company end}) $\rightarrow$ Company name.

The first term allows the "The" to be optional, the second term matches all capitalized abbreviations, and the last term matches all capitalized words that form the last word of any entry in a dictionary of company names. In Figure 2.1, we give a subset of the more than dozen rules for identifying company names in GATE, a popular entity recognition system [72].

2.1.3 Rules to Mark Entity Boundaries

For some entity types, in particular long entities like book titles, it is more efficient to define separate rules to mark the start and end of an entity boundary. These are fired independently and all tokens in between two start and end markers are called as the entity. Viewed another way, each rule essentially leads to the insertion of a single SGML tag in the text where the tag can be either a begin tag or an end tag. Separate consolidation policies are designed to handle inconsistencies like two begin entity markers before an end entity marker. An example of a rule to insert a $\langle$journal$\rangle$ tag to mark the start of a journal name in a citation record is

> ({String="to"} {String="appear"} {String="in"}):jstart ({Orthography type = Capitalized word}{2-5}) $\rightarrow$ insert $\langle$journal$\rangle$ after:jstart.

Many successful rule-based extraction systems are based on such rules, including $(\mathrm{LP})^2$ [60], STALKER [156], Rapier [43], and WEIN [121, 123].

Rule: TheGazOrganization
Priority: 50
// Matches "The <in list of company names>"
({Part of speech = DT | Part of speech = RB} {DictionaryLookup = organization})
→ Organization

Rule: LocOrganization
Priority: 50
// Matches "London Police"
({DictionaryLookup = location | DictionaryLookup = country} {DictionaryLookup = organization} {DictionaryLookup = organization}?) → Organization

Rule: INOrgXandY
Priority: 200
// Matches "in Bradford & Bingley", or "in Bradford & Bingley Ltd"
({Token string = "in"})
({Part of speech = NNP}+ {Token string = "&"} {Orthography type = upperInitial}+ {DictionaryLookup = organization end}?):orgName → Organization=:orgName

Rule: OrgDept
Priority: 25
// Matches "Department of Pure Mathematics and Physics"
({Token.string = "Department"} {Token.string = "of"} {Orthography type = upperInitial}+ ({Token.string = "and"} {Orthography type = upperInitial}+)?) → Organization

Fig. 2.1 A subset of rules for identifying company names paraphrased from the Named Entity recognizer in Gate.

2.1.4 Rules for Multiple Entities

Some rules take the form of regular expressions with multiple slots, each representing a different entity so that this rule results in the recognition of multiple entities simultaneously. These rules are more useful for record oriented data. For example, the WHISK [195] rule-based system has been targeted for extraction from structured records such as medical records, equipment maintenance logs, and classified ads. This rule rephrased from [195] extracts two entities, the number of bedrooms and rent, from an apartment rental ad.

({Orthography type = Digit}):Bedrooms ({String ="BR"}) ({}*) ({String ="$"}) ({Orthography type = Number}):Price → Number of Bedrooms = :Bedroom, Rent =: Price

2.1.5 Alternative Forms of Rules

Many state-of-the-art rule-based systems allow arbitrary programs written in procedural languages such as Java and C++ in place of both the pattern and action part of the rule. For example, GATE [72] supports Java programs in place of its custom rule scripting language called JAPE in the action of a rule. This is a powerful capability because it allows the action part of the rule to access the different features that were used in the pattern part of the rule and use those to insert new fields for the annotated string. For example, the action part could lead to the insertion of the standardized form of a string from a dictionary. These new fields could serve as additional features for a later rule in the pipeline. Similarly, in the Prolog-based declarative formulations of [190] any procedural code can be substituted as a pattern matcher for any subset of entity types.

2.2 Organizing Collection of Rules

A typical rule-based system consists of a very large collection of rules, and often for the same action multiple rules are used to cover different kinds of inputs. Each firing of a rule identifies a span of text to be called a particular entity or entity sub-type. It is possible that the spans demarcated by different rules overlap and lead to conflicting actions. Thus, an important component of a rule engine is how to organize the rules and control the order in which they are applied so as to eliminate conflicts, or resolve them when they arise. This component forms one the most nonstandardized and custom-tuned part of a rule-based system, often involving many heuristics and special case handling. We present an overview of the common practices.

2.2.1 Unordered Rules with Custom Policies to Resolve Conflicts

A popular strategy is to treat rules as an unordered collection of disjuncts. Each rule fires independently of the other. A conflict arises when two different overlapping text spans are covered by two different rules.

Special policies are coded to resolve such conflicts. Some examples of such policies are

- Prefer rules that mark larger spans of text as an entity type. For example in GATE [72] one strategy for resolving conflicts is to favor the rule matching a longer span. In case of a tie, a rule with a higher priority is selected.
- Merge the spans of text that overlap. This rule only applies when the action part of the two rules is the same. If not, some other policy is needed to resolve the conflict. This is one of the strategies that a user can opt for in the IE system described in [113, 179].

This laissez faire method of organizing rules is popular because it allows a user more flexibility in defining rules without worrying too much about overlap with existing rules.

2.2.2 Rules Arranged as an Ordered Set

Another popular strategy is to define a complete priority order on all the rules and when a pair of rules conflict, arbitrate in favor of the one with a higher priority [141]. In learning based systems such rule priorities are fixed by some function of the precision and coverage of the rule on the training data. A common practice is to order rules in decreasing order of precision of the rule on the training data.

An advantage of defining a complete order over rules is that a later rule can be defined on the actions of earlier rules. This is particularly useful for fixing the error of unmatched tags in rules where actions correspond to an insertion of either a start or an end tag of an entity type. An example of two such rules, is shown below where the second rule of lower priority inserts the ⟨/journal⟩ on the results of a earlier rule for inserting a ⟨journal⟩ tag.

R1: ({String = “to”} {String = “appear”} {String = “in”}):jstart ({Orthography type = Capitalized word}{2-5}) → insert ⟨journal⟩ after :jstart.
R2: {tag = ⟨journal⟩}({Orthography type = word}+):jend {String = “vol”}→ insert ⟨/journal⟩ after :jend.

$(\mathrm{LP})^2$ is an example of a rule learning algorithm that follows this strategy. $(\mathrm{LP})^2$ first uses high precision rules to independently recognize either the start or the end boundary of an entity and then handles the unmatched cases through rules defined on the inserted boundary and other possibly low confidence features of tokens.

2.2.3 Rule Consolidation via Finite State Machines

Both of the above forms of rules can be equivalently expressed as a deterministic finite state automata. But, the user at the time of defining the rules is shielded from the details of forming the unified automata. Sometimes, the user might want to exercise direct control by explicitly defining the full automata to control the exact sequence of firings of rules. Softmealy [106] is one such approach where each entity is represented as a node in an FST. The nodes are connected via directed edges. Each edge is associated with a rule on the input tokens that must be satisfied for the edge to be taken. Thus, every rule firing has to correspond to a path in the FST and as long as there is a unique path from the start to a sink state for each sequence of tokens, there is no ambiguity about the order of rule firings. However, for increasing recall Softmealy does allow multiple rules to apply at a node. It then depends on a hand-coded set of policy decisions to arbitrate between them.

2.3 Rule Learning Algorithms

We now address the question of how rules are formulated in the first place. A typical entity extraction system depends on a large finely tuned set of rules. Often these rules are manually coded by a domain expert. However in many cases, rules can be learnt automatically from labeled examples of entities in unstructured text. In this section, we discuss algorithms commonly used for inducing rules from labeled examples.

We concentrate on learning an unordered disjunction of rules as in Section 2.2.1. We are given several examples of unstructured documents $D = \mathbf{x}_1,\ldots,\mathbf{x}_N$, where all occurrences of entities in the examples are marked correctly. We call this the training set. Our goal is to learn a set of rules $R_1, \dots, R_k$ such that the action part of each rule is one of three action types described in Sections 2.1.2 through 2.1.4. The body of each rule R will match a fraction $S(R)$ of the data segments in the N training documents. We call this fraction the *coverage* of R. Of all segments R covers, the action specified by R will be correct only for a subset $S'(R)$ of them. The ratio of the sizes of $S'(R)$ and $S(R)$ is the precision of the rule. In rule learning, our goal is to cover all segments that contain an annotation by one or more rules and to ensure that the precision of each rule is high. Ultimately, the set of rules have to provide good recall and precision on new documents. Therefore, a trivial solution that covers each entity in D by its own very specific rule is useless even if this rule set has 100% coverage and precision. To ensure generalizability, rule-learning algorithms attempt to define the smallest set of rules that cover the maximum number of training cases with high precision. However, finding such a size optimal rule set is intractable. So, existing rule-learning algorithms follow a greedy hill climbing strategy for learning one rule at a time under the following general framework.

(1) Rset = set of rules, initially empty.
(2) While there exists an entity $\mathbf{x} \in D$ not covered by any rule in Rset
    (a) Form new rules around $\mathbf{x}$.
    (b) Add new rules to Rset.
(3) Post process rules to prune away redundant rules.

The main challenge in the above framework is in figuring out how to create a new rule that has high overall coverage (and therefore generalizes), is nonredundant given rules already existing in Rset, and has high precision. Several strategies and heuristics have been proposed for this. They broadly fall under two classes: bottom-up [42, 43, 60], or, top-down [170, 195]. In bottom-up a specific rule is generalized, and in top-down a general rule is specialized as elaborated next. In practice, the details of rule-learning algorithms are much more involved and we will present only an outline of the main steps.

2.3.1 Bottom-up Rule Formation

In Bottom-up rule learning the starting rule is a very specific rule covering just the specific instance. This rule has minimal coverage, but 100% precision, and is guaranteed to be nonredundant because it is grown from an instance that is not covered by the existing rule set. This rule is gradually made more general so that the coverage increases with a possible loss of precision. There are many variants on the details of how to explore the space of possible generalizations and how to trade-off coverage with precision. We describe $(\text{LP})^2$ a successful rule-learning algorithm specifically developed for learning entity extraction rules [60].

$(\text{LP})^2$ follows the rule format of Section 2.1.3 where rule actions correspond to an insertion of either a start or an end marker for each entity type. Rules are learnt independently for each action. When inducing rules for an action, examples that contain the action are positive examples; the rest are negative examples. For each tag type T, the following steps are repeatedly applied until there are no uncovered positive examples:

(1) Creation of a seed rule from an uncovered instance.
(2) Generalizations of the seed rule.
(3) Removal of instances that are covered by the new rules.

Creation of Seed Rule: A seed rule is created from a positive instance $\mathbf{x}$ not already covered by the existing rules. A seed rule is just the snippet of w tokens to the left and right of T in $\mathbf{x}$ giving rise to a very specific rule of the form: $x_{i-w} \cdots x_{i-1} x_i \cdots x_{i+w} \to T$, where T appears in position i of $\mathbf{x}$.

Consider the sentence in Figure 1.1 and let T=<PER> and $w = 2$. An example seed rule that will lead to the insertion of T before position i is

({String = "According"} {String = "to"}):pstart {String = "Robert"} {String = "Callahan"} $\to$ insert $\langle$PER$\rangle$ at :pstart

An interesting variant for the creation of seed rules is used in Rapier [42] another popular rule-learning algorithm. In Rapier a seed rule is created from a pair of instances instead of a single instance. This ensures that each selected rule minimally has a coverage of two.

Generalizing Seed Rules: The seed rule is generalized by either dropping a token or replacing the token by a more general feature of the token.

Here are some examples of generalizations of the seed rule above:

- ({String = "According"} {String = "to"}):pstart {Orthography type = "Capitalized word"} {Orthography type = "Capitalized word"} $\rightarrow$ insert ⟨PER⟩ after :pstart
- ({DictionaryLookup = Person}):pb ({DictionaryLookup = Person}) $\rightarrow$ insert ⟨PER⟩ before :Pb

The first rule is a result of two generalizations, where the third and fourth terms are replaced from their specific string forms to their orthography type. The second rule generalizes by dropping the first two terms and generalizing the last two terms by whether they appear in a dictionary of people names. Clearly, the set of possible generalizations is exponential in the number of tokens in the seed rule. Hence, heuristics like greedily selecting the best single step of generalization is followed to reduce the search space. Finally, there is a user-specified cap of k on the maximum number of generalizations that are retained starting from a single seed rule. Typically, the top-k rules are selected sequentially in decreasing order of precision over the uncovered instances. But $(LP)^2$ also allows a number of other selection strategies based on a combination of multiple measures of quality of rules, including its precision, its overall coverage, and coverage of instances not covered by other rules.

2.3.2 Top-down Rule Formation

A well-known rule learning algorithm is FOIL (for First Order Induction Logic) [170] that has been extensively used in many applications of inductive learning, and also in information extraction [7]. Another top-down rule learning algorithm is WHISK [195]. $(LP)^2$ also has a more efficient variant of its basic algorithm above that is top-down.

In a top-down algorithm, the starting rule covers all possible instances, which means it has 100% coverage and poor precision. The starting rule is specialized in various ways to get a set of rules with high precision. Each specialization step ensures coverage of the starting seed instance. We describe the top-down rule specialization method in $(\mathrm{LP})^2$ that follows an Apriori-style [6] search of the increasingly specialized rules. Let R_0 be the most specialized seed rule consisting of conditions at $2w$ positions that is used in bottom-up learning as described in the previous section. The top-down method starts from rules that generalize R_0 at only one of the $2w$ positions. This set is specialized to get a collection of rules such that the coverage of each rule is at least s, a user provided threshold. An outline of the algorithm is given below:

(1) $\mathcal{R}_1$ = set of level 1 rules that impose a condition on exactly one of the $2w$ positions and have coverage at least s.
(2) For level $L = 2$ to $2w$
    (a) $\mathcal{R}_L$ = Rules formed by intersecting two rules from $\mathcal{R}_{L-1}$ that agree on $L - 2$ conditions and differ on only one. This step is exactly like the join step in the Apriori algorithm.
    (b) Prune away rules from $\mathcal{R}_L$ with coverage less than s.

The above process will result in a set of rules, each of which cover R_0 and have coverage at least s. The set k of the most precise of these rules are selected. A computational benefit of the above method is that the coverage of a new rule can be easily computed by intersecting the list of instances that each of the parent rule covers.

Interactive Methods for Rule Induction: In practice, a purely automated data-driven method for rule induction cannot be adequate due to the limited availability of labeled data. The most successful rule-based systems have to provide a hybrid of automated and manual methods. First, the labeled data can be used to find a set of seed rules. Then, the user interacts with the system to modify or tune the rules or to provide more labeled examples. Often, this is a highly iterative process. An important requirement for the success of such a system is fast support for assessing the impact of a rule modification on the available labeled and unlabeled data. See [116] for a description of one such system that deploys a customized inverted index on the documents to assess the impact of each rule change.

Summary

In this section, we presented an overview of rule-based methods to entity extraction. We showed how rule-based systems provide a convenient method of defining extraction patterns spanning over various properties of the tokens and the context in which it resides. One key advantage of a rule-based system is that it is easy for a human being to interpret, develop, and augment the set of rules. One important component of a rule-based method is the strategy followed to resolve conflicts; many different strategies have evolved over the years but one of the most popular of these is ordering rules by priorities. Most systems allow the domain expert to choose a strategy from a set of several predefined strategies. Rules are typically hand-coded by a domain expert but many systems also support automatic learning of rules from examples. We presented two well-known algorithms for rule-learning.

Further Readings

Optimizing the Execution of a Rule-based Extractor: Most rule-based systems are based on regular grammars that can be compiled as a deterministic finite state automata (DFA) for the purposes of efficient processing. This implies that a single pass over the input document can be used to find all possible rule firings. Each input token results in a transition from one state to another based on properties applied on the features attached to the token. However, there are many issues in optimizing such executions further. Troussov et al. [207] show how to exploit the difference in the relative popularity of the states of the DFA to optimize a rule-execution engine. Another significant advancement in rule execution engines is to apply techniques from relational database query optimization to efficient rule execution [179, 190]. We revisit this topic in Section 5.1.

Untyped Entities: Typically, in entity extraction, the type of entities come from a small closed class, that is known in advance. When one is trying to build a knowledge base from open sources, such as the web, it may not be possible to define the set of entity types in advance. In such cases it makes sense to first extract all plausible entities using generic patterns for entity recognition and later figure out the type of the entity [81, 193]. For example, Downey et al. [81] exploit capitalization patterns of a text string and its repeated occurrences on the web to find such untyped entities from webpages.

3

Entity Extraction: Statistical Methods

Statistical methods of entity extraction convert the extraction task to a problem of designing a decomposition of the unstructured text and then labeling various parts of the decomposition, either jointly or independently.

The most common form of decomposition is into a sequence of tokens obtained by splitting an unstructured text along a predefined set of delimiters (like spaces, commas, and dots). In the labeling phase, each token is then assigned an entity label or an entity subpart label as elaborated in Section 3.1. Once the tokens are labeled, entities are marked as consecutive tokens with the same entity label. We call these token-level methods since they assign label to each token in a sequence of tokens and discuss these in Section 3.1.

A second form of decomposition is into word chunks. A common method of creating text chunks is via natural language parsing techniques [137] that identify noun chunks in a sentence. During labeling, instead of assigning labels to tokens, we assign labels to chunks. This method is effective for well-formed natural language sentences. It fails when the unstructured source does not comprise of well formed sentences, for example, addresses and classified ads. A more general method of handling multi-word entities is to treat extraction as a segmentation problem where each segment is an entity. We call these segment-level methods and discuss them in Section 3.2.

Sometimes, decompositions based on tokens or segments, fail to exploit the global structure in a source document. In such cases, context-free grammars driven by production rules, are more effective. We discuss these in Section 3.3.

We discuss algorithms for training and deploying these models in Sections 3.4 and 3.5, respectively.

We use the following notation in this section. We denote the given unstructured input as $\mathbf{x}$ and its tokens as $x_1 \cdots x_n$, where n is the number of tokens in string. The set of entity types we want to extract from $\mathbf{x}$ is denoted as E.

3.1 Token-level Models

This is the most prevalent of statistical extraction methods on plain text data. The unstructured text is treated as a sequence of tokens and the extraction problem is to assign an entity label to each token. Figure 3.1 shows two example sequences of eleven and nine words each. We denote the sequence of tokens as $\mathbf{x} = x_1 \cdots x_n$. At the time of extraction each x_i has to be classified into one of a set $\mathcal{Y}$ of labels. This gives rise to a tag sequence $\mathbf{y} = y_1 \cdots y_n$.

The set of labels $\mathcal{Y}$ comprise of the set of entity types E and a special label "other" for tokens that do not belong to any of the entity types. For example, for segmenting an address record into its constituent

Here is my review of Fermat's last theorem by S. Singh

i	1	2	3	4	5	6	7	8	9	10	11
x	Here	is	my	review	of	Fermat's	last	theorem	by	S.	Singh
y	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}

R. Fagin and J. Helpbern, Belief Awareness Reasoning

i	1	2	3	4	5	6	7	8	9
x	R.	Fagin	and	J.	Helpbern	,	Belief	Awareness	Reasoning
y	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9

Fig. 3.1 Tokenization of two sentences into sequence of tokens.

fields we use $\mathcal{Y}$ ={HouseNo, Street, City, State, Zip, Country, Other}. Since entities typically comprise of multiple tokens, it is customary to decompose each entity label as "Entity_Begin", "Entity_End", and "Entity_Continue." This is popularly known as the BCEO (where B=Begin C=Continue E=End O=Other) encoding. Another popular encoding is BIO that decomposes an entity label into "Entity_Begin", and "Entity_Inside." We will use $\mathcal{Y}$ to denote the union of all these distinct labels and m to denote the size of $\mathcal{Y}$. For example, in the second sentence the correct label for the nine tokens in the BCEO encoding is: Author_Begin, Author_End, Other, Author_Begin, Author_End, Other, Title_Begin, Title_Continue, Title_End.

Token labeling can be thought of as a generalization of classification where instead of assigning a label to each token, we assign labels to a sequence of tokens. Features form the basis of this classification process. We present an overview of typical entity extraction features in Section 3.1.1. We then present models for predicting the label sequence given the features of a token sequence.

3.1.1 Features

A typical extraction task depends on a diverse set of clues capturing various properties of the token and the context in which it lies. Each of these can be thought of as a function $f : (\mathbf{x}, y, i) \mapsto R$ that takes as argument the sequence $\mathbf{x}$, the token position i, and the label y that we propose to assign x_i, and returns a real-value capturing properties of the ith token and tokens in its neighborhood when it is assigned label y. Typical features are fired for the ith token x_i, for each token in a window of w elements around x_i, and for concatenation of words in the window.

