Chapter 2

INFORMATION EXTRACTION FROM TEXT

Jing Jiang
Singapore Management University
jingjiang@smu.edu.sg

Abstract Information extraction is the task of finding structured information from unstructured or semi-structured text. It is an important task in text mining and has been extensively studied in various research communities including natural language processing, information retrieval and Web mining. It has a wide range of applications in domains such as biomedical literature mining and business intelligence. Two fundamental tasks of information extraction are named entity recognition and relation extraction. The former refers to finding names of entities such as people, organizations and locations. The latter refers to finding the semantic relations such as `FounderOf` and `HeadquarteredIn` between entities. In this chapter we provide a survey of the major work on named entity recognition and relation extraction in the past few decades, with a focus on work from the natural language processing community.

Keywords: Information extraction, named entity recognition, relation extraction

1. Introduction

Information extraction from text is an important task in text mining. The general goal of information extraction is to discover structured information from unstructured or semi-structured text. For example, given the following English sentence,

In 1998, Larry Page and Sergey Brin founded Google Inc.

we can extract the following information,

`FounderOf(`*Larry Page*`,` *Google Inc.*`),`
`FounderOf(`*Sergey Brin*`,` *Google Inc.*`),`
`FoundedIn(`*Google Inc.*`,` *1998*`).`

Such information can be directly presented to an end user, or more commonly, it can be used by other computer systems such as search engines and database management systems to provide better services to end users.

Information extraction has applications in a wide range of domains. The specific type and structure of the information to be extracted depend on the need of the particular application. We give some example applications of information extraction below:

- Biomedical researchers often need to sift through a large amount of scientific publications to look for discoveries related to particular genes, proteins or other biomedical entities. To assist this effort, simple search based on keyword matching may not suffice because biomedical entities often have synonyms and ambiguous names, making it hard to accurately retrieve relevant documents. A critical task in biomedical literature mining is therefore to automatically identify mentions of biomedical entities from text and to link them to their corresponding entries in existing knowledge bases such as the FlyBase.

- Financial professionals often need to seek specific pieces of information from news articles to help their day-to-day decision making. For example, a finance company may need to know all the company takeovers that take place during a certain time span and the details of each acquisition. Automatically finding such information from text requires standard information extraction technologies such as named entity recognition and relation extraction.

- Intelligence analysts review large amounts of text to search for information such as people involved in terrorism events, the weapons used and the targets of the attacks. While information retrieval technologies can be used to quickly locate documents that describe terrorism events, information extraction technologies are needed to further pinpoint the specific information units within these documents.

- With the fast growth of the Web, search engines have become an integral part of people's daily lives, and users' search behaviors are much better understood now. Search based on bag-of-word representation of documents can no longer provide satisfactory results. More advanced search problems such as entity search, structured search and question answering can provide users with better search experience. To facilitate these search capabilities, information ex-

Terrorism Template	
Slot	**Fill Value**
Incident: Date	07 Jan 90
Incident: Location	Chile: Molina
Incident: Type	robbery
Incident: Stage of execution	accomplished
Incident: Instrument type	gun
Human Target: Name	"Enrique Ormazabal Ormazabal"
Human Target: Description	"Businessman": "Enrique Ormazabal Ormazabal"
Human Target: Type	civilian: "Enrique Ormazabal Ormazabal"
Human Target: Number	1: "Enrique Ormazabal Ormazabal"
...	...

A Sample Document

Santiago, 10 Jan 90 – Police are carrying out intensive operations in the town of Molina in the seventh region in search of a gang of alleged extremists who could be linked to a recently discovered arsenal. It has been reported that Carabineros in Molina raided the house of of 25-year-old worker Mario Munoz Pardo, where they found a fal rifle, ammunition clips for various weapons, detonators, and material for making explosives.

It should be recalled that a group of armed individuals wearing ski masks robbed a businessman on a rural road near Molina on 7 January. The businessman, Enrique Ormazabal Ormazabal, tried to resist; The men shot him and left him seriously wounded. He was later hospitalized in Curico. Carabineros carried out several operations, including the raid on Munoz' home. The police are continuing to patrol the area in search of the alleged terrorist command.

Figure 2.1. Part of the terrorism template used in MUC-4 and a sample document that contains a terrorism event.

traction is often needed as a preprocessing step to enrich document representation or to populate an underlying database.