We list common families of token properties used in typical extraction tasks. We will soon see that the feature framework provides a convenient mechanism for capturing a wide variety of clues that are needed to recognize entities in noisy unstructured sources.

Word Features: The surface word itself is often a very strong clue for the label it should be assigned. Two examples of token features at position 2 of the second sequence $\mathbf{x}$ in Figure 3.1 is

$$\begin{aligned} f_1(y,\mathbf{x},i) &= [\![x_i \text{ equals "Fagin"}]\!] \cdot [\![y = \text{Author}]\!] \\ f_2(y,\mathbf{x},i) &= [\![x_{i+1} \text{ equals "and"}]\!] \cdot [\![y = \text{Author}]\!], \end{aligned}$$

where $[\![P]\!] = 1$ when predicate P is true and 0 otherwise.

Orthographic Features: Many valuable features for entity extraction are derived from various orthographic properties of the words, viz, its capitalization pattern, the presence of special symbols and alphanumeric generalization of the characters in the token.

Two examples of orthographic features are

$$\begin{aligned} f_3(y,\mathbf{x},i) &= [\![x_i \text{ matches INITIAL\_DOT}]\!] \cdot [\![y = \text{Author}]\!] \\ f_4(y,\mathbf{x},i) &= [\![x_i x_{i+1} \text{ matches INITIAL\_DOT CapsWord }]\!] \\ &\quad \cdot [\![y = \text{Author}]\!]. \end{aligned}$$

Feature f_3 fires when a token x_i is an initial followed by a dot, and it is being labeled Author. For the second sentence in Figure 3.1 this fires at position 1 and 4 and for the first at position 10. Feature f_4 fires when a token x_i is labeled author and x_i is a dotted initial and the word following it is a capitalized word. This feature fires at the same positions as f_3.

Dictionary Lookup Features: As mentioned in Section 1.5.1, there is often a database of entities available at the time of extraction. Match with words in a dictionary is a powerful clue for entity extraction. This can be expressed in terms of features as follows:

$$\begin{aligned} f_5(y,\mathbf{x},i) &= [\![x_i \text{ in Person\_dictionary}]\!] \cdot [\![y = \text{Author}]\!] \\ f_6(y,\mathbf{x},i) &= [\![x_i \text{ in City\_list}]\!] \cdot [\![y = \text{State}]\!]. \end{aligned}$$

3.1.2 Models for Labeling Tokens

A number of different models have been proposed for assigning labels to the sequence of tokens in a sentence. An easy model is to independently assign the label y_i of each token x_i using features derived from the token x_i and its neighbors in $\mathbf{x}$. Any existing classifier such as a logistic classifier or a Support Vector Machine (SVM) can be used to classify each token to the entity type it belongs. However, in typical extraction tasks the labels of adjacent tokens are seldom independent of each other. In the example in Figure 3.1, it might be difficult to classify "last" as being a word from a book title. However, when the word to the left and right of it is labeled a book title, it makes sense to label "last" as a book title too. This has led to a number of different models for capturing the dependency between the labels of adjacent words. The simplest of these is the ordered classification method that assigns labels to words in a fixed left to right order where the label of a word is allowed to depend on the label of the word to its left [200, 79]. Other popular choices were Hidden Markov Models (HMMs) [3, 20, 25, 171, 189] and Maximum entropy taggers [26, 177] also called maximum entropy Markov models (MEMM) [143] and conditional Markov models (CMM) [118, 135]. The state-of-the-art method for assigning labels to token sequences is Conditional Random Fields (CRFs) [125]. CRFs provide a powerful and flexible mechanism for exploiting arbitrary feature sets along with dependency in the labels of neighboring words. Empirically, they have been found to be superior to all the earlier proposed methods for sequence labeling. We elaborate on CRFs next.

Conditional Random Fields: A Conditional Random Field (CRF) models a single joint distribution $\Pr(\mathbf{y}|\mathbf{x})$ over the predicted labels $\mathbf{y} = y_1 \cdots y_n$ of the tokens of $\mathbf{x}$. The tractability of the joint distribution is ensured by using a Markov random field [119] to express the conditional independencies that hold between elements y_i of $\mathbf{y}$. In typical extraction tasks, a chain is adequate for capturing label dependencies. This implies that the label y_i of the ith token is directly influenced only by the labels of tokens that are adjacent to it. In other words, once the label y_{i-1} is fixed, label y_{i-2} has no influence on label y_i.

The dependency between the labels of adjacent tokens is captured by a scoring function $\psi(y_{i-1}, y_i, \mathbf{x}, i)$ between nodes y_{i-1} and y_i. This score is defined in terms of weighted functions of features as follows:

$$\psi(y_{i-1}, y_i, \mathbf{x}, i) = e^{\sum_{k=1}^{K} w_k f_k(y_i, \mathbf{x}, i, y_{i-1})} = e^{\mathbf{w} \cdot \mathbf{f}(y_i, \mathbf{x}, i, y_{i-1})}. \tag{3.1}$$

With these per-edge scores, the conditional distribution of a label sequence $\mathbf{y}$ given a token $\mathbf{x}$ is given as

$$\Pr(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{1}{Z(\mathbf{x})}\prod_{i=1}^{n}\psi(y_{i-1},y_i,\mathbf{x},i) = \frac{1}{Z(\mathbf{x})}e^{\sum_{i=1}^{n}\mathbf{w}\cdot\mathbf{f}(y_i,\mathbf{x},i,y_{i-1})}. \tag{3.2}$$

The term $Z(\mathbf{x})$ is a normalizing constant and is equal to $\sum_{\mathbf{y}} e^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})}$, where $\mathbf{f}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}\mathbf{f}(y_i,\mathbf{x},i,y_{i-1})$ the sum of the feature vector over all token positions of the sequence. Some subset of these features can be simplified further to depend only on the current state and are independent of the previous state. We will refer these as *state features* and denote these by $f(y_i,\mathbf{x},i)$ when we want to make the distinction explicit. The term *transition features* refers to the remaining features that are not independent of the previous label.

Given m labels and a sequence $\mathbf{x}$ of length n, there can be $O(m^n)$ possible labeling of $\mathbf{x}$. This exponential complexity is cut down to $O(nm^2)$ by the famous dynamic programming-based *Viterbi Algorithm* that is described in Section 3.5.1. The training algorithm is more involved and we discuss these in Section 3.4.

3.2 Segment-level Models

In segment-level methods, the output is a sequence of segments, with each segment defining an entity, rather than a sequence of labels as in earlier token-level models. More formally, a segmentation $\mathbf{s}$ of an input sequence of length n is a sequence of segments $s_1 \cdots s_p$ such that the last segment ends at n, the first segment starts at 1, and segment s_{j+1} begins right after segment s_j ends. Each segment s_j consists of a *start position* l_j, an *end position* u_j, and a *label* $y_j \in \mathcal{Y}$. In Figure 3.2, we show an ideal segmentation of the second sentence in Figure 3.1. The segmentation identifies three entities: two authors and one title.

R. Fagin and J. Helpbern, Belief Awareness Reasoning

i	1	2	3	4	5	6	7	8	9
$\mathbf{x}$	R. Fagin		and	J. Helpbern		,	Belief Awareness Reasoning		
(l_j,u_j,y_j)	$1,2,A$		$3,3,O$	$4,5,A$		$6,6,O$	$7,9,T$		

Fig. 3.2 Segmentation of a sentence.

In a segment-level model, features are defined over segments comprising of multiple tokens forming an entire entity string. This allows for the use of more powerful features than token-level models because features can now capture joint properties over all the tokens forming part of an entity. Like in the case of sequence models, the label of a segment depends on the label of the previous segment and the properties of the tokens comprising this segment. Thus a feature for segment $s_j = (y_j, l_j, u_j)$ is of the form $f(y_j, y_{j-1}, \mathbf{x}, l_j, u_j)$.

We present examples of segment level features that cannot be expressed in the earlier token-based models. Note that in a segment-level model, one would include token-level features in addition to the segment-level features. For example, it is often useful to define token-level features tied to the start and ending words of a segment.

3.2.1 Entity-level Features

Similarity to an Entity in the Database: Most entities consist of multiple tokens. In sequence models, it is not possible to define a feature that measures the similarity of an entity to the entire entity in a database. For example, if we have a database of company names, and a segment of our input **s** matches one of the entries entirely, then this gives a strong clue to mark the segment as a company name. In a sequence model, we cannot enforce arbitrary segment of text to take the same label. Thus, we can define a segment-level feature of the form:

$$\begin{aligned} f(y_i, y_{i-1}, \mathbf{x}, 3, 5) = {} & [\![x_3 x_4 x_5 \text{ appears in a list of journals}]\!] \\ & \cdot [\![y_i = \text{journal}]\!]. \end{aligned}$$

In general, since unstructured data is noisy instead of defining boolean features that measure exact match, it is more useful to define real-valued features that quantify the similarity between them [65]. For example, the feature below captures the maximum TF-IDF similarity between the text span $x_3 x_4 x_5$ and an entry in a list of Journal names.

$$\begin{aligned} f(y_i, y_{i-1}, \mathbf{x}, 3, 5) = {} & \max_{J \in \text{journals}} \text{TF-IDF-similarity}(x_3 x_4 x_5, J) \\ & \cdot [\![y_i = \text{journal}]\!]. \end{aligned}$$

Length of the Entity Segment: Another useful segment-level feature is its length as this helps capture typical length distributions of the entity. Thus, one can define features of the form that will be fired on all segments of length five when they are labeled as title

$$f(y_i, y_{i-1}, \mathbf{x}, l, u) = [\![u - l = 5]\!] \cdot [\![y_i = \text{title}]\!].$$

3.2.2 Global Segmentation Models

As in sequence models, it is possible to define a probability distribution over different segmentation of an input $\mathbf{x}$ as follows:

$$\Pr(\mathbf{s}|\mathbf{x}, \mathbf{W}) = \frac{1}{Z(\mathbf{x})} e^{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{s})}, \tag{3.3}$$

where $\mathbf{w}$ is a weight vector for feature vector $\mathbf{f}$, $\mathbf{f}(\mathbf{x}, \mathbf{s}) = \sum_{j=1}^{|\mathbf{s}|} \mathbf{f}(y_j, \mathbf{x}, l_j, u_j, y_{j-1})$, and $Z(\mathbf{x}) = \sum_{\mathbf{s}'} e^{\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{s}')}$ is the normalization term.

During *inference*, the goal is to find a segmentation $\mathbf{s} = s_1 \cdots s_p$ of the input sequence $\mathbf{x} = x_1 \cdots x_n$ such that $\mathbf{w} \cdot \mathbf{f}(\mathbf{x}, \mathbf{s})$ is maximized. It is possible to find the best segmentation efficiently using dynamic programming as we show later in Section 3.5.2.

3.3 Grammar-based Models

Some entity extraction systems require a better interpretation of the structure of the source, than is possible with segmentation models [202, 213]. For example, we expect author names in a single citation to be formatted similarly; either all author first names are initialized, or all of them are in full. Such a global structure cannot be exploited in any of the earlier models that can only capture dependency in the labels of adjacent words.

A grammar-based model uses a set of production rules, like in context-free grammars of programming language, to express the global structure of the entity. The productions are defined over terminals, which in this case are tokens in the unstructured source, and non-terminals that include entity labels and other additional meta-labels to capture the grouping amongst subset of labels. For example, to capture the style homogeneity amongst author names in a citation we can define a set of production rules which along with a scoring model that we will define shortly will achieve the desired effect.

R: S → AuthorsLF | AuthorsFL
R0: AuthorsLF → NameLF_Separator AuthorsLF
R1: AuthorsFL → NameFL_Separator AuthorsFL
R2: AuthorsFL → NameFL
R3: AuthorsLF → NameLF
R4: NameLF_Separator → NameLF Punctuation
R5: NameFL_Separator → NameFL Punctuation
R6: NameLF → LastName First_Middle
R7: NameFL → First_Middle LastName

The output of the extraction process is a parse tree. Unlike in programming languages, there can be many valid parses for a token sequence. Each output tree is assigned a score that decomposes over the productions in the parse tree. Early methods of assigning scores were based on a generative model where each nonterminal has a multinomial probability distribution over production rules where it is the head. More recently, analogously to the shift from HMMs to discriminative learners like CRFs, grammars are also scored discriminatively. Each production of the form: $R \to R_1 R_2$ is scored as follows:

$$s(R) = s(R_1) + s(R_2) + \mathbf{w}\cdot\mathbf{f}(R, R_1, R_2, \mathbf{x}, l_1, r_1, r_2), \qquad (3.4)$$

where (l_1, r_1) and $(r_1 + 1, r_2)$ are the text spans in $\mathbf{x}$ that R_1 and R_2 cover, respectively. The score of a node depends on the production used at the node and the text spans that its children R_1 and R_2 cover. It does not depend on the subtree underneath R_1 and R_2. In the base case we have terminal nodes whose score calculated as $\mathbf{w}\cdot\mathbf{f}(R, \mathbf{x}, l, r)$ indicates the score of assigning label R to the tokens between l and r. We show an example to illustrate the scoring of a grammar-based model such as the above. Consider a string "Peter Haas, George John" of five tokens. One of the many possible parses of the string is

R0→ R4 R3
R4→ R6 x_3
R3 → R6

R6$\rightarrow$ x_1x_2
R6$\rightarrow$ x_4x_5

The total score of this tree $\mathbf{w}\cdot\mathbf{f}(R0,R4,R3,\mathbf{x},1,3,5) + \mathbf{w}\cdot\mathbf{f}(R4,R6,$ Punctuation$,\mathbf{x},1,2,3) + \mathbf{w}\cdot\mathbf{f}(R3,R6,\phi,\mathbf{x},1,2,2) + \mathbf{w}\cdot\mathbf{f}(R6,\mathbf{x},1,2) + \mathbf{w}\cdot\mathbf{f}(R6,\mathbf{x},3,4)$. In this method of scoring, any arbitrary set of features can be used to capture the affinities between tokens and their labels. For example, we can use any of the features described in Section 3.1.1 to score labeling tokens "George John" as a name where the last name appears first.

The above method of scoring makes it possible to find the highest scoring parse tree in polynomial time. We present the inference algorithm in Section 3.5. Grammar-based models, although more expressive, are not as popular as the other methods because of the high cost of finding an optimal parse tree.

3.4 Training Algorithms

We now discuss training algorithms that apply to the different feature-based models introduced in Sections 3.1 through 3.3.

This section and the next, which cover the details of training and inference, can be involved for someone not already familiar with similar machine learning topics. We refer the reader to this text book [119] for a gentler introduction.

The model outputs a $\mathbf{y}$ for which the score $\mathrm{s}(\mathbf{y}) = \mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})$ is maximum where $\mathbf{f}(\mathbf{x},\mathbf{y})$ is a feature vector defined jointly over the output $\mathbf{y}$ and input $\mathbf{x}$. Models differ in the form of the output $\mathbf{y}$ and the feature vector. For sequence models, $\mathbf{y}$ refers to a sequence of labels, for segment-level models $\mathbf{y}$ is a segmentation of $\mathbf{x}$, for grammar-based models $\mathbf{y}$ is a parse tree.

Let $D = \{(\mathbf{x}_\ell,\mathbf{y}_\ell)\}_{\ell=1}^N$ denote the labeled training set. Broadly there are two types of methods for training:

(1) Likelihood-based training discussed in Section 3.4.1.
(2) Max-margin training discussed in Section 3.4.2.

3.4.1 Likelihood Trainer

This method is applicable when $\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})$ is used to impose a probability distribution on the output $\mathbf{y}$ as follows:

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})} \tag{3.5}$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}} e^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})}$. The goal during training is to choose a weight vector $\mathbf{w}$ such that probability of the correct output as given in the training data is maximized. Let $L(\mathbf{w})$ denote the log of the probability of the training data when the weight vector is $\mathbf{w}$. We can write it as:

$$L(\mathbf{w}) = \sum_{\ell} \log \Pr(\mathbf{y}_\ell|\mathbf{x}_\ell,\mathbf{w}) = \sum_{\ell} (\mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}_\ell) - \log Z_{\mathbf{w}}(\mathbf{x}_\ell)).$$

Routinely, a term such as $-\frac{||\mathbf{w}||^2}{C}$ is added to prevent overfitting by penalizing large swings in the parameter values. This term has the effect of performing a soft form of feature selection. Ideally, we would like to maximize accuracy while using the smallest number of features. Since this leads to a difficult discrete optimization problem we instead use $||\mathbf{w}||^2$ weighted by a user-provided constant $1/C$. The training objective can then be expresssed as

$$\max_{\mathbf{w}} \sum_{\ell} (\mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}_\ell) - \log Z_{\mathbf{w}}(\mathbf{x}_\ell)) - ||\mathbf{w}||^2/C.$$

The above equation is concave in $\mathbf{w}$, and can thus be maximized by gradient ascent type of methods that iteratively move toward the optimum $\mathbf{w}$. At each iteration they require the computation of the gradient $\nabla L(\mathbf{w})$ of the objective which can be calculated as:

$$\begin{aligned}\nabla L(\mathbf{w}) &= \sum_{\ell} \mathbf{f}(\mathbf{x}_\ell,\mathbf{y}_\ell) - \frac{\sum_{\mathbf{y}'} \mathbf{f}(\mathbf{y}',\mathbf{x}_\ell)\mathbf{e}^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}')}}{Z_{\mathbf{w}}(\mathbf{x}_\ell)} - 2\mathbf{w}/C \\ &= \sum_{\ell} \mathbf{f}(\mathbf{x}_\ell,\mathbf{y}_\ell) - E_{\Pr(\mathbf{y}'|\mathbf{w},\mathbf{x}_\ell)}\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}') - 2\mathbf{w}/C.\end{aligned}$$

The first term is easy to compute. However, the second term that involves a summation over exponentially many outputs $\mathbf{y}'$ require special algorithms that exploit the decomposability of the feature vector.

We will show how these are computed in Section 3.5 where we discuss inference algorithms.

The overall training algorithm appears in Figure 3.1. The running time of the algorithm is $O(INn(m^2 + K))$, where I is the total number of iterations. This basic algorithm has been improved substantially and now there are many faster variants like semi-Newton methods [132, 136] and stochastic gradient methods [27, 214].

Algorithm 3.1. Train($D = \{(\mathbf{x}_\ell, \mathbf{y}_\ell)\}_{\ell=1}^N$, $\mathbf{f} : f_1 \cdots f_K$)

```
1. Output:  $\mathbf{w} = \text{argmax} \sum_{\ell=1}^N (\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}_\ell) - \log Z_\mathbf{w}(\mathbf{x}_\ell)) - ||\mathbf{w}||^2/C$
2. Initialize $\mathbf{w}^0 = \mathbf{0}$
3. for $t = 1 \cdots$ maxIters do
4.   for $\ell = 1 \cdots N$ do
5.     $g_{k,\ell} = f_k(\mathbf{x}_\ell, \mathbf{y}_\ell) - E_{\Pr(\mathbf{y}'|\mathbf{w},\mathbf{x}_\ell)} f_k(\mathbf{x}_\ell, \mathbf{y}')$  $k = 1 \cdots K$
6.   $g_k = \sum_\ell g_{k,\ell}$  $k = 1 \cdots K$
7.   $w_k^t = w_k^{t-1} + \gamma_t(g_k - 2w_k^{t-1}/C)$  $k = 1 \cdots K$  ($\gamma_t$ = learning rate)
8.   Exit if $||\mathbf{g}|| \approx zero$
```

3.4.2 Max-margin Training

Max-margin trainers [201, 208] are an extension of support vector machines for training structured models. They are more generally applicable because they require only the computation of the output with the highest score. In contrast, for likelihood trainers it is necessary to compute the expected value of features, which is sometimes difficult.

The goal during max-margin training is to find a $\mathbf{w}$ such that the score $\mathbf{w} \cdot \mathbf{f}(\mathbf{x}_\ell, \mathbf{y}_\ell)$ of the correct labeling $\mathbf{y}_\ell$ has a margin of at least $\text{err}(\mathbf{y}, \mathbf{y}_\ell)$ more than the score of a labeling $\mathbf{y}$, where $\text{err}(\mathbf{y}, \mathbf{y}_\ell)$ is a user-provided error function that indicates the penalty of outputting $\mathbf{y}$ when the correct label is $\mathbf{y}_\ell$. An example of an error function is the Hamming error that measures the number of tokens where the label is incorrect. The max-margin objective can be formulated as a constrained optimization problem as follows:

$$\begin{aligned}
&\min_{\mathbf{w},\xi} C\sum_{\ell=1}^{N}\xi_\ell + \frac{1}{2}||\mathbf{w}||^2\\
&\qquad \text{s.t. } \mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}_\ell) \geq \text{err}(\mathbf{y},\mathbf{y}_\ell) + \mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}) - \xi_\ell \quad \forall\mathbf{y}\neq\mathbf{y}_\ell, \ell : 1\cdots N\\
&\qquad\quad \xi_\ell \geq 0 \ \ell : 1\cdots N
\end{aligned} \tag{3.6}$$

The variables ξ_ℓ capture the gap between the desired margin and the margin achieved by the current values of **w**. The training objective is to minimize this gap. The above program is convex in $\mathbf{w},\xi$, so a local minima is the global minima. But the number of constraints is exponential in the length of each sequence. Clearly, any method that explicitly enumerates these constraints cannot be practical. A cutting plane algorithm is used to find the best **w** iteratively. A sketch of the algorithm appears in Figure 3.2.

Algorithm 3.2. Train($D = \{(\mathbf{x}_\ell,\mathbf{y}_\ell)\}_{\ell=1}^{N}$, $\mathbf{f} : f_1\cdots f_K$), C

```
1. Output: $\mathbf{w} = \text{argmin}\frac{1}{2}||\mathbf{w}||^2 + C\sum_{\ell=1}^{N}\max_{\mathbf{y}}(\text{err}(\mathbf{y},\mathbf{y}_\ell)+$
   $\mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}) - \mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}_\ell))$
2. Initialize $\mathbf{w}^0 = \mathbf{0}$, Active constraints = Empty.
3. for $t = 1\cdots$maxIters do
4.    for $\ell = 1\cdots N$ do
5.       $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}}(\text{err}(\mathbf{y},\mathbf{y}_\ell) + \mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}))$
6.       if $\mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\mathbf{y}_\ell) < \text{err}(\hat{\mathbf{y}},\mathbf{y}_\ell) + \mathbf{w}\cdot\mathbf{f}(\mathbf{x}_\ell,\hat{\mathbf{y}}) - \xi_\ell - \epsilon$ then
7.          Add $(\mathbf{x}_\ell,\hat{\mathbf{y}})$ to the set of constraints.
8.          $\mathbf{w},\xi$ = solve QP with active constraints.
9.    Exit if no new constraint added.
```

The two time consuming steps in the above algorithms are

- Step 5 that finds the most violating constraint. To solve this efficiently, the err(.,.) function is assumed to be decomposable, in the same manner as the features themselves. A typical decomposable err function is the Hamming error, which just counts the number of positions in **y** and $\mathbf{y}_\ell$ that are different.

- Step 8 that solves the constrained quadratic program to find the new values of the variables. Many methods are available to solve this efficiently. A popular method is to solve a dual of the program that turns out to be simpler than the primal [208]. Recently, other methods have been proposed including stochastic online methods [24], and extra-gradient methods [203].

3.5 Inference Algorithms

We encountered two kinds of inference queries when training and deploying the different models.

- Highest scoring (MAP) labeling: Find $\mathbf{y}^* = \text{argmax}_{\mathbf{y}}\ \mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})$.
- Expected feature values: Find $\sum_{\mathbf{y}} \mathbf{f}(\mathbf{x},\mathbf{y})\Pr(\mathbf{y}|\mathbf{x})$, where $\Pr(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}e^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})}$.

The key to solving both these efficiently, in spite of the exponential number of possible values of $\mathbf{y}$, is that the feature vector $\mathbf{f}(\mathbf{x},\mathbf{y})$ decomposes over finer substructures in $\mathbf{y}$. This enables us to design a dynamic programming algorithm based on the following general principle. Let S_1 and S_2 be a disjoint decomposition of the entire output S that is, $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \phi$. Let $S_3 \subset S_1$ be the smallest part of S_1 such that all features spanning S_1 and S_2 involve only the subset in S_3. That is, there are no features involving a label in S_2 and a label from $S_1 - S_3$.

We can find the MAP score $\mathcal{V}(S)$ over S recursively as:

$$\mathcal{V}(S) = \max_{\text{label } \mathbf{y}' \text{ of } S_3} \mathcal{V}(S_1|S_3 = \mathbf{y}') + \mathcal{V}(S_2|S_3 = \mathbf{y}'). \tag{3.7}$$

As long as the number of possible labelings of the common part S_3 is small, this equation will be efficient. The same principle holds for finding the expected values.

Sections 3.5.1, 3.5.2, and 3.5.3 elaborate on MAP labeling for the special cases of sequential labeling, segmentation, and parsing, respectively. Section 3.5.4 elaborates on computing expected features values for sequence models.

3.5.1 MAP for Sequential Labeling

In sequence labeling the feature vector decomposes over labels of adjacent positions, that is, $\mathbf{f}(\mathbf{x},\mathbf{y}) = \sum_i^{|\mathbf{x}|} \mathbf{f}(y_i,\mathbf{x},i,y_{i-1})$. Let $\mathcal{V}(i|y)$ denote the best score of the sequence from 1 to i with the ith position labeled y. We can express its value using dynamic programming as follows:

$$\mathcal{V}(i|y) = \begin{cases} \max_{y'} \ \mathcal{V}(i-1,y') + \mathbf{w} \cdot \mathbf{f}(y,\mathbf{x},i,y') & \text{if } i > 0 \\ 0 & \text{if } i = 0. \end{cases} \tag{3.8}$$

The best label then corresponds to the path traced by $\max_y \mathcal{V}(n|y)$, where n is the length of $\mathbf{x}$. The running time of this algorithm is $O(nm^2)$, where m is the number of labels. This is well-known as the Viterbi algorithm.

3.5.2 MAP for Segmentations

In segmentation tasks, $\mathbf{y}$ denotes a segmentation of $\mathbf{x}$. Let $\mathbf{y} = s_1 \dots s_p$, where each $s_j = (t_j,u_j,y_j)$ with t_j = segment start position, u_j = segment end position, and y_j = segment label. The feature vector decomposes over segments and the label of its previous segment as $\mathbf{f}(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{p} \mathbf{f}(\mathbf{x},s_j,y_{j-1})$. Let $\mathcal{V}(i|y)$ denote the best score of segmentations ending at i with label y.

$$\mathcal{V}(i|y) = \begin{cases} \max_{y'} \max_{i'=i-L \cdots i-1} \ \mathcal{V}(i'|y') + \mathbf{w} \cdot \mathbf{f}(\mathbf{x},y,i'+1,i,y') & \text{if } i > 0 \\ 0 & \text{if } i = 0 \\ -\infty & \text{if } i < 0, \end{cases}$$

where L is the size of the largest segment. The best segmentation then corresponds to the path traced by $\max_y \mathcal{V}(n|y)$. This runs in time $O(nLm^2)$.

In entity extraction, segmentation models use a mix of token-level and segment-level features with the former comprising a much larger fraction of the total feature set. In such cases it is possible to make inference over segmentation models comparable to that over sequence models as shown in [186].

3.5.3 MAP for Parse Trees

In this case, the output $\mathbf{y}$ denotes a parse tree. Each internal node j of the tree denotes a rule R_j and the span of text (t_j,u_j) on which it applies. The feature vector $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes over the nodes in tree $\mathbf{y}$ and spans of its immediate children. We assume trees are binary for simplicity and denote the children of R_j as R_{j1} and R_{j2}. Thus, $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{\text{node } j \in \mathbf{y}} \mathbf{f}(\mathbf{x}, R_j, R_{j1}, R_{j2}, t_j, u_j, u_{j_1})$, where u_{j_1} denotes the ending position of the first child of the node. Let $\mathcal{V}(i, j|y)$ denote the score of the best tree rooted at nonterminal y over text span (i, j).

$$\mathcal{V}(i,j|y) = \begin{cases} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, y, i, j) & \text{if } y \text{ is terminal} \\ \max_{i<i'<j} \max_{R':y\mapsto R_1R_2} \ \mathcal{V}(i,i'|R_1) & \\ \qquad +\mathcal{V}(i'+1, j|R_2) + \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, R', R_1, R_2, i, j, i') & \end{cases}$$

The best tree is $\max_y \mathcal{V}(1, n, y)$, where y goes over all possible nonterminals. The running time is $O(n^3 m')$, where m' denotes the total number of nonterminals and terminals.

3.5.4 Expected Features Values for Sequential Labelings

Consider first the simpler problem of computing $Z(\mathbf{x}) = \sum_{\mathbf{y}} e^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y})}$. Again, we exploit the decomposability of $\mathbf{f}(\mathbf{x}, \mathbf{y})$ to define a dynamic program over partial solutions. Let $\alpha(i, y)$ denote the value of $\sum_{\mathbf{y}' \in \mathbf{y}_{i:y}} \mathbf{e}^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y}')}$, where $\mathbf{y}_{i:y}$ denotes all label sequences from 1 to i with the ith position labeled y. For $i > 0$, this can be expressed recursively as

$$\alpha(i, y) = \sum_{y' \in \mathcal{Y}} \alpha(i-1, y')\mathbf{e}^{\mathbf{w}\cdot\mathbf{f}(y,\mathbf{x},i,y')}$$

with the base cases defined as $\alpha(0, y) = 1$. The value of $Z_{\mathbf{w}}(\mathbf{x})$ can then be written as $Z_{\mathbf{w}}(\mathbf{x}) = \sum_y \alpha(n, y)$.

A similar approach can be used to compute the expectation $\sum_{\mathbf{y}'} \mathbf{f}(\mathbf{x}, \mathbf{y}')\mathbf{e}^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y}')}$. For the kth component of $\mathbf{f}$, let $\eta^k(i, y)$ be the value of the sum $\sum_{\mathbf{y}' \in \mathbf{y}_{i:y}} f_k(\mathbf{x}, \mathbf{y}')\mathbf{e}^{\mathbf{w}\cdot\mathbf{f}(\mathbf{x},\mathbf{y}')}$, restricted to the part of the label ending at position i. The following recursion can then be used to compute $\eta^k(i, y)$:

$$\eta^k(i, y) = \sum_{y' \in \mathcal{Y}} (\eta^k(i-1, y') + \alpha(i-1, y') f_k(y, \mathbf{x}, i, y'))e^{\mathbf{w}\cdot\mathbf{f}(y,\mathbf{x},i,y')}.$$

Finally, we let $E_{\Pr(\mathbf{y}'|\mathbf{w})} f_k(\mathbf{x}, \mathbf{y}') = \frac{1}{Z_{\mathbf{w}}(\mathbf{x})} \sum_y \eta^k(n, y)$.

The space requirements here can be reduced from $Km + nm$ to $K + nm$, where K is the number of features, by using a backward recursion to computing an appropriate set of $\beta(i,y)$ values defined, analogously to the α terms, as $\beta(i,y) = \sum_{\mathbf{y}' \in B_{i:y}} \mathbf{e}^{\mathbf{w}\cdot\mathbf{f}(\mathbf{y}',\mathbf{x})}$, where $B_{i:y}$ denotes all label sequences from $i+1$ to n with the ith position labeled y.

$$\beta(i,y) = \sum_{y' \in \mathcal{Y}} \beta(i+1,y')\mathbf{e}^{\mathbf{w}\cdot\mathbf{f}(y',\mathbf{x},i+1,y)}.$$

Now, the expected value of a feature f_k can be computed more simply as:

$$E(f_k(y,\mathbf{x},i,y')) = \alpha(i-1,y')e^{\mathbf{w}\cdot\mathbf{f}(y,\mathbf{x},i,y')}\beta(i,y).$$

Summary

In this section, we studied models for entity extraction using statistical models. There is a long history behind the development of such models and many different models have been proposed over the years. The most prominent of these are Maximum entropy taggers or Conditional Markov Models (CMMs), Hidden Markov Models (HMMs), and Conditional Random Fields (CRFs). CRFs are now established as the state-of-the-art methods and have shown clear advantages over CMMs and HMMs both theoretically and empirically [118, 163] because of the ease with which diverse set of clues can be exploited as features. There are many existing implementations of CRFs [92, 146, 185] that can be easily customized by modifying only the feature set. Segment-level models and the grammar-based models provide all the flexibility of CRFs and more, but they are not as popular because of the increased overheads of inference and training.

Further Readings

The topic of statistical learning for information extraction is relatively new and there continues to be much active research on many different aspects that we have not covered in this section. We present a brief overview of some of these.

Active Learning: The accuracy of a statistical model crucially depends on finding adequate amount of labeled data for training the model parameters. Since collecting training data often requires painstaking manual effort, there has been much work on reducing this effort. Active learning is one technique for reducing labeling effort that is used with a limited labeled and a large unlabeled pool of instances. The labeled set forms the training data for an initial preliminary classifier. The active learner seeks out from the unlabeled pool those instances which when labeled will help strengthen the classifier at the fastest possible rate. This is an important conceptual problem in machine learning and there is much ongoing work on developing theoretically sound methods for active learning [12, 13, 66, 76, 95]. Specifically, for information extraction studies [68, 155, 169, 205] have shown that active learning can significantly reduce the number of examples that need to be manually labeled.

Bootstrapping from Structured Data: Another practical method of addressing the problem of limited training data is to bootstrap models from existing structured entities, which are often readily available. This requires very careful handling, particularly when unstructured labeled is available along with structured databases. When there is only structured data at one's disposable, the technique proposed in [3] would be useful. Methods of training with a mix of structured and unstructured data for HMMs are discussed in [189], for CRFs are discussed in [138], and for rule-based systems are discussed in [99].

Transfer Learning and Domain Adaptation: Statistical learning techniques crucially depend on the training data being representative of the distribution on which the trained model is deployed. This assumption often does not hold when extraction systems get deployed on a large scale. A topic of much recent interest now is domain adaptation [23, 108, 188] where labeled data from some domain is used to train a model that maximizes accuracy in a target domain for which we only have unlabeled data available. Another related variant is transfer learning [9, 52, 77, 129] where the goal is to use training data from a related domain, along with training data from the target domain, to train the target classifier. A popular technique is to use the classifier in the related domain to define a prior [9, 52, 129] for the classifier trained using the in-domain data.

Collective Inference: Most information extraction (IE) models are based on sequential models that capture the dependency between labels of words. Recently, several researchers [31, 93, 101, 120, 199] have reported increased accuracy of collectively labeling repeated words within a document or across multiple documents. In the corresponding graphical model this leads to additional edges between non-adjacent positions that share a word.

4

Relationship Extraction

In many extraction tasks, there is an explicit need to relate the extracted entities because the unstructured source is not naturally partitioned into records. For example, it is not just enough to find occurrences of company names and people names in a news article but also to identify if there is a "is acquired by" relationship between pairs of companies, or, a "is appointed CEO of" relationship between a person and company, or "is employee of" relationship between a person and an organization. Figure 1.1 shows instances of the extraction of two relationships from a news article.

The problem of relationship extraction has been studied extensively on natural language text, including news articles [1], scientific publications [166], Blogs, emails [113], and sources like Wikipedia [196, 197] and the general web [4, 14]. As in entity extraction, much of the impetus to relationship extraction research was provided by the various competitions: starting with the MUC competitions [57, 100], and later the ACE task [1] and the BioCreAtIvE II Protein–Protein Interaction tasks [22, 209]. For example, the ACE task defines five top-level relations: "located at", "near", "part", "role", and "social" over pairs from five top-level entity types "person", "organization", "facility", "location", and, "geo-political entity." The "located at" relation relates "person" or "organization" with "location". The "role" relation links a "person" to an "organization", and so on. In the Bio-medical literature three kinds of relationship extractions are common: gene-disease relations, protein–protein interaction, and subcellular regularizations. The NAGA knowledge base [196, 197] comprises of 26 relationships such as isA, bornInYear, establishedInYear, hasWonPrize, locatedIn, politicianOf, actedIn, discoveredInYear, discoveredBy, and isCitizenOf extracted from sources such as Wikipedia.

In this section, we concentrate on the extraction of binary relationships, although in general relationships can be multi-way involving three or more entities. Two popular examples of multi-way relationships are: Event Extraction and Semantic Role Labeling [124]. Another term for multi-way relationship extraction is record extraction. We will present a brief overview of techniques for multiway relations at the end of this section.

The binary relationship extraction problem can be posed at three different levels. The first case is where entities are preidentified in the unstructured text, and for a given fixed pair of entities we need to find out the type of relationship that exists between the pair. The second case is when we are given a relationship type r and an entity name e, and our goal is to extract the entities with which e has relationship r. The third case is where the unstructured corpus is large and open-ended like the web where we cannot assume that entity pairs are marked. Given a fixed relationship type r, our goal is to extract all instances of entity pairs that have relationship r between them through appropriate filtering and recognition techniques. Most of the early work in relationship extraction has been on the first case and we review the main ideas that have emerged in Section 4.1. In Section 4.2, we present techniques for relationship extraction in the third case. This is a more recent topic that has emerged in the context of building knowledge bases from open domains like the web.

4.1 Predicting the Relationship Between a Given Entity Pair

Given a fixed set R of relationship types each involving a pair of entity types, our goal is to identify all occurrences of the relationships in R

in an input natural language document where all entities have been marked. Typically, in relationship extraction from natural language text it is assumed that the two argument entities are within a close proximity of each other, or are part of the same sentence. So, the basic recognition problem is as follows: given a text snippet $\mathbf{x}$ and two marked entities E_1 and E_2 in $\mathbf{x}$, identify if there is any of the relationships $\mathcal{Y}$ between E_1 and E_2. The set $\mathcal{Y}$ indicates all relationship types R and a special member "other" for the case when none of the relationships applies to this entity pair.

This prediction problem is simpler than the one that arises in entity extraction because here we only need to make a scalar prediction, instead of a vector of predictions. However, relationship extraction is considered a harder problem than entity extraction because relating two entity words in a sentence requires a skillful combination of local and nonlocal noisy clues from diverse syntactic and semantic structures in a sentence. We review the most common types of resources that are useful for relationship extractions.

Surface Tokens: The tokens around and in-between the two entities often hold strong clues for relationship extraction. For example, a "is_situated" relationship between a Company entity and a Location entity is strongly indicated by the presence of unigram token "located" and bigram token "located in" between the two entities, as in

> ⟨Company⟩ Kosmix ⟨/Company⟩ is located in the ⟨Location⟩ Bay area ⟨/Location⟩.

Similarly, an "outbreak" relationship between a disease and location is strongly indicated by the presence of words like "epidemic".

> The Centers for Disease Control and Prevention, which is in the front line of the world's response to the deadly ⟨Disease⟩ Ebola ⟨/Disease⟩ epidemic in ⟨Location⟩ Zaire ⟨/Location⟩.

Often, a token is generalized or stemmed to its morphological root. For example, "located" is stemmed to "locate."

Part of Speech Tags: Part of speech (POS) tags play a more central role in relationship extraction than in entity extraction. Verbs in a sentence are key to defining the relationship between entities, that are typically nouns or noun phrases. For example, in the sentence,

> ⟨Location⟩ The University of Helsinki ⟨/Location⟩ hosts ⟨Conference⟩ ICML ⟨/Conference⟩ this year.

more reliable extraction of the "held in" relationship between conferences and locations is possible if the word "hosts" is tagged as a verb (VBZ) instead of a noun as shown below.

> The/DT University/NNP of/IN Helsinki/NNP hosts/VBZ ICML/NNP this/DT year/NN.

Syntactic Parse Tree Structure: A parse tree groups words in a sentence into prominent phrase types such as noun phrases, prepositional phrases and verb phrases, and thus is significantly more valuable than POS tags in understanding the relationship between the entities in a sentence. For example, in the sentence

> ⟨Location⟩ Haifa ⟨/Location⟩, located 53 miles from ⟨Location⟩ Tel Aviv ⟨/Location⟩ will host ⟨Conference⟩ ICML ⟨/Conference⟩ in 2010.

"Tel Aviv, ICML" is likely to be preferred over "Haifa, ICML" as an instance of the "held in" relationship based on its relative proximity to ICML. But consider the parse tree of the sentence in Figure 4.1. This tree brings "ICML" closer to "Haifa" than "Tel Aviv" because "Haifa" is the head of the noun phrase "Haifa, located 53 miles from Tel Aviv" which forms the subject of the verb phrase "will host ICML in 2010."

Dependency Graph: Full parse trees are expensive to create. A dependency graph that links each word to the words that depend on it, is often found to be just as adequate as a parse tree. For example, for the sentence above the dependency graph is shown in Figure 4.2. In the graph it is clear that the verb "host" is linked to by both

```
(ROOT
  (S
    (NP
      (NP (NNP Haifa))
      (VP (VBN located)
        (PP
          (NP (CD 53) (NNS miles))
          (IN from)
          (NP (NNP Tel) (NNP Aviv)))))
    (VP (MD will)
      (VP (VB host)
        (NP
          (NP (NNP ICML))
          (PP (IN in)
            (NP (CD 2010)))))))
```

Fig. 4.1 Parse tree of a sentence.