While extraction of structured information from text dates back to the '70s (e.g. DeJong's FRUMP program [28]), it only started gaining much attention when DARPA initiated and funded the Message Understanding Conferences (MUC) in the '90s [33]. Since then, research efforts on this topic have not declined. Early MUCs defined information extraction as filling a predefined template that contains a set of predefined slots. For example, Figure 2.1 shows a subset of the slots in the terrorism template used in MUC-4 and a sample document from which template slot fill values were extracted. Some of the slot fill values such as *"Enrique Ormazabal Ormazabal"* and *"Businessman"* were extracted directly from the text while others such as *robbery*, *accomplished* and *gun* were selected from a predefined value set for the corresponding slot based on the document.

Template filling is a complex task and systems developed to fill one template cannot directly work for a different template. In MUC-6, a number of template-independent subtasks of information extraction were defined [33]. These include named entity recognition, coreference resolution and relation extraction. These tasks serve as building blocks to support full-fledged, domain-specific information extraction systems.

Early information extraction systems such as the ones that participated in the MUCs are often rule-based systems (e.g. [32, 42]). They use linguistic extraction patterns developed by humans to match text and locate information units. They can achieve good performance on the specific target domain, but it is labor intensive to design good extraction rules, and the developed rules are highly domain dependent. Realizing the limitations of these manually developed systems, researchers turned to statistical machine learning approaches. And with the decomposition of information extraction systems into components such as named entity recognition, many information extraction subtasks can be transformed into classification problems, which can be solved by standard supervised learning algorithms such as support vector machines and maximum entropy models. Because information extraction involves identifying segments of text that play different roles, sequence labeling methods such as hidden Markov models and conditional random fields have also been widely used.

Traditionally information extraction tasks assume that the structures to be extracted, e.g. the types of named entities, the types of relations, or the template slots, are well defined. In some scenarios, we do not know in advance the structures of the information we would like to extract and would like to mine such structures from large corpora. For example, from a set of earthquake news articles we may want to automatically discover that the date, time, epicenter, magnitude and casualty of an earthquake are the most important pieces of information reported in news articles. There have been some recent studies on this kind of unsupervised information extraction problems but overall work along this line remains limited.

Another new direction is open information extraction, where the system is expected to extract *all* useful entity relations from a large, diverse corpus such as the Web. The output of such systems includes not only the arguments involved in a relation but also a description of the relation extracted from the text. Recent advances in this direction include systems like TextRunner [6], Woe [66] and ReVerb [29].

Information extraction from semi-structured Web pages has also been an important research topic in Web mining (e.g. [40, 45, 25]). A major difference of Web information extraction from information extraction studied in natural language processing is that Web pages often contain structured or semi-structured text such as tables and lists, whose extraction relies more on HTML tags than linguistic features. Web information extraction systems are also called *wrappers* and learning such systems is called *wrapper induction*. In this survey we only cover information extraction from purely unstructured natural language text. Readers who are interested in wrapper induction may refer to [31, 20] for in-depth surveys.

In this chapter we focus on the two most fundamental tasks in information extraction, namely, named entity recognition and relation extraction. The state-of-the-art solutions to both tasks rely on statistical machine learning methods. We also discuss unsupervised information extraction, which has not attracted much attention traditionally. The rest of this chapter is organized as follows. Section 2 discusses current approaches to named entity recognition, including rule-based methods and statistical learning methods. Section 3 discusses relation extraction under both a fully supervised setting and a weakly supervised setting. We then discuss unsupervised relation discovery and open information extraction in Section 4. In Section 5 we discuss evaluation of information extraction systems. We finally conclude in Section 6.

2. Named Entity Recognition

A named entity is a sequence of words that designates some real-world entity, e.g. "California," "Steve Jobs" and "Apple Inc." The task of named entity recognition, often abbreviated as NER, is to identify named entities from free-form text and to classify them into a set of predefined types such as *person*, *organization* and *location*. Oftentimes this task cannot be simply accomplished by string matching against pre-compiled gazetteers because named entities of a given entity type usually do not form a closed set and therefore any gazetteer would be incomplete. Another reason is that the type of a named entity can be context-dependent. For example, "JFK" may refer to the person "John F. Kennedy," the location "JFK International Airport," or any other entity sharing the same abbreviation. To determine the entity type for "JFK" occurring in a particular document, its context has to be considered.

Named entity recognition is probably the most fundamental task in information extraction. Extraction of more complex structures such as relations and events depends on accurate named entity recognition as a preprocessing step. Named entity recognition also has many applications apart from being a building block for information extraction. In question answering, for example, candidate answer strings are often named entities that need to be extracted and classified first [44]. In entity-oriented search, identifying named entities in documents as well as in queries is the first step towards high relevance of search results [34, 21].