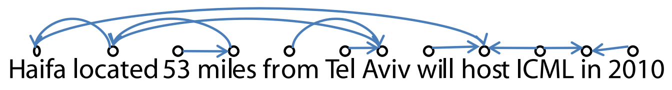

Fig. 4.2 Dependency parse of a sentence.

"Haifa" a location entity and "ICML" a conference entity. This directly establishes a close connection between them. In contrast, the path between ICML and Tel Aviv goes through "Haifa" and "Located."

We next present methods that use the above clues in different ways for classifying an input $(\mathbf{x}, E_1, E_2)$ into one of the $\mathcal{Y}$ classes. Assume we are given N training examples of the form $(\mathbf{x}^i, E_1^i, E_2^i, r^i) : i = 1 \cdots N$, where $r^i \in \mathcal{Y}$ denotes the relationship that exists between entities E_1^i and E_2^i in the sentence $\mathbf{x}^i$.

The main challenge is handling the diversity of structural forms that the different inputs represent. For example, the tokens and part of speech tags form a sequence, the parse information is a tree, and the dependency structure is a graph. Further, there could be errors in any of the input clues since the linguistic libraries used for these tasks are not perfect. While redundancy provides robustness to errors, too much redundancy brings in noise and increases running time.

The methods used for relationship extraction can be categorized into one of three main types:

- Feature-based methods that extract a flat set of features from the input and then invoke an off the shelf classifier like a decision tree or a SVM. (Section 4.1.1)
- Kernel-based methods that design special kernels to capture the similarity between structures such as trees and graphs. (Section 4.1.2)
- Rule-based methods that create propositional and first order rules over structures around the two entities. We refer the reader to [7, 147, 113, 190] for examples of various rule-based relationship extraction systems. Many of the issues in using rule-based system for relationship extraction are similar to the ones that arise in entity extraction covered in Section 2.

4.1.1 Feature-based Methods

A diverse set of methods have been used to convert the extraction clues in structures such as sequences, trees, and graphs to a flat set of features for use by conventional classifiers. Jiang and Zhai in [114] present a systematic method of designing such features, while also capturing most of the earlier proposed methods of feature-based relationship extraction [115, 196].

Let $\mathbf{x}$ denote the input sentence where x_i denotes the word at position i and E_1, E_2 denote the segments in $\mathbf{x}$ corresponding to the two entities whose relationship we wish to predict. For simplicity assume that E_1 and E_2 consist of single words each. Each word x_i is associated with a set of properties $p_1 \cdots p_k$ much like the features used in entity extraction. Examples of such properties include the string form of x_i, the orthographic type of x_i, the class of x_i in a given ontology, entity label of x_i, and the POS tag of x_i.

The first set of features are obtained by taking all possible conjunctions of the properties of the two tokens representing the two entities E_1 and E_2. Examples of such features are:

> ⟦Entity label of E_1 = Person, Entity label of E_2 = Location⟧.
> ⟦String E_1 = "Einstein" and Orthography type of E_2 = 4-digits⟧.

The first feature could be useful for the "resides in" relationship when the first entity in the sentence is a Person and the second marked entity is a Location. The second feature that could be useful for the "born" relationship when the string at E_1 is "Einstein" and E_2 consists of four digits.

Next, we present how to extract features from the structural inputs that captures the relationships among the words in the sentence. We have three kinds of inputs: sequences, parse trees, and dependency graphs. To unify the feature generation step from these inputs, we view each of them as a graph where the nodes are words, and in the case of parse trees also the nonterminals such as NNP, VP, etc. Each word node is associated with the set of k properties $p_1, \ldots, p_k$. In addition there is a special flag attached to each node that takes four possible values: 1 when it subsumes E_1, 2 when it subsumes E_2, "both" when it subsumes both, and "none" when it subsumes neither. Features are derived from properties of individual nodes, pairs of nodes connected by an edge, or triplets of nodes connected by at least two edges, and for each combination of values of the flag attached to the nodes. We explain this with examples from the sentence $\mathbf{x}$ with E_1 = Haifa and E_2 = ICML.

> ⟨Location⟩ Haifa ⟨/Location⟩, located 53 miles from ⟨Location⟩ Tel Aviv ⟨/Location⟩ will host ⟨Conference⟩ ICML ⟨/Conference⟩ in 2010.

Features from Word Sequence: We first give examples of features for the sequence graph formed out of the ten words between E_1 and E_2. In this case, the only edges are between adjacent words. Each node has k properties and one of three possible flag values (1, 2, none). Example unigram features are

> ⟦String =“host”, flag =“none”⟧
> ⟦Part of speech = Verb, flag = “none”⟧.

Example bigram features are:

> ⟦Strings = “(host, ICML)”, flags = “(none,2)”, type = “sequence”⟧
> ⟦Part of speech = (Verb,Noun) flag = “(none,2)”, type = “sequence”⟧
> ⟦(String = “host”, Part of speech = Noun), flag = “(none,2)”, type = “sequence”⟧.

Example trigram features are:

> ⟦Strings = “(will, host, ICML)”, flags = “(none,none,2)”, type = “sequence”⟧
> ⟦Part of speech = (Modifier, Verb,Noun) flag = “(none,none,2)”, type = “sequence”⟧.

Using this template we can easily calculate the maximum number of features of each type. Let $d(p_i)$ denote the number of possible values that property i can take and let $d = \sum_{i=1}^{k} d(p_i)$ denote the sum of the sizes of all properties. Then the number of unigram features is $3d$, the bigram features is 3^2d^2, and the trigram features is 3^3d^3. In practice, the number of features is much smaller because during training only the combination of properties present in at least one training instance are included.

Features from Dependency Graph: Next, consider the dependency graph in Figure 4.2. Since the set of nodes is the same as in sequences, we do not need to generate any more unigram features. The edges in the graph gives rise to many new bigram and trigram features. Here are some examples:

> ⟦(Entity label = Location, Part of speech = Verb), flag = “(1, none)”, type = “dependency”⟧.

This feature fires on node pair "(Haifa, host)" connected as "host $\leftarrow$ Haifa" in the dependency graph. A useful trigram feature that we get out of the dependency graph is

$$[\![(\text{POS} = (\text{noun,verb,noun}), \text{flag} = \text{“(1,none,2)”}, \text{type} = \text{“dependency”}]\!].$$

This feature fires on the nodes "(Haifa, host, ICML)" because of the edge pattern: "Haifa $\rightarrow$ host $\leftarrow$ ICML."

Features from Parse Trees: The node set in parse trees consists of the word nodes at the leaf and the internal nodes. This gives rise to new unigram features of the form:

$$[\![\text{Node} = \text{“VP”}, \text{flag} = \text{“2”}]\!].$$

Also, some of the internal nodes that subsume both the entity nodes E_1 and E_2 can now have a flag value of "both". For example, in Figure 4.1, the "S" node subsumes both the entities and is associated with a flag of "both." In addition, using the parent to child edges in the parse tree we can define bigram and trigram features as in the case of dependency graphs.

Jiang and Zhai [114] report that on eight relationship extraction tasks on the ACE corpus, the accuracy obtained with these simple features derived from unigram, bigram, and trigram units of input graphs is competitive with other methods that interpret the structure more globally.

4.1.2 Kernel-based Methods

Kernels provide a natural alternative to classification using graphs because then, instead of worrying about how to convert each graph into a flat set of features, one just needs to encode the similarity between the two graphs as a kernel function. Feature-based methods give rise to a very high dimensional decision space, whereas a suitable kernel in conjunction with a sparse trainer like SVM can significantly reduce the effective dimensionality of the space. All that is needed is to define over pairs of instances, a Kernel function $K(X, X')$ that roughly captures the similarity between two structures X and X'. Then, any method like a Support Vector Machine (SVM) classifier can be used to predict the relationship type as follows:

An SVM associates each training instance i and relationship type r with a weight α_{ir}. Recall our training data is of the form $(\mathbf{x}^i, E_1^i, E_2^i, r^i) : i = 1,\ldots,N$. We use X_i to denote $(\mathbf{x}^i, E_1^i, E_2^i)$. Given a new instance $X = (\mathbf{x}, E_1, E_2)$ the predicted relationship type $\hat{r}$ is defined as:

$$\hat{r} = \operatorname*{argmax}_{r \in \mathcal{Y}} \sum_{i=1}^{N} \alpha_{ir} K(X_i, X).$$

The values of α_{ir} are estimated during training. Details of training are not relevant to our discussion here and can be found in [208]. We discuss instead how to define meaningful kernel functions over the various kinds of structural inputs to a relationship extraction task.

Many kernel functions that apply either to parse trees or the dependency graph [33, 69, 215, 223, 224, 225] or a composition of the two have been proposed. Of these, kernels over dependency graphs are most popular. We describe a shortest path based kernel on dependency graphs that has been proposed in [33]. Let T and T' represent the dependency trees of two different training instances $X = (\mathbf{x}, E_1, E_2)$ and $X' = (\mathbf{x}', E_1', E_2')$, respectively. The kernel function $K(X, X')$ is defined as follows. Let the unique shortest path connecting the entities (E_1, E_2) in T be P and in T' be P'. Along each node of the path we have associated a set of properties of type $p_1,\ldots,p_k$ as described in the previous section on feature-based methods. Two nodes are considered similar if the value of many of these k properties are common. The node similarities are used to define the kernel function as follows:

$$K(P, P') = \begin{cases} 0 & \text{if } P, P' \text{ have different lengths} \\ \lambda \prod_{k=1}^{|P|} \text{CommonProperties}(P_k, P_k') & \text{otherwise,} \end{cases} \tag{4.1}$$

where CommonProperties(P_k, P_k') measures the number of properties common between the kth node along the paths P and P', respectively. Thus, the kernel value is high when the length of the shortest path between the two entities is the same in both sentences and the nodes along the path share many common properties.

A shortcoming of the above kernel is that it assigns zero similarity to paths of different lengths. A more thorough method of capturing the similarity between two paths is to use convolution kernels, defined originally for strings but later generalized to trees. A follow-up work in [224] reports slightly better performance with using convolution kernels on parse trees. Kernels have also been proposed to combine the similarity along different kinds of input clues including sequences, parse trees, and dependency graphs [224].

4.2 Extracting Entity Pairs Given a Relationship Type

In the previous section, our task was predicting the type of relationship between two marked entities in an input text. An alternative scenario is where we are given one or more relationship types, and our goal is to find all occurrences of those relationships in a corpus [2, 4, 14, 30, 182, 192, 211]. Most work in this area has been on open document collections such as the web, where one cannot assume that the entities are already marked. Typically, there is no labeled unstructured data for training unlike what we assumed in the previous section. Instead, the seeding of these systems is done by specifying for each relationship type r the following kinds of inputs

- The types of the entity pair that form the argument of r. The type of the entity is often specified at a high level, for example, if the entity argument is a proper or a common noun or a numeric such as year or currency. More specific types like “Person name” and “Company names” make sense only if they are accompanied by patterns that be used to recognize them in the unstructured text.
- A seed database S of pairs of entities that exhibit the relationships. In rare cases, negative examples of entity pairs that do not satisfy the relationship are also given.
- Sometimes, a seed set of manually coded patterns are also available. This is particularly easy for generic relationship types such as hypernym relationship (Parrot is a bird), or part-of or meronym relationship (steering wheel is a part of a car). For example, for extracting hypernym relationships,

Hearst et al. [105], propose patterns of the form: "⟨Noun⟩ such as ⟨List of Noun phrases⟩."

There are three steps to solving the problem of relationship extraction starting from the above input. We are given a corpus D, set of relationship types $r_1,\ldots,r_k$, entity types T_{r1},T_{r2} forming arguments of relationship type r, and a seed set S of examples of the form (E_{i1},E_{i2},r_i) $1 \leq i \leq N$ indicating that E_{i1} has relationship r_i with E_{i2}. Optionally, some of these might be marked as negative examples. An example of the seed input for two relationship types: "IsPhDAdvisorOf" and "Acquired" appears below. The entity type arguments for the first relationship is "(Person, Person)" and the second is "(Company, Company)".

E_1	E_2	r	Polarity
Donald Knuth	Andrei Broder	IsPhDAdvisorOf	+
Alon Halevy	Anhai Doan	IsPhDAdvisorOf	+
Jeff Ullman	Surajit Chaudhari	IsPhDAdvisorOf	+
Alon Halevy	Dan Suciu	IsPhDAdvisorOf	−
Google	You Tube	Acquired	+
Google	Yahoo!	Acquired	−
Microsoft	Powerset	Acquired	+

The first step is to use the given seed database S of entity pairs to learn extraction patterns M. The second step is to use a subset of the patterns to create from the unstructured corpus D candidate triplets (E_i,E_j,r_{ij}) indicating that entity E_i has relationship r_{ij} with entity E_j. A final validation step is to select only a subset of these candidates as true entity-relationship triplets based on additional statistical tests. We elaborate on each of these steps next.

4.2.1 Learn Patterns from Seed Triples

For each relationship type r we are given several entity pairs (E_1,E_2) indicating that E_1 and E_2 are related by relationship r. Optionally, an example might be marked as being a negative example for that triplet. There is an implicit assumption in this task that a given entity pair cannot be in more than one relationship with each other. Therefore, all entity pairs of a relationship r can serve as negative examples for relationship of a different type r'. A special case is when there is only one relationship type and there are no negative examples for that type. We do not consider this case and refer the reader to [211] for a customized method of extracting high frequency patterns using only positive seed examples of one relationship type.

The learning of patterns proceeds in three steps that we elaborate next.

Query Corpus to Find Sentences Containing an Entity Pair: The first step in learning the patterns, is to query the corpus to find text snippets containing both the entities. Typically, a query of the form "E_1 NEAR E_2" is posed for each example triple of the form (E_1, E_2, r). For example, the seed database above would give rise to seven queries of the form "Donald Knuth NEAR Andrei Broder", "Alon Halevy NEAR Anhai Doan", and so on. The NEAR predicate can be made more specific so as to retrieve only documents where the distance is less than a user-provided threshold [30, 211]. Most systems only consider text snippets that form a complete sentence. This yields for each entity pair (E_1, E_2) a bag of sentences $s_1, s_2, \ldots, s_n$ such that both entities appear in the sentence. Lightweight linguistic processing on the sentence is used to filter away sentences where E_1 and E_2 do not match the stated entity types T_{r1}, T_{r2} of relationship r.

Filter Sentences Supporting the Relationship: Not all sentences containing the entity pair necessarily support the relationship type. For example, if we start with a seed triple (Donald Knuth, Andrei Broder, isPhDAdvisorOf) and get two sentences:

> Broder completed his doctoral thesis under the supervision of Donald Knuth
> The invited speakers for the conference are A. Broder, R. Karp, and D. Knuth.

then, only the first sentence is a positive example of the relationship. Automatic filtering of positive sentences is challenging because there is no labeled data to go by. Banko et al. [14] propose a simple heuristic for filtering based on dependency trees. They propose to first perform a dependency parse of the sentence and retain only those sentences as positive examples where the length of the dependency links between the occurrences of E_1 and E_2 in the sentence is no more than a certain length. This will not always work. In the two sentences above, the “and” in the second sentence makes “Knuth” directly connected to “Broder”.

Learn Patterns from Entity Pairs and Sentences: Now the learning problem reduces to that of Section 4.1 where we have labeled data in the form of a set of sentences with marked entity pairs and a named relationship type between the pair. We can then formulate it as a standard classification problem where given an entity pair marked in a sentence, we need to predict the relationship between the two entities. There are two practical problems to training a model by treating each sentence as an independent training example.

First, in spite of the filtering there is no guarantee that all sentences are indeed positive examples for the relationship of interest. To address this problem, Bunescu and Mooney [30] explored the possibility of casting this as a multi-instance learning problem. In multi-instance learning, each positive example is associated with a bag of instances where at least one of them is guaranteed to be positive but it is not known which one. However, they obtained better results by treating each sentence as an independent training instance and invoking a high performing classifier such as an SVM.

A second difference from the setup in Section 4.1 is that typically for each entity pair, say (Alon Halevy, Anhai Doan) many sentences will contain that pair of entities. When the set of seed pairs in the database is small, the classifier might include features that are specific to particular entity pairs. For example, words like “Schema”, “Integration” might surface as significant features because they appear in many positive sentences corresponding to Alon Halevy, Anhai Doan, isPhDAdvisorOf. But, these features do not generalize for the “isPhDAdvisorOf” relationship prediction task. Bunescu and Mooney [30] propose to solve this problem by down-weighting terms that have a strong correlation with the words in the entity pair.

4.2.2 Extract Candidate Entity Pairs

Now we get to the actual task of finding all entity pairs in the corpus D that support the given relationship types. Let $\mathcal{M}$ denote the relationship extraction model learnt in the previous section. This model takes as input a sentence $\mathbf{x}$ and two entities in it E_1 and E_2 and predicts the relationship type between E_1 and E_2. So, a straightforward method to find all entities in D is to make a sequential pass over D, for each sentence find if it contains two entities of types T_{r1} and T_{r2} for each relationship type r, and if yes, invoke model $\mathcal{M}$ to get the right prediction. This approach would work for small to medium sized collections such as Wikipedia, or if an enterprise like a large search engine chooses to integrate relationship extraction as a part of its processing pipeline of during offline crawling.

When a sequential scan is not feasible but an index on the corpus exists, it makes sense to retrieve only relevant sentences based on keyword search patterns, and apply $\mathcal{M}$ on the fetched subset. The problem of finding the sequence of queries so as to provide the right tradeoff of cost, recall, and precision raises interesting research challenges particularly in webscale corpora with all its attendant noise and redundancy. Existing methods are of two types.

Pattern-based: The first type depend on a set of manually coded phrase patterns that match sentences which contain the relationships. For example, Is–a–E relationships where E is an entity type are extracted by phrase searches of the form “E such as”, “or other E”, “such E as”, “E like” and so on [86, 105, 211]. Hearst et al. [105] first proposed patterns of the form: “⟨Noun⟩ such as ⟨List of Noun phrases⟩.” These patterns have since been enriched and deployed in projects like KnowITAll project [86] for extracting instances of a given type from the web. For part-of relationships, patterns like “Y’s X” and “X of the Y” have been proposed in [18]. Similarly, it is easy for a human expert to provide patterns of the form “PhD under supervision”, “PhD NEAR Advisor” for collecting instances of “isPhDAdvisorOf” relationship.

Keyword-based: The second type depend on keyword matches to perform subject-level filtering of documents. For example, filters on keywords “vaccine” and “cure” are used to subset documents containing disease outbreaks in [99]. This raises questions about selecting the right sequence of words to ensure that the result of each successive query provides the largest set of relevant documents containing entities not already present in the database. Agichtein and Gravano [2, 5] give an iterative solution to the problem. The basic premise of the method is that documents that contain relevant entities are implicitly linked by shared keywords. Therefore, by starting from a seed set of entities, it is possible to fetch, using keyword searches alone, all documents that contain relevant relationships. The problem is broadly applicable even for generic information retrieval queries. Flake et al. [94] and others in the IR community have addressed the problem of generating a sequence of keywords based on a user’s feedback on relevant and irrelevant documents. The main difference with the IR setting is in the notion of relevance of a document. In extraction, relevance is defined based on whether an entity was found in a newly fetched page. In IR, relevance is judged by a user providing feedback.

4.2.3 Validate Extracted Relationships

The lack of proper training data often results in high error rates in extractions of the previous step. Many systems deploy an additional validation phase that depends on more expensive aggregate statistics on the entire corpus to prune away candidates with low support. For large corpora such as the web, the same relationship instance will be extracted from many different sentences, typically coming from many independent sources. The number of occurrences of a relationship can therefore be used to derive a probability that the given extraction is correct as proposed in [82]. We elaborate on how such probabilities are calculated in Section 5.4.3. Even when a particular relationship instance is rare, the contextual pattern in which it occurs is likely to be frequent and these can be exploited to extract only high confidence relationships [83]. An alternative method is proposed in [182] based on the observation that a major cause of error in relationship extraction is errors in marking the boundaries of entities. They correct this error under the premise that the entity with the correct boundary will be significantly more frequent than either its subset or superset.

Summary

In this section, we covered the problem of relationship extraction under two settings. The first setting is where we needed to classify the types of relationships that exist between entities that are already marked in an unstructured text. We presented feature-based and kernel-based methods for combining clues from diverse structures such as the word sequence, parse tree, and dependency graph for the classification task. For this part, we assumed that the training data consists of a document collection with labeled relationship types between entity pairs. The second setting is where we needed to find entity pairs in a corpus that exhibit a given set of relationship types. In this setting, we do not have labeled unstructured data, just a seed database of entity pairs exhibiting the relationships. We discussed a number of techniques for creating a relationship extraction model by bootstrapping from such seed databases. We then discussed how a pre-existing index can be used to filter only relevant subsets likely to exhibit the relationship.

In spite of the extensive research on the topic, relationship extraction is by no means a solved problem. The accuracy values still range in the neighborhood of 50%–70% even in closed benchmark datasets such as ACE. In open domains like the web the state-of-the-art systems still involve a lot of special case handling that cannot easily be described as principled, portable approaches.

Further Readings

In this section, we concentrated on the extraction of binary relationship. A natural extension is to the extraction of records involving multi-way relationships. Record extraction is significantly more complicated since binary relations are assumed to be at sentence level, whereas multi-way relationships span sentences. On natural language text this makes it necessary to perform cross sentence analysis involving co-reference resolution and discourse analysis. We refer the reader to [148, 218] for existing work on this topic.

We assumed here that the set of relationship types is known in advance. As extractions systems start getting deployed in more open-ended settings, for example, the web and Wikipedia, it is not possible to define the set of relationships in advance. Instead, a part of the extraction process is automatically discovering relevant relationship types. Early work on automatic relationship discovery appears in [14, 39, 192, 211].

5

Management of Information Extraction Systems

In the previous sections we presented a number of different models and algorithms for various kinds of extraction tasks. Real-life deployment of these extraction techniques in the context of an operational system raises many practical engineering, performance, and usability issues: How scalable are these techniques? How to integrate the structures extracted with existing data? Since extractions are not guaranteed to be error-free, how does the storage and querying model adapt to erroneous sources? As the unstructured source evolves, how does the extraction model detect the change and adapt to it?

We devote this section to addressing these issues. We first discuss performance-related issues in Section 5.1 which includes topics like the design and use of indices to filter relevant unstructured sources, efficient lookup of large structured databases during entity extraction and optimization of extraction pipelines.

Next, in Section 5.2 we discuss issues related to handling dynamically changing unstructured sources. We discuss techniques for optimizing the incremental process of extraction, detecting when data drift causes extractors to fail, and repairing extractors to work with the modified data.

In Section 5.3, we discuss issues related to the integration of extracted entities with existing entities in the database and with repeated occurrences of that entity in the unstructured source. This is a challenging problem that has been extensively researched under various names like deduplication, coreference resolution, record linkage and so on.

In Section 5.4, we review techniques for managing errors that arise in the extraction process. There is much interest in the management of imprecise data, however the imprecision of information extraction models raises its own set of challenges. It is not obvious how to assign numerical confidence values with each extraction and choose an efficient imprecise data model for storing extraction results.

5.1 Performance Optimization

While many extraction systems have been in operation both commercially and as research prototypes, published research addressing the various performance aspects of extraction is only starting to appear.

Extraction systems can be deployed in two modes. In one mode the unstructured source is naturally available, for example in closed systems like a data warehouse or a complaint processing center. In the second mode, the unstructured source is open-ended and large, like the web, and a part of the extraction process is also selecting the relevant subset of documents. Typically, in the first mode the user is interested in annotating all occurrences of the entities/relationships in the unstructured source, whereas in the second mode, the user's interest is in creating a repository of structured entities. Therefore, only documents that are likely to contain new entities need to be processed. We consider optimizations that have been proposed for making such selections efficient in Section 5.1.1.

This is followed by a phase of within document filtering where we zoom to the right sections of the document. For example, in a citation system it is necessary to design quick tests to locate the paper headers and reference sections of papers. Existing solutions here are domain-specific, and we do not have anything general-purpose to discuss.

Finally, on selected subset of documents the extractors are deployed. Most extraction algorithms scale linearly with the length of the input. Even so, there is a need to optimize their performance because the pre-processing and feature generation steps tend to be expensive. Existing pattern-based extraction systems are almost always CPU-bound; I/O figures only when finding matches to existing database of entities. Both rule-based and statistical methods depend on features of the text for recognizing entities and relationships. Features have varying costs of evaluations. Some of these, for example if a word is capitalized, are cheap to evaluate, whereas others, for example that check whether a span has a large match to a database of entities, are expensive. In Section 5.1.2, we present techniques for efficiently evaluating such expensive matches on large databases. In Section 5.1.3, we present optimizations that apply during extractions that depend on a mixture of cheap and expensive features.

5.1.1 Document Selection Strategies

When the source is really large, there is no alternative to manually restricting the set through a list of addresses or address domains. For example, DBLife [78] uses a listing of URLs pointing to database researchers' homepages, conference web sites, and mailing lists like DBWorld. When such manual subsetting is not possible, there is a need for additional levels of pruning and two solutions have been proposed. The first one, applicable only for hyperlinked sources is to perform some form of focused crawling [47]. The second option is exploiting pre-existing indices on the unstructured source to fetch only the documents of interest. This raises issues on how to search the index and how to design appropriate indices for extraction that we discuss in the later part of this section. Even after a document is fetched, the cost of running the full-fledged extraction algorithm on the entire document is often expensive. It might be possible to design more sophisticated tests, such as a statistical whole-document classifier as a second-level filter of relevance [2].

In general, there is an interesting trade off between recall and time among these three options of selecting documents: focused crawling, searching via keywords, and filtering documents after fetching them using a classifier. This is captured as an optimization problem in [110]. One of the contributions of the work is providing a detailed estimation procedure for recall and running time of various options.

Index Search Techniques: Depending on the richness of the index, queries can be of two types: standard IR-style keyword queries and pattern queries for entity-level filtering.

The keyword mode of access is typically used to perform only a crude subject-level filtering of documents. For example, filters on keywords "vaccine" and "cure" are used to subset documents containing disease outbreaks in [99]. More discussion of this topic appears in the context of relationship extraction in Section 4.2.2.

Pattern queries offer a finer grained filtering of only the entities of interest from the corpus. As we saw in earlier sections, entity recognition is more often driven by patterns capturing orthographic properties of text than keywords. For example, to retrieve mentions of person names in an indexed document collection, it is not possible to depend on keyword queries alone. Regular expression queries such as "[Mr.| Dr. | Mrs.] Initial Dot Capitalized_Word" could be more useful. Similarly, the pattern query "Thomas w+ Edison" will be more useful to extract the middle name of Thomas Edison instead of the IR query "Thomas NEAR Edison." Specialized indices and search algorithms are needed to support pattern-level queries, particularly if they are to support character-level pattern queries such as "<a href=.*ṁp3>".

Some strategies for searching an inverted index with regular expressions are proposed in [58, 172]. An interesting challenge is deploying an index to retrieve entities for statistical models. An example of such an index for extracting entity names using patterns involving various features like part of speech tags, or orthographic patterns appears in [37].

Index Design for Efficient Extraction: The use of indices in filtering for information extraction, raises many engineering questions, which have not been sufficiently addressed. In general, the index should provide efficient support for proximity queries, regular expression patterns, and allow efficient storage of tags like POS, phrase tags, common entity tags like person and company names, and tags from standard ontologies such as WordNet [180]. Cafarella and Etzioni [38], for example, suggest that the standard inverted index for keyword searches be augmented to neighborhood tag information along with each (document, position) entry in an inverted list. While this makes it efficient to answer queries of the form "cities such as NP", the storage overhead is huge. Chakrabarti et al. [48] discuss how to efficiently search an index where the query contains tags coming from a hierarchy. It also addresses workload-driven strategies for selecting the subset of tags in a hierarchy that should be directly indexed, instead of being computed at query time by ORing inverted lists of tags below it.

In order to efficiently support queries involving regular expressions at the level of characters, it will be necessary to go beyond existing word-level inverted indices. Two options are suffix trees and q-gram indices. Suffix trees are the classical solution for pattern search queries but they are too space intensive, and are not a scalable option for large amounts of data. Inverted indices are the *de facto* choice for large scale keyword searches. One option to support character-level searches on inverted indices it to index q-grams, and possibly variable length q-grams as explored in [117].

5.1.2 Efficient Querying of Entity Databases for Extraction

As we discussed in Sections 2 and 3, a valuable feature to an entity extraction system is the degree of similarity of a candidate entity to an existing entity database. For example, for extracting book titles from Blogs, a very strong clue is provided by a close match with a book title in a database of published books. However, in order to be able to use these features we have to find the similarity of each possible text span in the unstructured source to the database of entities. This is an expensive operation that needs to be optimized separately from other generic pattern matching operations [49, 179, 190]. Since extraction decisions depend on a diverse set of patterns, only one of which could be the degree of match with an existing database, in general one needs to find match values for each possible span in an input document. Also, it is meaningless to look for exact match between an input unstructured source and an entity database given that slight mismatch in forms is inevitable in real-data.

We formalize the problem as follows: We are given an input token sequence $\mathbf{x}$ and a large database of entities D. Our goal is to find each possible segment in $\mathbf{x}$ whose similarity to an entry in D is greater than a threshold ϵ. We call this the Batch-Top-K search problem. We concentrate on the TF–IDF similarity score, which has been found to be highly effective in text searches. The TF–IDF similarity between two text records r_1 and r_2 is defined as follows:

$$\begin{aligned} \text{TF–IDF}(r_1,r_2) &= \sum_{t \in r_1 \cap r_2} V(t,r_1)V(t,r_2) \\ V(t,r) &= \frac{V'(t,r)}{\sum_{t' \in r} V'(t',r)^2} \\ V'(t,r) &= \log(\text{TF}(t,r)+1)\log(\text{IDF}(t)). \end{aligned} \tag{5.1}$$

In the above, the IDF term makes the weight of a token inversely proportional to its frequency in the database and the TF term makes it proportional to its frequency in the record. Intuitively, this assigns low scores to frequent tokens (stop-tokens) and high scores to rare tokens.

The basic Top-K search problem with the TF–IDF similarity measure is extensively researched [51, 88, 204, 219] in the information retrieval and database literature. Thus, a simple mechanism of solving our Batch-Top-K problem is to invoke the basic Top-K algorithm for each segment in the input sequence. However, since segments have overlapping tokens there is scope for significant gains by batching their computations. The state-of-the-art Top-K algorithms are highly optimized to balance the number of tidlist[1] fetches with record fetches via lower and upper bound score estimates. Utilizing these optimizations along with effectively sharing the partial and final results of one sub-query in the evaluation of another is a challenging problem. Chandel et al. [49] present an algorithm for optimizing such evaluations and report significant runtime reductions by designing special purpose index lookups for entity extraction tasks.

[1] Tidlist refers to list of tuple-ids containing a given token. This is analogous to document-id list in an inverted-word index used in IR applications.

5.1.3 Optimizations for Patterns of Variable Cost

Patterns that depend only on the surface properties of a text span are cheaper than patterns that require external resources like match with a database, or expensive operations like inferring the part of speech of tokens in the span. In rule-based systems where a rule contains a conjunction of such patterns, it is possible to use tricks from expensive predicate evaluation in relational systems to reorder predicates based on the selectivity and cost estimates of each pattern as suggested in [179, 190].

In statistical systems where hard predicates are missing, the steps required to minimize the evaluations of expensive patterns are more involved. Consider entity extraction based on a log-linear model where the score of assigning a particular entity label to a text span is a weighted sum of features of the span, a subset of which are expensive. Chandel et al. [49] show how to minimize the exact evaluations of the expensive features by exploiting cheap upper and lower bounds on the values of the expensive features.

In general, both for rule-based and statistical systems an interesting, unexplored problem is to design rules or models which explicitly account for the cost of evaluating features. The goal during training then becomes not just accuracy, but also cost minimization during model deployment. Many techniques exist for cost-based learning in the ML literature [80, 210, 221], but these have not been deployed on extraction models.

5.1.4 Optimization Frameworks: Relational Style

Recently, relational engine style frameworks have been proposed for optimizing the evaluation of rule-based systems for entity and relationship extraction [179, 190]. In rule-based systems, a typical extraction consists of the following operations:

- Predicates that can be evaluated independently on text spans.
- Predicates on text spans that depend on expensive database lookups.

- Predicates that involve two text spans. Typically, these are proximity predicates that control the gap between text spans that form different attributes of a single entity. For example, the title and abstract of a paper should not be separated by more than 10 words.

These operations can be cast as select and join predicates of varying costs operating over document streams. The order of evaluation of these predicates could have a big impact on running time. For example, the last operation could be viewed as a join over two streams of text spans: the first that extracts titles and the second that extracts abstracts. There are three options for evaluating it: find title spans first and then check if there is an abstract within 10 words to the right, find abstracts first and check if there is a title 10 words to the left, or independently find all title and abstract spans and join to retain pairs that satisfy the distance constraint. The selection of the right order can be cast as a relational plan selection problem when proper statistics of selectivity and running time is attached with each predicate as shown in [179, 190].

5.2 Handling Change

In an operational setting where extractors are deployed repeatedly on changing sources, there is a scope for improving performance by reusing work from previous extractions on the same page. We discuss the limited work that exists on this topic in Section 5.2.1. A related issue with dynamic data is detecting when an extraction model, trained or manually tuned, no longer works on newly evolved pages. This issue has been addressed in the context of generating wrappers for web pages [56, 122, 128, 176]. In Section 5.2.2, we discuss one representative work in this area [128].

5.2.1 Incremental Extractions on Changing Sources

An easy optimization, with clear scope for performance boost, is to run the extractor only on the changed portions of a page, instead of the entire page. The success of this optimization rests on being able to efficiently detect the regions of the page that have remained unchanged. This is a well-explored problem and there are standard tools like Unix

Diff for flat documents. Formatted documents like web pages which follow hierarchical layouts present somewhat different issues. These have been explored in the context of fragment-level caching in [174, 175]. Specially for extraction, Chen et al. [53] evaluate the option of detecting regions of change using Unix diff and suffix trees. Their results show that both these options are comparable in reducing running time beyond running the extractor afresh.

5.2.2 Detecting When Extractors Fail on Evolving Data

We discuss the method proposed in [128] on detecting when a set of pages have changed enough to cause an extractor to return incorrect results. The problem is challenging because we need to detect when the extractor fails even though the content of the source keeps changing. For example, consider an extractor for locating product names and price from an auctioning web site. The only change that we are interested in flagging is when the pattern used for extracting product names and price changes, not in any change of the content or layout of the page.

Let S be a set of documents on which an extractor E is known to produce correct structured data D. Let S' be the new version of the documents. S' might not contain necessarily the same set of pages as S, as some pages might have been deleted, and new pages might be added. Let D' be the data extracted by applying E on S'.

The first step is to derive from D a set of data characteristics $\mathcal{F}$ that capture certain key generalizable properties of the data. With the change from S to S', recalculate the characteristics $\mathcal{F}'$ of D'. The extractor E is declared to be wrong when $\mathcal{F}$ and $\mathcal{F}'$ differ drastically. We first describe how to calculate generalizable characteristics of a dataset so that we can detect if two datasets D and D' represent entities of the same type in spite of differences in their content. Next, we will describe how to quantify such differences.

Defining Characteristic Patterns: The main challenge in capturing a set of characteristics of D is that at the time we capture these characteristics we do not have any other reference dataset that could be used as a negative example. Most pattern learning algorithms require both positive and negative examples. In the absence of negative examples, one idea for ensuring generalizability is to ensure that the chosen data characteristics apply to a large number of instances in the dataset D. The algorithm, DataProg proposed in [128] captures data characteristics as a set of patterns where a pattern is selected only if its frequency is statistically significant vis-a-vis frequencies of subsets of the pattern. Another design decision in DataProg to avoid choosing very specific patterns is to define patterns over either the beginning or the end of the data records, instead of the entire record.

We repeat the toy example from [128] to illustrate the kind of patterns that DataProg finds. The dataset D is the list of the following five street addresses:

4676 Admiralty Way
10924 Pico Boulevard
512 Oak Street
2431 Main Street
5257 Adams Boulevard

and the two patterns that comprise the characterizing set $\mathcal{F}$ of D are:

P_1: Number UpperCaseWord Boulevard
P_2: Number UpperCaseWord Street

The patterns P_1 and P_2 jointly cover four of the five addresses above. We do not include a pattern such as "Number UpperCaseWord Way" because that would cover too few addresses in this set and possibly would not generalize.

The algorithm for finding such patterns follows a top-down pattern growth procedure and we refer the reader to the paper [128] for its details.

Detecting Significant Change: Let $P_1 \cdots P_m$ be m characteristic patterns discovered in the previous section. Let F_i be the total count of each pattern i in (D,S) and F'_i be the count in the modified version (D',S'). The distribution represented by F'_i is said to be statistically different from F_i, if the expected values e'_i of counts in (D',S') obtained by extrapolated from F_i, differs a lot from F'_i. The expected value $e'_i = F_i \frac{N'}{N}$, where N' is the size of S' and N is the size of S. The check of whether e'_i is different enough from F'_i is performed using the χ^2 statistic

$$\sum_{i=1}^{m} \frac{(F'_i - e'_i)^2}{e'_i}. \tag{5.2}$$

The above statistic is reliable only when the m patterns are independent of each other. It is observed in [128] that patterns derived from both the start and end parts of data performed worse than those derived from just the starting parts of data because of high correlation among the two pattern types. Also [128] proposes a number of other user-defined simple aggregate features derived from the pages S and S' instead of restricting to the extracted datasets D and D'.

5.3 Integration of Extracted Information

As entities and relationships are extracted from the unstructured source, they need to be integrated with existing databases and with repeated mentions of the same information in the unstructured source. The main challenge in this task is deciding if two strings refer to the same entity in spite of the many noisy variants in which it appears in the unstructured source. This problem is variously called deduplication, coreference resolution, record linkage and so on and has been extensively studied in many different communities [8, 21, 61, 64, 90, 145, 187, 217].

When data integration is performed on automatically extracted information, a number of design alternatives arise. Ideally, extraction of all repeated mentions should be done simultaneously and jointly with integration with existing sources. However, this increases complexity and running time. Existing methods of data integration in the context of extraction can be categorized under three scenarios depending on the sequence in which the different tasks are scheduled.

5.3.1 Decoupled Extractions and Integration

In this case each extraction decision is taken independently of any subsequent data integration steps and independent of the extraction of other repeated mentions. The task during integration is to decide if a given extracted record is the same as any of the existing database entries, or if it is a new record. This is a basic de-duplication problem that is solved by reducing to a binary classification problem as follows:

The classifier takes as input a pair of records and outputs a binary decision of whether the records in the pair are duplicates of each other or not. The input features to the classifier are string similarity functions such as cosine similarity, edit distance, Jaccard similarity, and Soundex. Some of these similarity functions could be domain dependent. For example, for comparing page numbers in scientific publications, a user could define a special purpose similarity function that makes "408–11" look similar to "pp. 408–411." The classifier could be either a set of manually defined rules on the similarity functions or an automatically trained classifier such as a SVM or a decision tree. An example of a decision tree created on similarity functions defined over various fields of citation records appears in Figure 5.1.

The integration of an extracted record r happens as follows. For each entry e in the existing database D, we apply the above classifier on the pair (r,e) and get a "yes/no" verdict on whether r is a duplicate of e. If the answer is no for all entries, r is inserted as a new entry. Otherwise, it is integrated with the best matching entry e. This sequential process can be sped up considerably through index lookups for efficiently finding

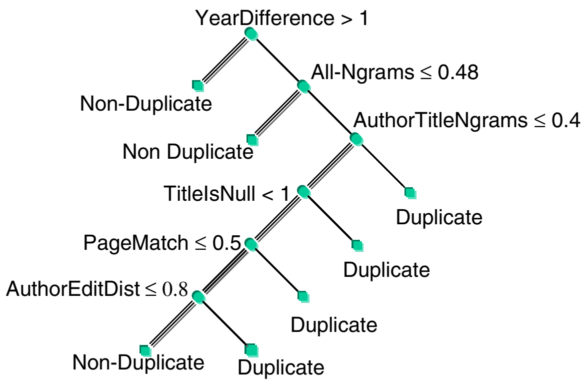

Fig. 5.1 An example decision tree for resolving if two citation records are duplicates of each other or not. Each internal node is a similarity function on one or more attributes of the record.

only likely matches. For example, an inverted index on the entries in the database D can be used to efficiently retrieve only those records that contain more than a threshold number of common words with the extracted record r. Word-based indices may not be adequate when spelling mistakes are common and the text records are short. In such cases, an index on the sliding window of q-grams in the text is more useful. In general, a hybrid of the two approaches might yield the best results as proposed in [51].