Although the study of named entity recognition dates back to the early '90s [56], the task was formally introduced in 1995 by the sixth Message Understanding Conference (MUC-6) as a subtask of information extraction [33]. Since then, NER has drawn much attention in the research community. There have been several evaluation programs on this task, including the Automatic Content Extraction (ACE) program [1], the shared task of the Conference on Natural Language Learning (CoNLL) in 2002 and 2003 [63], and the BioCreAtIvE (Critical Assessment of Information Extraction Systems in Biology) challenge evaluation [2].

The most commonly studied named entity types are *person*, *organization* and *location*, which were first defined by MUC-6. These types are general enough to be useful for many application domains. Extraction of expressions of dates, times, monetary values and percentages, which was also introduced by MUC-6, is often also studied under NER, although strictly speaking these expressions are not named entities. Besides these general entity types, other types of entities are usually defined for specific domains and applications. For example, the GENIA corpus uses a fine-grained ontology to classify biological entities [52]. In online search and advertising, extraction of product names is a useful task.

Early solutions to named entity recognition rely on manually crafted patterns [4]. Because it requires human expertise and is labor intensive to create such patterns, later systems try to automatically learn such patterns from labeled data [62, 16, 23]. More recent work on named entity recognition uses statistical machine learning methods. An early attempt is Nymble, a name finder based on hidden Markov models [10]. Other learning models such as maximum entropy models [22], maximum entropy Markov models [8, 27, 39, 30], support vector machines [35] and conditional random fields [59] have also been applied to named entity recognition.

2.1 Rule-based Approach

Rule-based methods for named entity recognition generally work as follows: A set of rules is either manually defined or automatically learned. Each token in the text is represented by a set of features. The text is then compared against the rules and a rule is fired if a match is found.

A rule consists of a pattern and an action. A pattern is usually a regular expression defined over features of tokens. When this pattern matches a sequence of tokens, the specified action is fired. An action can be labeling a sequence of tokens as an entity, inserting the start or end label of an entity, or identifying multiple entities simultaneously. For example, to label any sequence of tokens of the form "Mr. X" where X is a capitalized word as a person entity, the following rule can be defined:

$$(\texttt{token = }\text{"Mr."}\ \texttt{orthography type = }\mathit{FirstCap}) \rightarrow \texttt{person name}.$$

The left hand side is a regular expression that matches any sequence of two tokens where the first token is "Mr." and the second token has the orthography type *FirstCap*. The right hand side indicates that the matched token sequence should be labeled as a person name.

This kind of rule-based methods has been widely used [4, 62, 16, 61, 23]. Commonly used features to represent tokens include the token itself, the part-of-speech tag of the token, the orthography type of the token (e.g. first letter capitalized, all letters capitalized, number, etc.), and whether the token is inside some predefined gazetteer.

It is possible for a sequence of tokens to match multiple rules. To handle such conflicts, a set of policies has to be defined to control how rules should be fired. One approach is to order the rules in advance so that they are sequentially checked and fired.

Manually creating the rules for named entity recognition requires human expertise and is labor intensive. To automatically learn the rules, different methods have been proposed. They can be roughly categorized into two groups: top-down (e.g. [61]) and bottom-up (e.g. [16, 23]). With either approach, a set of training documents with manually labeled named entities is required. In the top-down approach, general rules are first defined that can cover the extraction of many training instances. However, these rules tend to have low precision. The system then iteratively defines more specific rules by taking the intersections of the more general rules. In the bottom-up approach, specific rules are defined based on training instances that are not yet covered by the existing rule set. These specific rules are then generalized.

2.2 Statistical Learning Approach

More recent work on named entity recognition is usually based on statistical machine learning. Many statistical learning-based named entity recognition algorithms treat the task as a sequence labeling problem. Sequence labeling is a general machine learning problem and has been used to model many natural language processing tasks including part-of-speech tagging, chunking and named entity recognition. It can be formulated as follows. We are given a sequence of observations, denoted as $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. Usually each observation is represented as a

Steve	Jobs	was	a	co-founder	of	Apple	Inc.
`B-PER`	`I-PER`	`O`	`O`	`O`	`O`	`B-ORG`	`I-ORG`

Figure 2.2. An example sentence with NER labels in the BIO notation. `PER` stands for person and `ORG` stands for organization.

feature vector. We would like to assign a label y_i to each observation x_i. While one may apply standard classification to predict the label y_i based solely on x_i, in sequence labeling, it is assumed that the label y_i depends not only on its corresponding observation x_i but also possibly on other observations and other labels in the sequence. Typically this dependency is limited to observations and labels within a close neighborhood of the current position i.