5.3.2 Decoupled Extraction and Collective Integration

Typically when the same entry is extracted from multiple locations, more accurate integration is possible if all mentions are collectively grouped into entities instead of integrating one entry at a time. As an example, consider three records extracted in the order R1, R2, R3.

R1. Alistair MacLean
R2. A Mclean
R3. Alistair Mclean

Before seeing R3, it is unlikely that R1 will be classified as a duplicate of R2 because the last names of R1 and R2 do not match exactly and the first name is abbreviated. However, after seeing R3, R2 and R1 start looking similar because of the transitivity of the duplicate-of relation. This insight is used to cast the collective integration of multiple records as a graph partitioning problem as follows:

The nodes of the graph are the records. An edge is drawn between record pairs e_i and e_j weighted with a score w_{ij}. The sign of the score denotes if the pair is likely to be a duplicate or a nonduplicate, and the magnitude denotes the confidence in this outcome. All pairs with no edges between them are assumed to be connected with a large negative weight. Such scores can be obtained through a number of means: hand tuned weighted combination of the similarity between the record pairs, log probability of the confidence value of a binary probabilistic classifier such as a logistic regression, scores from a binary SVM classifier, and so on.

In joint deduplication, the goal is to harden the scores to consistent 0/1 labels on all edges. A label assignment is consistent if it satisfies the transitivity of the is-duplicate-of relation. That is, if the output label on the edge between e_i, e_j is 1 and between e_j, e_k is 1, then the output on e_i, e_k also has to be 1. Such an output can also be thought of as a disjoint partitioning of the nodes of the graph such that each partition denotes a distinct entity. All the edges within a partition are ones and across partitions are zeros. Our goal is to generate a consistent edge label assignment that minimizes the scores of the positive edges across partitions and negative edges within partitions. This is called the Correlation Clustering (CC) problem in [15]. As an example, consider Figure 5.2 which shows a toy weighted graph, the inconsistent solution obtained by independently classifying each pair, and the optimal consistent solution. However, for general graphs finding such optimal solutions is NP-hard. Charikar et al. [50] provide a linear programming (LP) formulation of a relaxed version of the problem. Such LP based approaches are not likely to be practical because for each of the cubic triples (e_i, e_j, e_k) of records, the LP requires a constraint to enforce transitivity. A practical alternative is to depend on well-known clustering algorithms, such as bottom-up agglomerative, or top-down divisive where the criteria for each greedy step is modified to be the reduction of the correlation clustering objective as shown in [54].

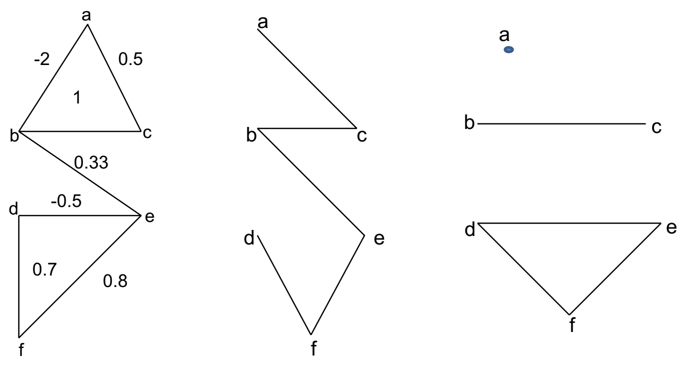

Fig. 5.2 An example of correlation clustering. The leftmost figure is the starting weighted graphs. The middle graph shows an inconsistent partitioning obtained by retaining only the positive edges. The rightmost graph is the optimal consistent partitioning.

Collective Multi-attribute Integration: When the information extracted spans multiple columns, instead of a single column, collective integration can have even a greater impact. Consider the four citation records below (from [161]).

Id	Title	Author	Venue
b1	Record Linkage using CRFs	Linda Stewart	KDD-2003
b2	Record Linkage using CRFs	Linda Stewart	9th SIGKDD
b3	Learning Boolean Formulas	Bill Johnson	KDD-2003
b4	Learning of Boolean Expressions	William Johnson	9th SIGKDD

The similarity between b1 and b2 could be easy to establish because of the high similarity of the title and author fields. This in turn forces the venues "KDD-2003", "9th SIGKDD" to be called duplicates even though intrinsic textual similarity is not too high. These same venue names are shared between b3 and b4 and now it might be easy to call b3 and b4 duplicates in spite of not such high textual similarity between the author and title fields. In fact, such an iterative scheme can be shown to provably converge to the optimal decisions about pairs being duplicates of each other. Alternately, the best pairwise labeling of attributes and record pairs being duplicates or not can be found efficiently using a minimum cut-based algorithm as discussed in [194]. However, the optimal pairwise decisions may not satisfy transitivity. So, a second step is needed to convert the pairwise decisions obtained here to consistent decisions that correspond to an entity-level partitioning of the data.

5.3.3 Coupled Extraction and Integration

When extraction is performed in the presence of a large database of existing entities, we already discussed how to improve extraction accuracy by adding features such as the maximum similarity of a text segment to the existing entities. When our goal is also to resolve which of the existing entities is the same as the entity extracted, a question that arises is if performing the two steps jointly instead of independently can improve accuracy. We believe that there is little to be gained out of joint extraction and integration when the database is not guaranteed to be complete, and when we are extracting single entities at a time. In contrast, when extracting records or multi-way relationships consisting of multiple entity subtypes, it is possible to boost accuracy by coupling the two steps. Consider, an example string:

"In his foreword to Transaction Processing Concepts and Techniques, Bruce Lindsay"

Suppose we have an existing Books database consisting of the following three tables:

- Book names where one of the entries is "Transaction Processing: Concepts and Techniques."
- People names consisting of entries like "A. Reuters", "J. Gray", "B. Lindsay", "D Knuth", and so on.
- Authors table linking the book titles with the people who wrote them.

If we try to independently extract and integrate, it is quite likely that the string "Transaction Processing: Concepts and Techniques" will be extracted as a book title because of the strong similarity to the entry in the booknames table. Similarly, "Bruce Lindsay" is likely to be extracted as the author because the string "Bruce Lindsay" matches an entry in the people names tables and because of its placement next to the string marked as book title. After the extraction, when we try to integrate the results with the database entries we will fail because of inconsistency with the existing author names in the database. In contrast, combined extraction and integration will avoid the error during extraction itself. However, modifying the inference algorithms used for extraction to handle such database imposed constraints raises new challenges. We refer the readers to [151, 217], and [138] for three different mechanisms in which such constraints are handled.

5.4 Imprecision of Extraction

Errors in extractions are inevitable, particularly in open domain unstructured sources where it is impossible to foresee all different patterns of noise. In many cases users of automatically extracted databases have come to live with erroneous outputs. For example, users of automatically curated citation databases such as the Google Scholar and Citeseer, are aware that a Bibtex entry on these sites is to be trusted less than that obtained from manually curated web sites such as the ACM Digital library. The automatically extracted databases still serve their role in terms of providing a more comprehensive, but by no means error-free, list of forward citations.

As the scope of extraction based systems expands to business critical situations, there is increasing research interest in more formally capturing the errors of extraction as an imprecise data model so that the answers of queries can be associated with correctness indicators. Many imprecise data models are being actively explored in the database research community. A top-level characterization is based on whether imprecision is represented quantitatively as a probability distribution, or not. In the probabilistic approach each possible world has a probability value associated with it, whereas in a nonprobabilistic approach there is a hard notion of whether a particular combination of values is possible. Such constraints are often provided as logical predicates attached to rows, columns, or entire tables in the database. We will consider the probabilistic approach for the rest of the discussion.

We concentrate on the problem of how to populate a probabilistic database so as to faithfully represent the uncertainty of extraction. We consider only simple models based on row and column uncertainties because the cost of query processing on more complicated probabilistic models could get prohibitively high. We give an overview of existing work under three different scenarios.

5.4.1 Confidence Values for Single Extractions

Consider the simplest setting where we are performing independent extractions of some structured entity, say list of Protein names, or book titles. We can represent the imprecision of this extraction in one of two possible ways. The first is to associate each extracted information with a probability value. The second method extends the first approach to output multiple possible extractions instead of a single extraction. For a given source string, the sum of the probability of the different possibilities is one. For example, the output of an extraction of book titles from sentences could look like this:

Id	Title	Pr
1	Last Theorem	0.5
1	Fermat's Last Theorem	0.3
1	The Fermat's Last Theorem	0.2
2	"Transaction Processing: Concepts and Techniques"	0.95
2	Transaction Processing	0.05

The above example shows that the first option of keeping only the highest scoring extraction would miss the second correct extraction from the first source.

Surprisingly, it is very difficult to get such probability estimates from typical extraction models. The reason is that the probability values convey more than just vague notions of correctness. There is a specific interpretation of the probability estimates with regard to what they tell us about the correctness of the results. Suppose, we have a ground truth database using which we can find out which of the extractions are correct and which are wrong. If the probability estimates are sound, then we expect that roughly p fraction of the entries marked with probabilities between $p - \epsilon$ and $p + \epsilon$ are correct. Many well-known entity recognition methods such as rules, decision trees, statistical classifiers such as SVMs, and naive Bayes classifiers, while adequate for outputting a single most preferred entity, fail badly when used to associate confidence values with their outputs.

A study that compared different classifiers on the goodness of the confidence scores that they output is reported in [158]. A useful visual tool to measure the soundness of the probabilities output by a classifier is a reliability plot. The X-axis of a reliability plot are binned probabilities output by a classifier and the Y-axis is the fraction of test instances in that probability bin whose predictions are correct. Two examples of reliability plots for extractions using a CRF-based extractor (Section 3.1.2) are shown in Figure 5.3. The closer the bin heights are to the 45 degree line, the better calibrated are the probability estimates of the classifier. The study concluded that popular methods such as decision tree classifiers, naive Bayes classifiers, and, SVMs provide really poor

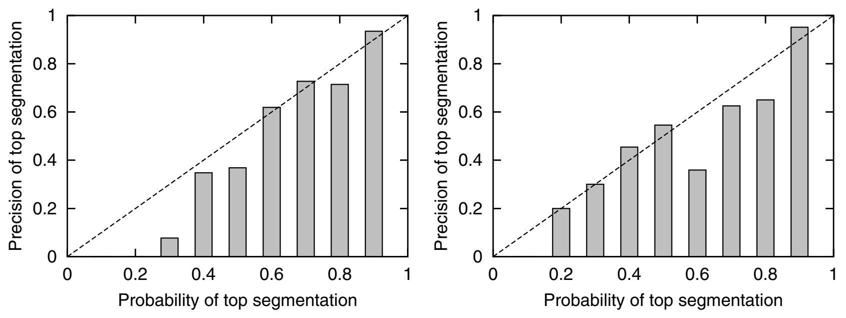

Fig. 5.3 Reliability plots for two datasets. The diagonal line denotes the ideal ending points of the bars.

probability values. In contrast, classifiers like logistic regressions and neural networks provide very sound estimates. CRFs are a generalization of logistic regression classifiers, and HMMs are a generalization of naive Bayes classifiers. So, we expect similar conclusions to hold for the relative performance of these two entity extraction models.

It is easy to extend sequence models like CRFs to return a set of k highest probability extractions instead of a single most likely extraction. We only need to change the Viterbi algorithm discussed in Section 3.5.1 to maintain top-k highest scoring solution at each position.

Uncertainty management in rule-based models is more difficult than for statistical models that are rooted in probabilistic modeling to start with. Many rule-learning systems associate each rule with a precision value that indicates for all firings of the rule condition, the fraction of cases where the action associated with the rule is correct. However, there is little principled work on obtaining sound probabilities when an extraction is due to the combined application of many rules, or when the firings of multiple rules overlap. Even for a single rule, obtaining confidence estimates is not easy as evidenced by the poor performance of decision tree classifiers in the evaluation conducted in [158].

5.4.2 Multi-attribute Extractions

We now consider the case where we are extracting multiple attributes of an entity from a single source string. Examples include the extraction of fields like House number, Area, City, and Zipcode from address strings and the extraction of model name, make, number of doors and price from car sales ads. Unlike in the previous case we cannot assume that the different attributes extracted are independent of each other.

Assume that the results of these extractions are stored as multiple columns of a single database table. A simple extension of the previous multi-row imprecision model is to maintain with each row a probability value exactly as in the single column case. An example for the case of address strings is given below where we show four possible extractions of the fields of an address from the string "52-A Goregaon West Mumbai 400 076." Again, these four rows together provide a more informative summary of the imprecision in the data than possible with the highest probability row alone.

Id	House_no	Area	City	Pincode	Prob
1	52	Goregaon West	Mumbai	400 062	0.1
1	52-A	Goregaon	West Mumbai	400 062	0.2
1	52-A	Goregaon West	Mumbai	400 062	0.5
1	52	Goregaon	West Mumbai	400 062	0.2

However, for multi-attribute data, another possible method of representing uncertainty is through a probability distribution attached to each column. An example is shown below:

Id	House_no	Area	City	Pincode
1	52 (0.3) 52-A (0.7)	Goregaon West (0.6) Goregaon (0.4)	Mumbai (0.6) West Mumbai (0.4)	400 062 (1.0)

In the above, each column stores a distribution of the possible values that it can take. The probability of any specific combination of values is obtained by multiplying the probability of the corresponding values from each column. This representation is more compact than explicitly storing the probability for each possible combination at the row-level. However, a downside of the method is that it may not faithfully capture the probabilities of the original extraction output. [102] proposes a hybrid method of keeping both row and column level distributions and an example is shown below.

Id	House_no	Area	City	Pincode	Prob
1	52 (0.167) 52-A (0.833)	Goregaon West (1.0)	Mumbai (1.0)	400 062 (1.0)	0.6
1	52 (0.5) 52-A (0.5)	Goregaon (1.0)	West Mumbai (1.0)	400 062 (1.0)	0.4

The creation of such imprecise models is however not easy. For the specific case of a CRF model, [102] presents algorithms for obtaining the best model for each unstructured source. Since the details of the steps are specific to CRFs we do not present the method here. An interesting future work is creating such hybrid models from other baseline extraction models.

5.4.3 Multiple Redundant Extractions

When the same information is extracted from multiple sources, there is a need to reason about the resultant uncertainty out of a data integration phase following an extraction phase. Consider the simple case of extracting single attribute entities from multiple independent sources. Also, for the sake of simplicity, assume that each extraction generates a single most likely output with a probability denoting its confidence. As an example, consider the table of extractions below:

Id	Title	Pr
1	Last Theorem	0.5
2	"Transaction Processing: Concepts and Techniques"	0.95
3	Transaction Processing	0.4
4	Fermat's Last Theorem	0.3
5	The Fermat's Last Theorem	0.5
6	Transaction Processing: Concepts and Techniques	1.0
7	Transaction Processing Concepts and Techniques	1.0
8	Fermat's Last Theorem	0.9
9	Fermat's Last Theorem	0.8

There are two kinds of uncertainties in the above table: First is the single source extraction uncertainty indicated by the probabilities in each row. Second, we have co-reference uncertainty between every pair of strings of whether they refer to the same entity or not (not quantified in the table). Including both these sources of uncertainties into a single uncertainty value is a challenging problem. Existing work addresses one of the following two settings:

(1) Assume only extraction uncertainty and ignore co-reference uncertainty by assuming that an exact method exists for resolving if two strings are the same.
(2) Assume there is only co-reference uncertainty and each string has no uncertainty attached to it referring to an entity.

We will discuss only the first kind of uncertainty and refer the reader to [145, 162] for uncertainty models for the second setting. We present three different methods of combining evidence from multiple extractions into a single uncertainty value.

The Noisy-OR Model: Consider the case where for a given extracted string $\mathbf{x}$ we have n repetitions, and the probabilities attached with the n extractions are $p_1,\ldots,p_n$. Our goal is to convert this into a single probability value p of $\mathbf{x}$ being a correct instance of the entity type y. One simple approach is to assume that each extraction is an independent event. Then the probability that $\mathbf{x}$ is of type y is equal to the probability that at least one of the extractions is correct. This is called the noisy-OR model and is given by

$$p = 1 - \prod_{i=1}^{n}(1 - p_i). \tag{5.3}$$

Thus, in our example table, the probability that the string "Fermat's Last Theorem" that appears thrice with probability 0.8, 0.9, and 0.3 is a book title is calculated as $1 - 0.2 \times 0.1 \times 0.7 = 0.986$ — a very high probability indeed.

A shortcoming of the noisy-OR model is that the assumption that different extractions are independent of each other often does not hold in practice. To see an extreme case where the noisy-OR model can give nonintuitive results, consider the case where we have 100 extractions of a string each with a confidence of 0.1. The combined probability in this case is very close to 1, but if all 100 extractions were due to the same incorrect pattern then it is quite likely that the pattern is wrong given the individual confidence value of 0.1. The soft-OR function is another proposed method that chooses the overall precision as the maximum of any of the existing precision.

$$p = \max_{i=1}^{n} p_i. \tag{5.4}$$

In this example, soft-OR will return a confidence of 0.1. However, it seems rather pessimistic to stick to a confidence value of 0.1 given 100 occurrences of the string.

We next discuss more sophisticated models that reward repetitions without assuming independence. These models assume that there are multiple patterns that can generate an entity. Let $M_1,\ldots,M_k$ denote these patterns. For example, in a rule learning system the patterns could correspond to the different rules whose action lead to the extraction of an entity of a given type y. Each extraction is associated with the pattern that generated it. Thus, for a given string $\mathbf{x}$ we can calculate counts $n_1,\ldots,n_K$ of the number of times $\mathbf{x}$ was extracted using pattern $M_1,\ldots,M_K$, respectively. For this part, we ignore the confidence values attached with individual extractions. Our goal is to use the counts $n_1,\ldots n_K$ to estimate for a given string $\mathbf{x}$ the probability that it is indeed an instance of type y. That is, we want, $\Pr(\mathbf{x} \in y|n_1,\ldots,n_K)$.

Two category of methods have been proposed for this. We first present an easy to implement method applicable for the case where we have labeled data. Later we present a more complicated generative method that does not necessarily require labeled data.

Conditional Probability Models from Labeled Data: This approach based on labeled data does not make any assumption about the independence of the different patterns. Instead, it directly learns the conditional probability $\Pr(y|n_1,\ldots,n_K)$ using well-known methods such as logistic classifiers as follows:

$$\Pr(y|n_1,\ldots,n_K) = \frac{1}{1 + \exp(\sum_{i=1}^{K} w_i n_i + b)} \tag{5.5}$$

the parameters $w_1 \cdots w_K, b$ are estimated using labeled data in an offline training phase.

Generative Models for Unlabeled Data: The second method taken from [82] is more involved and can be skipped on first reading.

Consider the case of first a single pattern M_1. Let n_{1j} denote the total number of times string $\mathbf{x}_j$ is extracted using M_1. Therefore, the total number of times pattern M_1 fired over all strings is $\sum_j n_{1j} = n_1$. Assume we have a distribution function for each class y, $\Pr(f|y)$ that gives the probability that $\mathbf{x}$ appears in fraction f of the total cases n_1. We can then use Bayes rule to estimate $\Pr(y|n_{1j})$ as

$$\Pr(y|n_{1j}) = \frac{\sum_f C(n_1, n_{1j}) f^{n_{1j}} (1-f)^{n_1 - n_{1j}} \Pr(f|y) \Pr(y)}{\sum_{y=0,1} \sum_f C(n_1, n_{1j}) f^{n_{1j}} (1-f)^{n_1 - n_{1j}} \Pr(f|y) \Pr(y)}. \tag{5.6}$$

In the above equation, the outer summation goes over all possible values f of the probability of getting $\mathbf{x}$ from pattern M_1. The unknown terms in the above are $\Pr(f|y)$ and $\Pr(y)$. Before we discuss how these are estimated, let us generalize to the case of multiple patterns.

A straightforward generalization is to assume that we have a joint distribution over the full space of the fractional occurrence of each string in each pattern. Thus, if we had a full distribution $\Pr(f_1 \cdots f_K|y)$ we could generalize the above equation to estimate the joint distribution $\Pr(y|n_{1j} \cdots n_{Kj})$ as:

$$\frac{\sum_{f_1,\ldots,f_K} \prod_{i=1}^{K} C(n_i, n_{ij}) f_i^{n_{ij}} (1-f_i)^{n_i - n_{ij}} \Pr(f_1 \cdots f_K|y) \Pr(y)}{\sum_{y=0,1} \sum_{f_1,\ldots,f_K} \prod_{i=1}^{K} C(n_i, n_{ij}) f_i^{n_{ij}} (1-f_i)^{n_i - n_{ij}} \Pr(f_1 \cdots f_K|y) \Pr(y)}. \tag{5.7}$$

However, it is difficult to estimate the joint distribution. Downey et al. [82] simplifies the space in the following way. For the case when $y = 1$, create a discrete distribution as follows. Assume there are C distinct strings of type y numbered arbitrarily from 1 to C. The jth string has probability f_{ij} in pattern i and is given by a Zipfian distribution as:

$$f_{ij} = p_i \frac{j^{-z_C}}{\sum_k k^{-z_C}},$$

where p_i is a user-provided term denoting the precision of pattern M_i, z_c is a parameter that is estimated.

Similarly, for the case of $y = 0$, that is all the wrong strings, assume there are a total E of them and a different parameter z_E. Using this we get a different set of E fractions calculated for different values of j as

$$g_{ij} = (1 - p_i)\frac{j^{-z_E}}{\sum_k k^{-z_E}}.$$

With these values calculated, the joint probability is calculated as follows: $\Pr(v_1 \cdots v_K | y = 1) = \frac{1}{C}$ if there exists an index j for which $v_i = f_{ij}$. If no such index exists the probability is zero. $\Pr(v_1 \cdots v_K | y = 0) = \frac{1}{E}$ if for each pattern i, we have an index position k_i such that $v_i = g_{ik_i}$. Note for the case of $y = 1$ we assume that the probabilities are correlated in the sense that there exists a shared ordering of the strings across the different patterns. This captures the intuition that strings which have high probability of occurrence in one pattern will also have high probability of occurrence in another. In contrast, for the case of $y = 0$ we do not make any such assumption. The values of z_C, z_E, C, and E are calculated using an expectation minimization procedure where the hidden variable denotes if a particular string is an instance of y or not.

5.4.4 Other Techniques for Managing Extraction Errors

There are many other dimensions to the handling of imprecision in extraction systems that we have not discussed here. One important issue is querying the results of uncertain extractions. The approach in this discussion have been to represent extraction uncertainty in terms of generic imprecision models, and then leverage on the active research on efficient querying of uncertain data [28, 55, 75, 183] for getting probabilistic answers. However, since automatically extracted data is most popular in decision support systems, models and algorithms for answering multidimensional aggregate queries on such imprecise data is a popular research problem on its own right. We refer the reader to [34, 35] for existing work in this direction. Another practical solution to the problem of uncertain extractions is to provide provenance support in databases created out of the results of extractions. This can enable a user to track down erroneous results or missed extractions as shown in [107]. Other issues that we have not discussed here are: modeling the reliability of the unstructured source, capturing the dependence of current extraction on the outputs of previous imprecise extractions, and managing uncertainties with evolving data.

Overall, existing methods for the management of imprecision in information extraction is still at a preliminary stage. Substantially more experience is needed before we understand imprecision in a way that we can create reliable models that are useful without being unmanageably complex.

6

Concluding Remarks

Many applications depend on the automatic extraction of structure from unstructured data for better means of querying, organizing, and analyzing data connecting the structured and unstructured world. Starting from research in the natural language community on basic named entity recognition systems, the topic now engages a veritable community of researchers spanning machine learning, databases, web, and information retrieval. A lot of work now exists on various aspects of the information extraction problem including core statistical and rule-based models, frameworks and architectures for managing the extraction pipelines, performance optimization, uncertainty management, and so on.

In the first part of the survey, we concentrated on core models for the extraction of entities and relationships via rule-based and statistical models. We presented different forms of rules for entity extraction and for resolving conflicts when multiple rules fire on overlapping tokens. Most often rule-based systems are manually coded but there also exist efficient methods for learning rules from labeled data. When labeled data is available, such rule-learning algorithms provide a valuable starting set of rules for manual tuning later. Statistical methods are more useful when the input sources are noisy so that hard predicates of rule-based extractors hurt. We described Conditional Random Fields, a state-of-the-art method for entity recognition that imposes a joint distribution over the sequence of entity labels assigned to a given sequence of tokens. Although the details of training and inference on statistical models are somewhat involved for someone outside the field of statistical machine learning, the models are easy to deploy and customize due to their fairly nonrestrictive feature based framework.

On the topic of relationship extraction we considered two scenarios: one where entities are already annotated in the corpus and our goal is to classify the type of relationship that exists between a given entity pair, and second where for a given relationship type we need to retrieve all instances of entity pairs from an un-annotated corpus given only a seed set of pairs that exhibit that relationship. For the first problem, we presented different methods of combining clues from intervening words and their properties, parse trees, and dependency graphs to perform the classification. The solution for the second problem builds upon the first and the two additional challenges: training models based on only a seed database, and designing filter queries to retrieve only the relevant portions of a large corpus. We discussed existing methods for solving these problems, but compared to entity extraction, the problem of relationship extraction from open domains is still at a preliminary stage of exploration. More extensive study from a larger community of researchers is needed to get consensus on sound ideas that generalize.

We then discussed existing work on a number of practical issues like performance optimization, dynamic data management, uncertainty handling, and data integration.

In spite of more than two decades of research on designing models for extraction, accuracy continues to be of prime concern. While for basic named entities, it is possible to achieve close to 90% accuracy with state-of-the-art extraction models, for relationship extraction the accuracy is in the neighborhood of 70% even in restricted domains such as news articles. More complex types of extractions such as, extraction of long entities, soft attributes of entities, and higher order structures, are only starting to be explored. Although there are many existing systems that use information extraction in a core way, published research literature providing principled solutions to many of the practical issues arising in the management of extraction systems are only starting to appear. The time is ripe now for lot more exciting and useful work in practical large-scale deployments of information extraction systems.

Acknowledgments

I am indebted to many people for making this survey possible. First, I would like to thank Surajit Chaudhuri for getting me to take a break from research and spend time on the survey instead. I thank the anonymous reviewers and AnHai Doan for suffering through my hurriedly drafted first version and helping me produce a significantly improved final version. Ganesh Ramakrishnan helped me with the rule-learning section by providing important references and helping me maintain balance with the statistical methods. Soumen Chakrabarti deserves thanks for encouraging me to stay focused on completing the survey.

My life as a researcher during this period was supported by grants from Microsoft Research and IBM, and IIT Bombay, my employer.

References

[1] 2004. ACE. Annotation guidelines for entity detection and tracking.

[2] E. Agichtein, "Extracting relations from large text collections," PhD thesis, Columbia University, 2005.

[3] E. Agichtein and V. Ganti, "Mining reference tables for automatic text segmentation," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Seattle, USA, 2004.

[4] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.

[5] E. Agichtein and L. Gravano, "Querying text databases for efficient information extraction," in *ICDE*, 2003.

[6] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," in *Advances in Knowledge Discovery and Data Mining*, (U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, eds.), ch. 12, pp. 307–328, AAAI/MIT Press, 1996.

[7] J. Aitken, "Learning information extraction rules: An inductive logic programming approach," in *Proceedings of the 15th European Conference on Artificial Intelligence*, pp. 355–359, 2002.

[8] R. Ananthakrishna, S. chaudhuri, and V. Ganti, "Eliminating fuzzy duplicates in data warehouses," in *International Conference on Very Large Databases (VLDB)*, 2002.

[9] R. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, vol. 6, pp. 1817–1853, 2005.

[10] D. E. Appelt, J. R. Hobbs, J. Bear, D. J. Israel, and M. Tyson, "Fastus: A finite-state processor for information extraction from real-world text," in *IJCAI*, pp. 1172–1178, 1993.

[11] A. Arasu, H. Garcia-Molina, and S. University, "Extracting structured data from web pages," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 337–348, 2003.

[12] S. Argamon-Engelson and I. Dagan, "Committee-based sample selection for probabilistic classifiers," *Journal of Artificial Intelligence Research*, vol. 11, pp. 335–360, 1999.

[13] M.-F. Balcan, A. Beygelzimer, and J. Langford, "Agnostic active learning," in *ICML*, pp. 65–72, 2006.

[14] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web," in *IJCAI*, pp. 2670–2676, 2007.

[15] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," in *FOCS '02: Proceedings of the 43rd Symposium on Foundations of Computer Science*, USA, Washington, DC: IEEE Computer Society, 2002.

[16] G. Barish, Y.-S. Chen, D. DiPasquo, C. A. Knoblock, S. Minton, I. Muslea, and C. Shahabi, "Theaterloc: Using information integration technology to rapidly build virtual applications," in *International Conference on Data Engineering (ICDE)*, pp. 681–682, 2000.

[17] R. Baumgartner, S. Flesca, and G. Gottlob, "Visual web information extraction with lixto," in *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 119–128, USA, San Francisco, CA: Morgan Kaufmann Publishers Inc, 2001.

[18] M. Berland and E. Charniak, "Finding parts in very large corpora," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 57–64, 1999.

[19] M. Bhide, A. Gupta, R. Gupta, P. Roy, M. K. Mohania, and Z. Ichhaporia, "Liptus: Associating structured and unstructured information in a banking environment," in *SIGMOD Conference*, pp. 915–924, 2007.

[20] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel, "Nymble: A high-performance learning name-finder," in *Proceedings of ANLP-97*, pp. 194–201, 1997.

[21] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name-matching in information integration," *IEEE Intelligent Systems*, 2003.

[22] 2006. Biocreative — critical assessment for information extraction in biology. http://biocreative.sourceforge.net/.

[23] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

[24] A. Bordes, L. Bottou, P. Gallinari, and J. Weston, "Solving multiclass support vector machines with larank," in *ICML*, pp. 89–96, 2007.

[25] V. R. Borkar, K. Deshmukh, and S. Sarawagi, "Automatic text segmentation for extracting structured records," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, Santa Barabara, USA, 2001.

[26] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman, "Exploiting diverse knowledge sources via maximum entropy in named entity recognition," in *Sixth Workshop on Very Large Corpora New Brunswick*, New Jersey, Association for Computational Linguistics, 1998.

[27] L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning, number LNAI 3176 in Lecture Notes in Artificial Intelligence*, (O. Bousquet and U. von Luxburg, eds.), pp. 146–168, Springer Verlag, 2004.

[28] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu, "Mystiq: A system for finding more answers by using probabilities," in *ACM SIGMOD*, 2005.

[29] A. Z. Broder, M. Fontoura, V. Josifovski, and L. Riedel, "A semantic approach to contextual advertising," in *SIGIR*, pp. 559–566, 2007.

[30] R. Bunescu and R. Mooney, "Learning to extract relations from the web using minimal supervision," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 576–583, June 2007.

[31] R. Bunescu and R. J. Mooney, "Collective information extraction with relational markov networks," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pp. 439–446, 2004.

[32] R. C. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani, and Y. W. Wong, "Comparative experiments on learning information extractors for proteins and their interactions," *Artificial Intelligence in Medicine*, vol. 33, pp. 139–155, 2005.

[33] R. C. Bunescu and R. J. Mooney, "A shortest path dependency kernel for relation extraction," in *HLT '05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 724–731, USA, Morristown, NJ: Association for Computational Linguistics, 2005.

[34] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan, "OLAP over uncertain and imprecise data," in *Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 970–981, VLDB Endowment, 2005.

[35] D. Burdick, A. Doan, R. Ramakrishnan, and S. Vaithyanathan, "Olap over imprecise data with domain constraints," in *VLDB*, pp. 39–50, 2007.

[36] M. Cafarella, N. Khoussainova, D. Wang, E. Wu, Y. Zhang, and A. Halevy, "Uncovering the relational web," in *WebDB*, 2008.

[37] M. J. Cafarella, D. Downey, S. Soderland, and O. Etzioni, "KnowItNow: Fast, scalable information extraction from the web," in *Conference on Human Language Technologies (HLT/EMNLP)*, 2005.

[38] M. J. Cafarella and O. Etzioni, "A search engine for natural language applications," in *WWW*, pp. 442–452, 2005.

[39] M. J. Cafarella, C. Re, D. Suciu, and O. Etzioni, "Structured querying of web text data: A technical challenge," in *CIDR*, pp. 225–234, 2007.

[40] D. Cai, ShipengYu, Ji-RongWen, and W.-Y. Ma, "Vips: A vision based page segmentation algorithm," Technical Report MSR-TR-2003-79, Microsoft, 2004.

[41] Y. Cai, X. L. Dong, A. Y. Halevy, J. M. Liu, and J. Madhavan, "Personal information management with semex," in *SIGMOD Conference*, pp. 921–923, 2005.

[42] M. Califf and R. Mooney, *Bottom-up Relational Learning of Pattern Matching Rules for Information Extraction*, 2003.

[43] M. E. Califf and R. J. Mooney, "Relational learning of pattern-match rules for information extraction," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 328–334, July 1999.

[44] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. K. Mohania, "Efficiently linking text documents with relevant structured information," in *VLDB*, pp. 667–678, 2006.

[45] S. Chakrabarti, *Mining the Web: Discovering Knowledge from Hypertext Data.* Morgan-Kauffman, 2002.

[46] S. Chakrabarti, J. Mirchandani, and A. Nandi, "Spin: searching personal information networks," in *SIGIR*, p. 674, 2005.

[47] S. Chakrabarti, K. Punera, and M. Subramanyam, "Accelerated focused crawling through online relevance feedback," in *WWW*, Hawaii, ACM, May 2002.

[48] S. Chakrabarti, K. Puniyani, and S. Das, "Optimizing scoring functions and indexes for proximity search in type-annotated corpora," in *WWW*, pp. 717–726, 2006.

[49] A. Chandel, P. Nagesh, and S. Sarawagi, "Efficient batch top-k search for dictionary-based entity recognition," in *Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE)*, 2006.

[50] M. Charikar, V. Guruswami, and A. Wirth, "Clustering with qualitative information," *Journal of Computer and Systems Sciences*, vol. 71, pp. 360–383, 2005.

[51] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and efficient fuzzy match for online data cleaning," in *SIGMOD*, 2003.

[52] Chelba and Acero, "Adaptation of maximum entropy capitalizer: Little data can help a lot," in *EMNLP*, 2004.

[53] F. Chen, A. Doan, J. Yang, and R. Ramakrishnan, "Efficient information extraction over evolving text data," in *ICDE*, 2008.

[54] D. Cheng, R. Kannan, S. Vempala, and G. Wang, "A divide-and-merge methodology for clustering," *ACM Transactions on Database Systems*, vol. 31, pp. 1499–1525, 2006.

[55] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pp. 551–562, USA, New York, NY: ACM Press, 2003.

[56] B. Chidlovskii, B. Roustant, and M. Brette, "Documentum eci self-repairing wrappers: Performance analysis," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pp. 708–717, USA, New York, NY: ACM, 2006.

[57] 1998. N. A. Chinchor, *Overview of MUC-7/MET-2*.

[58] J. Cho and S. Rajagopalan, "A fast regular expression indexing engine," in *ICDE*, pp. 419–430, 2002.

[59] Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan, "Identifying sources of opinions with conditional random fields and extraction patterns," in *HLT/EMNLP*, 2005.

[60] F. Ciravegna, "Adaptive information extraction from text by rule induction and generalisation," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI2001)*, 2001.

[61] W. Cohen and J. Richman, "Learning to match and cluster entity names," in *ACM SIGIR' 01 Workshop on Mathematical/Formal Methods in Information Retrieval*, 2001.

[62] W. W. Cohen, M. Hurst, and L. S. Jensen, "A flexible learning system for wrapping tables and lists in html documents," in *Proceedings of the 11th World Wide Web Conference (WWW2002)*, 2002.

[63] W. W. Cohen, E. Minkov, and A. Tomasic, "Learning to understand web site update requests," in *IJCAI*, pp. 1028–1033, 2005.

[64] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," in *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003. (To appear).

[65] W. W. Cohen and S. Sarawagi, "Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[66] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," in *Advances in Neural Information Processing Systems*, (G. Tesauro, D. Touretzky, and T. Leen, eds.), pp. 705–712, The MIT Press, 1995.

[67] V. Crescenzi, G. Mecca, P. Merialdo, and P. Missier, "An automatic data grabber for large web sites," in *vldb'2004: Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pp. 1321–1324, 2004.

[68] A. Culotta, T. T. Kristjansson, A. McCallum, and P. A. Viola, "Corrective feedback and persistent learning for information extraction," *Artificial Intelligence*, vol. 170, nos. 14–15, pp. 1101–1122, 2006.

[69] A. Culotta and J. Sorensen, "Dependency tree kernels for relation extraction," in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pp. 423–429, Barcelona, Spain, July 2004.

[70] C. Cumby and D. Roth, "Feature extraction languages for propositionalzed relational learning," in *Working Notes of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data (SRL-2003)*, (L. Getoor and D. Jensen, eds.), pp. 24–31, Acapulco, Mexico, August 11, 2003.

[71] H. Cunningham, "Information extraction, automatic," *Encyclopedia of Language and Linguistics*, 2005. second ed.

[72] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "Gate: A framework and graphical development environment for robust nlp tools and applications," in *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[73] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A framework and graphical development environment for robust nlp tools and applications," in *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, 2002.

[74] E. Cutrell and S. T. Dumais, "Exploring personal information," *Communications on ACM*, vol. 49, pp. 50–51, 2006.

[75] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," in *VLDB*, pp. 864–875, 2004.

[76] S. Dasgupta, "Coarse sample complexity bounds for active learning," in *NIPS*, 2005.

[77] H. Daumé III, "Frustratingly easy domain adaptation," in *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, 2007.

[78] P. DeRose, W. Shen, F. C. 0002, Y. Lee, D. Burdick, A. Doan, and R. Ramakrishnan, "Dblife: A community information management platform for the database research community (demo)," in *CIDR*, pp. 169–172, 2007.

[79] T. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic and Statistical Pattern Recognition; Lecture Notes in Computer Science*, (T. Caelli, ed.), Vol. 2396, pp. 15–30, Springer-Verlag, 2002.

[80] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, 1999.

[81] D. Downey, M. Broadhead, and O. Etzioni, "Locating complex named entities in web text," in *IJCAI*, pp. 2733–2739, 2007.

[82] D. Downey, O. Etzioni, and S. Soderland, "A probabilistic model of redundancy in information extraction," in *IJCAI*, 2005.

[83] D. Downey, S. Schoenmackers, and O. Etzioni, "Sparse information extraction: Unsupervised language models to the rescue," in *ACL*, 2007.

[84] D. W. Embley, M. Hurst, D. P. Lopresti, and G. Nagy, "Table-processing paradigms: A research survey," *IJDAR*, vol. 8, nos. 2–3, pp. 66–86, 2006.

[85] D. W. Embley, Y. S. Jiang, and Y.-K. Ng, "Record-boundary discovery in web documents," in *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999*, pp. 467–478, Philadephia, Pennsylvania, USA, 1999.

[86] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in KnowItAll: (preliminary results)," in *WWW*, pp. 100–110, 2004.

[87] O. Etzioni, B. Doorenbos, and D. Weld, "A scalable comparison shopping agent for the world-wide web," in *Proceedings of the International Conference on Autonomous Agents*, 1997.

[88] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," *Journal of Computer and System Sciences*, vol. 66, nos. 614, 656, September 2001.

[89] R. Feldman, B. Rosenfeld, and M. Fresko, "Teg-a hybrid approach to information extraction," *Knowledge and Information Systems*, vol. 9, pp. 1–18, 2006.

[90] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Society*, vol. 64, pp. 1183–1210, 1969.

[91] D. Ferrucci and A. Lally, "Uima: An architectural approach to unstructured information processing in the corporate research environment," *Natural Language Engineering*, vol. 10, nos. 3–4, pp. 327–348, 2004.

[92] J. R. Finkel, T. Grenager, and C. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, 2005.

[93] J. R. Finkel, T. Grenager, and C. D. Manning, "Incorporating non-local information into information extraction systems by gibbs sampling," in *ACL*, 2005.

[94] G. W. Flake, E. J. Glover, S. Lawrence, and C. L. Giles, "Extracting query modifications from nonlinear svms," in *WWW*, pp. 317–324, 2002.

[95] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine Learning*, vol. 28, nos. 2–3, pp. 133–168, 1997.

[96] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from web tables," in *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pp. 71–80, ACM, 2007.

[97] R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano, "Text mining for product attribute extraction," *SIGKDD Explorations Newsletter*, vol. 8, pp. 41–48, 2006.

[98] R. Grishman, "Information extraction: Techniques and challenges," in *SCIE*, 1997.

[99] R. Grishman, S. Huttunen, and R. Yangarber, "Information extraction for enhanced access to disease outbreak reports," *Journal of Biomedical Informatics*, vol. 35, pp. 236–246, 2002.

[100] R. Grishman and B. Sundheim, "Message understanding conference-6: A brief history," in *Proceedings of the 16th Conference on Computational Linguistics*, pp. 466–471, USA, Morristown, NJ: Association for Computational Linguistics, 1996.

[101] R. Gupta, A. A. Diwan, and S. Sarawagi, "Efficient inference with cardinality-based clique potentials," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*, USA, 2007.

[102] R. Gupta and S. Sarawagi, "Curating probabilistic databases from information extraction models," in *Proceedings of the 32nd International Conference on Very Large Databases (VLDB)*, 2006.

[103] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo, "Extracting semistructure information from the web," in *Workshop on Mangement of Semistructured Data*, 1997.

[104] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang, "Accessing the deep web," *Communications on ACM*, vol. 50, pp. 94–101, 2007.

[105] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th Conference on Computational Linguistics*, pp. 539–545, 1992.

[106] C.-N. Hsu and M.-T. Dung, "Generating finite-state transducers for semistructured data extraction from the web," *Information Systems Special Issue on Semistructured Data*, vol. 23, 1998.

[107] J. Huang, T. Chen, A. Doan, and J. Naughton, *On the Provenance of Non-answers to Queries Over Extracted Data.*

[108] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," in *Advances in Neural Information Processing Systems 20*, Cambridge, MA: MIT Press, 2007.

[109] M. Hurst, "The interpretation of tables in texts," PhD thesis, University of Edinburgh, School of Cognitive Science, Informatics, University of Edinburgh, 2000.

[110] P. G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano, "Towards a query optimizer for text-centric tasks," *ACM Transactions on Database Systems*, vol. 32, 2007.

[111] N. Ireson, F. Ciravegna, M. E. Califf, D. Freitag, N. Kushmerick, and A. Lavelli, "Evaluating machine learning for information extraction," in *ICML*, pp. 345–352, 2005.

[112] M. Jansche and S. P. Abney, "Information extraction from voicemail transcripts," in *EMNLP '02: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pp. 320–327, USA, Morristown, NJ: Association for Computational Linguistics, 2002.

[113] T. S. Jayram, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu, "Avatar information extraction system," *IEEE Data Engineering Bulletin*, vol. 29, pp. 40–48, 2006.

[114] J. Jiang and C. Zhai, "A systematic exploration of the feature space for relation extraction," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pp. 113–120, 2007.

[115] N. Kambhatla, "Combining lexical, syntactic and semantic features with maximum entropy models for information extraction," in *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pp. 178–181, Barcelona, Spain: Association for Computational Linguistics, July 2004.

[116] S. Khaitan, G. Ramakrishnan, S. Joshi, and A. Chalamalla, "Rad: A scalable framework for annotator development," in *ICDE*, pp. 1624–1627, 2008.

[117] M.-S. Kim, K.-Y. Whang, J.-G. Lee, and M.-J. Lee, "n-gram/2l: A space and time efficient two-level n-gram inverted index structure," in *VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases*, pp. 325–336, 2005.

[118] D. Klein and C. D. Manning, "Conditional structure versus conditional estimation in NLP models," in *Workshop on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[119] D. Koller and N. Friedman, "Structured probabilistic models," Under preparation, 2007.

[120] V. Krishnan and C. D. Manning, "An effective two-stage model for exploiting non-local dependencies in named entity recognition," in *ACL-COLING*, 2006.

[121] N. Kushmerick, “Wrapper induction for information extraction,” PhD thesis, University of Washington, 1997.

[122] N. Kushmerick, “Regression testing for wrapper maintenance,” in *AAAI/IAAI*, pp. 74–79, 1999.

[123] N. Kushmerick, D. Weld, and R. Doorenbos, “Wrapper induction for information extraction,” in *Proceedings of IJCAI*, 1997.

[124] S. R. Labeling 2008. http://www.lsi.upc.es/ srlconll/refs.html.

[125] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.

[126] S. Lawrence, C. L. Giles, and K. Bollacker, “Digital libraries and autonomous citation indexing,” *IEEE Computer*, vol. 32, pp. 67–71, 1999.

[127] W. Lehnert, J. McCarthy, S. Soderland, E. Riloff, C. Cardie, J. Peterson, F. Feng, C. Dolan, and S. Goldman, “Umass/hughes: Description of the circus system used for tipster text,” in *Proceedings of a Workshop on Held at Fredericksburg, Virginia*, pp. 241–256, USA, Morristown, NJ: Association for Computational Linguistics, 1993.

[128] K. Lerman, S. Minton, and C. A. Knoblock, “Wrapper maintenance: A machine learning approach,” *Journal of Artificial Intellgence Research (JAIR)*, vol. 18, pp. 149–181, 2003.

[129] X. Li and J. Bilmes, “A bayesian divergence prior for classifier adaptation,” *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS-2007)*, 2007.

[130] Y. Li and K. Bontcheva, “Hierarchical, perceptron-like learning for ontology-based information extraction,” in *WWW ’07: Proceedings of the 16th International Conference on World Wide Web*, pp. 777–786, ACM, 2007.

[131] B. Liu, M. Hu, and J. Cheng, “Opinion observer: Analyzing and comparing opinions on the web,” in *WWW ’05: Proceedings of the 14th international conference on World Wide Web*, pp. 342–351, 2005.

[132] D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large-scale optimization,” *Mathematic Programming*, vol. 45, pp. 503–528, 1989.

[133] L. Liu, C. Pu, and W. Han, “Xwrap: An xml-enabled wrapper construction system for web information sources,” in *International Conference on Data Engineering (ICDE)*, pp. 611–621, 2000.

[134] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, “Tableseer: Automatic table metadata extraction and searching in digital libraries,” in *JCDL ’07: Proceedings of the 2007 Conference on Digital Libraries*, pp. 91–100, USA, New York, NY: ACM, 2007.

[135] R. Malouf, “Markov models for language-independent named entity recognition,” in *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, 2002.

[136] R. Malouf, “A comparison of algorithms for maximum entropy parameter estimation,” in *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pp. 49–55, 2002.

[137] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA: The MIT Press, 1999.

[138] I. Mansuri and S. Sarawagi, “A system for integrating unstructured data into relational databases,” in *Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE)*, 2006.

[139] S. Mao, A. Rosenfeld, and T. Kanungo, “Document structure analysis algorithms: A literature survey,” *Document Recognition and Retrieval X*, vol. 5010, pp. 197–207, 2003.

[140] B. Marthi, B. Milch, and S. Russell, “First-order probabilistic models for information extraction,” in *Working Notes of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data (SRL-2003)*, (L. Getoor and D. Jensen, eds.), pp. 71–78, Acapulco, Mexico, August 11 2003.

[141] D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks, “Named entity recognition from diverse text types,” *Recent Advances in Natural Language Processing 2001 Conference*, Tzigov Chark, Bulgaria, 2001.

[142] A. McCallum, “Information extraction: Distilling structured data from unstructured text,” *ACM Queue*, vol. 3, pp. 48–57, 2005.

[143] A. McCallum, D. Freitag, and F. Pereira, “Maximum entropy markov models for information extraction and segmentation,” in *Proceedings of the International Conference on Machine Learning (ICML-2000)*, pp. 591–598, Palo Alto, CA, 2000.

[144] A. McCallum, K. Nigam, J. Reed, J. Rennie, and K. Seymore, *Cora: Computer Science Research Paper Search Engine*, http://cora.whizbang.com/, 2000.

[145] A. McCallum and B. Wellner, “Toward conditional models of identity uncertainty with application to proper noun coreference,” in *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pp. 79–86, Acapulco, Mexico, August 2003.

[146] A. K. McCallum, *Mallet: A Machine Learning for Language Toolkit.* http://mallet.cs.umass.edu, 2002.

[147] D. McDonald, H. Chen, H. Su, and B. Marshall, “Extracting gene pathway relations using a hybrid grammar: The arizona relation parser,” *Bioinformatics*, vol. 20, pp. 3370–3378, 2004.

[148] R. McDonald, K. Crammer, and F. Pereira, “Flexible text segmentation with structured multilabel classification,” in *HLT/EMNLP*, 2005.

[149] G. Mecca, P. Merialdo, and P. Atzeni, “Araneus in the era of xml,” in *IEEE Data Engineering Bullettin, Special Issue on XML, IEEE*, September 1999.

[150] M. Michelson and C. A. Knoblock, “Semantic annotation of unstructured and ungrammatical text,” in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1091–1098, 2005.

[151] M. Michelson and C. A. Knoblock, “Creating relational data from unstructured and ungrammatical data sources,” *Journal of Artificial Intelligence Research (JAIR)*, vol. 31, pp. 543–590, 2008.

[152] E. Minkov, R. C. Wang, and W. W. Cohen, “Extracting personal names from email: Applying named entity recognition to informal text,” in *HLT/EMNLP*, 2005.

[153] R. J. Mooney and R. C. Bunescu, “Mining knowledge from text using information extraction,” *SIGKDD Explorations*, vol. 7, pp. 3–10, 2005.

[154] I. Muslea, “Extraction patterns for information extraction tasks: A survey,” in *The AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[155] I. Muslea, S. Minton, and C. Knoblock, “Selective sampling with redundant views,” in *Proceedings of the Fifteenth National Conference on Artificial Intelligence, AAAI-2000*, pp. 621–626, 2000.

[156] I. Muslea, S. Minton, and C. A. Knoblock, “A hierarchical approach to wrapper induction,” in *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, WA, 1999.

[157] I. Muslea, S. Minton, and C. A. Knoblock, “Hierarchical wrapper induction for semistructured information sources,” *Autonomous Agents and Multi-Agent Systems*, vol. 4, nos. 1/2, pp. 93–114, 2001.

[158] A. Niculescu-Mizil and R. Caruana, “Predicting good probabilities with supervised learning,” in *ICML*, 2005.

[159] NIST. Automatic content extraction (ACE) program. 1998–present.

[160] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Foundations and Trends in Information Retrieval*, vol. 2, nos. 1–2, pp. 1–135, 2008.

[161] Parag and P. Domingos, “Multi-relational record linkage,” in *Proceedings of 3rd Workshop on Multi-Relational Data Mining at ACM SIGKDD*, Seattle, WA, August 2004.

[162] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser, “Identity uncertainty and citation matching,” in *Advances in Neural Processing Systems 15*, Vancouver, British Columbia: MIT Press, 2002.

[163] F. Peng and A. McCallum, “Accurate information extraction from research papers using conditional random fields,” in *HLT-NAACL*, pp. 329–336, 2004.

[164] D. Pinto, A. McCallum, X. Wei, and W. B. Croft, “Table extraction using conditional random fields,” in *SIGIR ’03: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pp. 235–242, USA, New York, NY: ACM, 2003.

[165] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovič, and R. Studer, “Transforming arbitrary tables into logical form with tartar,” *Data Knowledge Engineering*, vol. 60, pp. 567–595, 2007.

[166] C. Plake, T. Schiemann, M. Pankalla, J. Hakenberg, and U. Leser, “Alibaba: Pubmed as a graph,” *Bioinformatics*, vol. 22, pp. 2444–2445, 2006.

[167] A.-M. Popescu and O. Etzioni, “Extracting product features and opinions from reviews,” in *HLT ’05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 339–346, 2005.

[168] F. Popowich, “Using text mining and natural language processing for health care claims processing,” *SIGKDD Explorartion Newsletter*, vol. 7, pp. 59–66, 2005.

[169] K. Probst and R. Ghani, “Towards ‘interactive’ active learning in multi-view feature sets for information extraction,” in *ECML*, pp. 683–690, 2007.

[170] J. R. Quinlan, “Learning logical definitions from examples,” *Machine Learning*, vol. 5, 1990.

[171] L. Rabiner, “A tutorial on Hidden Markov Models and selected applications in speech recognition,” in *Proceedings of the IEEE*, vol. 77, 1989.

[172] G. Ramakrishnan, S. Balakrishnan, and S. Joshi, “Entity annotation based on inverse index operations,” in *EMNLP*, 2006.

[173] G. Ramakrishnan, S. Joshi, S. Balakrishnan, and A. Srinivasan, “Using ilp to construct features for information extraction from semi-structured text,” in *ILP*, 2007.

[174] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis, “Automatic fragment detection in dynamic web pages and its impact on caching,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 859–874, 2005.

[175] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis, “Automatic detection of fragments in dynamically generated web pages,” in *WWW*, pp. 443–454, 2004.

[176] J. Raposo, A. Pan, M. Álvarez, and N. Ángel Vira, “Automatic wrapper maintenance for semi-structured web sources using results from previous queries,” in *SAC ’05: Proceedings of the 2005 ACM Symposium on Applied Computing*, pp. 654–659, ACM, 2005.

[177] A. Ratnaparkhi, “Learning to parse natural language with maximum entropy models,” *Machine Learning*, vol. 34, 1999.

[178] L. Reeve and H. Han, “Survey of semantic annotation platforms,” in *SAC ’05: Proceedings of the 2005 ACM Symposium on Applied Computing*, pp. 1634–1638, USA, New York, NY: ACM, 2005.

[179] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan, “An algebraic approach to rule-based information extraction,” in *ICDE*, 2008.

[180] P. Resnik and A. Elkiss, “The linguist’s search engine: An overview (demonstration),” in *ACL*, 2005.

[181] E. Riloff, “Automatically constructing a dictionary for information extraction tasks,” in *AAAI*, pp. 811–816, 1993.

[182] B. Rosenfeld and R. Feldman, “Using corpus statistics on entities to improve semi-supervised relation extraction from the web,” in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 600–607, June 2007.

[183] R. Ross, V. S. Subrahmanian, and J. Grant, “Aggregate operators in probabilistic databases,” *Journal of ACM*, vol. 52, pp. 54–101, 2005.

[184] A. Sahuguet and F. Azavant, “Building light-weight wrappers for legacy web data-sources using w4f,” in *International Conference on Very Large Databases (VLDB)*, 1999.

[185] S. Sarawagi, *The CRF Project: A Java Implementation.* http://crf.sourceforge.net, 2004.

[186] S. Sarawagi, “Efficient inference on sequence segmentation models,” in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, Pittsburgh, PA, USA, 2006.

[187] S. Sarawagi and A. Bhamidipaty, “Interactive deduplication using active learning,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2002)*, Edmonton, Canada, July 2002.

[188] S. Satpal and S. Sarawagi, “Domain adaptation of conditional probability models via feature subsetting,” in *ECML/PKDD*, 2007.

[189] K. Seymore, A. McCallum, and R. Rosenfeld, "Learning Hidden Markov Model structure for information extraction," in *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, pp. 37–42, 1999.

[190] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan, "Declarative information extraction using datalog with embedded extraction predicates," in *VLDB*, pp. 1033–1044, 2007.

[191] M. Shilman, P. Liang, and P. Viola, "Learning non-generative grammatical models for document analysis," *ICCV*, vol. 2, pp. 962–969, 2005.

[192] Y. Shinyama and S. Sekine, "Preemptive information extraction using unrestricted relation discovery," in *HLT-NAACL*, 2006.

[193] J. F. Silva, Z. Kozareva, V. Noncheva, and G. P. Lopes, "Extracting named entities. a statistical approach," in *Proceedings of the XIme Confrence sur le Traitement des Langues Naturelles — TALN, 19–22 Avril, Fez, Marroco*, (B. Bel and I. Merlien, eds.), pp. 347–351, ATALA — Association pour le Traitement Automatique des Langues, 04, 2004.

[194] P. Singla and P. Domingos, "Entity resolution with markov logic," in *ICDM*, pp. 572–582, 2006.

[195] S. Soderland, "Learning information extraction rules for semi-structured and free text," *Machine Learning*, vol. 34, 1999.

[196] F. M. Suchanek, G. Ifrim, and G. Weikum, "Combining linguistic and statistical analysis to extract relations from web documents," in *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 712–717, 2006.

[197] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pp. 697–706, 2007.

[198] B. M. Sundheim, "Overview of the third message understanding evaluation and conference," in *Proceedings of the Third Message Understanding Conference (MUC-3)*, pp. 3–16, San Diego, CA, 1991.

[199] C. Sutton and A. McCallum, "Collective segmentation and labeling of distant entities in information extraction," Technical Report TR # 04-49, University of Massachusetts Presented at ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields, July 2004.

[200] K. Takeuchi and N. Collier, "Use of support vector machines in extended named entity recognition," in *The 6th Conference on Natural Language Learning (CoNLL)*, 2002.

[201] B. Taskar, "Learning structured prediction models: A large margin approach," PhD Thesis, Stanford University, 2004.

[202] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning, "Max-margin parsing," in *EMNLP*, July 2004.

[203] B. Taskar, S. Lacoste-Julien, and M. I. Jordan, "Structured prediction, dual extragradient and bregman projections," *Journal on Machine Learning Research*, vol. 7, pp. 1627–1653, 2006.

[204] M. Theobald, G. Weikum, and R. Schenkel, "Top-k query evaluation with probabilistic guarantees," in *VLDB*, pp. 648–659, 2004.

[205] C. A. Thompson, M. E. Califf, and R. J. Mooney, "Active learning for natural language parsing and information extraction," in *Proceedings of 16th International Conference on Machine Learning*, pp. 406–414, Morgan Kaufmann, San Francisco, CA, 1999.

[206] E. F. Tjong Kim Sang and F. D. Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Seventh Conference on Natural Language Learning (CoNLL-03)*, (W. Daelemans and M. Osborne, eds.), pp. 142–147, Edmonton, Alberta, Canada: Association for Computational Linguistics, May 31–June 1, 2003. (In association with HLT-NAACL, 2003).

[207] A. Troussov, B. O'Donovan, S. Koskenniemi, and N. Glushnev, "Per-node optimization of finite-state mechanisms for natural language processing," in *CICLing*, pp. 221–224, 2003.

[208] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research (JMLR)*, vol. 6, pp. 1453–1484, September 2005.

[209] J. Turmo, A. Ageno, and N. Català, "Adaptive information extraction," *ACM Computer Services*, vol. 38, p. 4, 2006.

[210] P. D. Turney, "Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm," *Journal of Artificial Intelligence Research*, pp. 369–409, 1995.

[211] P. D. Turney, "Expressing implicit semantic relations without supervision," in *ACL*, 2006.

[212] V. S. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: Requirements and a survey of the state of the art," *Journal of Web Semantics*, vol. 4, pp. 14–28, 2006.

[213] P. Viola and M. Narasimhan, "Learning to extract information from semi-structured text using a discriminative context free grammar," in *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 330–337, USA, New York, NY: ACM, 2005.

[214] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy, "Accelerated training of conditional random fields with stochastic gradient methods," in *ICML*, pp. 969–976, 2006.

[215] M. Wang, "A re-examination of dependency path kernels for relation extraction," in *Proceedings of IJCNLP*, 2008.

[216] Y. Wang and J. Hu, "A machine learning based approach for table detection on the web," in *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, pp. 242–250, ACM, 2002.

[217] B. Wellner, A. McCallum, F. Peng, and M. Hay, "An integrated, conditional model of information extraction and coreference with application to citation matching," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

[218] M. Wick, A. Culotta, and A. McCallum, "Learning field compatibilities to extract database records from unstructured text," in *Proceedings of the*

2006 Conference on Empirical Methods in Natural Language Processing, pp. 603–611, Sydney, Australia: Association for Computational Lingistics, July 2006.

[219] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishing, San Francisco, 1999.

[220] F. Wu and D. S. Weld, "Autonomously semantifying wikipedia," in *CIKM*, pp. 41–50, 2007.

[221] B. Zadrozny and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown," in *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining (KDD)*, 2001.

[222] R. Zanibbi, D. Blostein, and R. Cordy, "A survey of table recognition: Models, observations, transformations, and inferences," *International Journal on Document Analysis and Recognition*, vol. 7, pp. 1–16, 2004.

[223] D. Zelenko, C. Aone, and A. Richardella, "Kernel methods for relation extraction," *Journal of Machine Learning Research*, vol. 3, pp. 1083–1106, 2003.

[224] M. Zhang, J. Zhang, J. Su, and G. Zhou, "A composite kernel to extract relations between entities with both flat and structured features," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 825–832, Sydney, Australia: Association for Computational Linguistics, July 2006.

[225] S. Zhao and R. Grishman, "Extracting relations with integrated information using kernel methods," in *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 419–426, 2005.

[226] G. Zhu, T. J. Bethea, and V. Krishna, "Extracting relevant named entities for automated expense reimbursement," in *KDD*, pp. 1004–1012, 2007.