To map named entity recognition to a sequence labeling problem, we treat each word in a sentence as an observation. The class labels have to clearly indicate both the boundaries and the types of named entities within the sequence. Usually the BIO notation, initially introduced for text chunking [55], is used. With this notation, for each entity type `T`, two labels are created, namely, `B-T` and `I-T`. A token labeled with `B-T` is the beginning of a named entity of type `T` while a token labeled with `I-T` is inside (but not the beginning of) a named entity of type `T`. In addition, there is a label `O` for tokens outside of any named entity. Figure 2.2 shows an example sentence and its correct NER label sequence.

2.2.1 Hidden Markov Models. In a probabilistic framework, the best label sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ for an observation sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is the one that maximizes the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$, or equivalently, the one that maximizes the joint probability $p(\boldsymbol{x}, \boldsymbol{y})$. One way to model the joint probability is to assume a Markov process where the generation of a label or an observation is dependent only on one or a few previous labels and/or observations. If we treat $\boldsymbol{y}$ as hidden states, then we essentially have a hidden Markov model [54].

An example is the Nymble system developed by BBN, one of the earliest statistical learning-based NER systems [10]. Nymble assumes the following generative process:

(1) Each y_i is generated conditioning on the previous label y_{i-1} and the previous word x_{i-1}.

(2) If x_i is the first word of a named entity, it is generated conditioning on the current and the previous labels, i.e. y_i and y_{i-1}.

(3) If x_i is inside a named entity, it is generated conditioning on the previous observation x_{i-1}.

For subsequences of words outside of any named entity, Nymble treats them as a NOT-A-NAME class. Nymble also assumes that there is a magical `+end+` word at the end of each named entity and models the probability of a word being the final word of a named entity. With the generative process described above, the probability $p(\boldsymbol{x}, \boldsymbol{y})$ can be expressed in terms of various conditional probabilities.

Initially x_i is simply the word at position i. Nymble further augments it into $x_i = \langle w, f \rangle_i$, where w is the word at position i and f is a word feature characterizing w. For example, the feature `FourDigitNum` indicates that the word is a number with four digits. The rationale behind introducing word features is that these features may carry strong correlations with entity types. For example, a four-digit number is likely to be a year.

The model parameters of Nymble are essentially the various multinomial distributions that govern the generation of x_i and y_i. Nymble uses supervised learning to learn these parameters. Given sentences labeled with named entities, Nymble performs maximum likelihood estimation to find the model parameters that maximize $p(\boldsymbol{X}, \boldsymbol{Y})$ where $\boldsymbol{X}$ denotes all the sentences in the training data and $\boldsymbol{Y}$ denotes their true label sequences. Parameter estimation essentially becomes counting. For example,

$$p(y_i = \mathtt{c}_1 | y_{i-1} = \mathtt{c}_2, x_{i-1} = \mathtt{w}) = \frac{c(\mathtt{c}_1, \mathtt{c}_2, \mathtt{w})}{c(\mathtt{c}_2, \mathtt{w})}, \qquad (2.1)$$

where $\mathtt{c}_1$ and $\mathtt{c}_2$ are two class labels and $\mathtt{w}$ is a word. $p(y_i = \mathtt{c}_1 | y_{i-1} = \mathtt{c}_2, x_{i-1} = \mathtt{w})$ is the probability of observing the class label $\mathtt{c}_1$ given that the previous class label is $\mathtt{c}_2$ and the previous word is $\mathtt{w}$. $c(\mathtt{c}_1, \mathtt{c}_2, \mathtt{w})$ is the number of times we observe class label $\mathtt{c}_1$ when the previous class label is $\mathtt{c}_2$ and the previous word is $\mathtt{w}$, and $c(\mathtt{c}_2, \mathtt{w})$ is the number of times we observe the previous class label to be $\mathtt{c}_2$ and the previous word to be $\mathtt{w}$ regardless of the current class label.

During prediction, Nymble uses the learned model parameters to find the label sequence $\boldsymbol{y}$ that maximizes $p(\boldsymbol{x}, \boldsymbol{y})$ for a given $\boldsymbol{x}$. With the Markovian assumption, dynamic programming can be used to efficiently find the best label sequence.

2.2.2 Maximum Entropy Markov Models. The hidden Markov models described above are generative models. In general, researchers have found that when training data is sufficient, compared with generative models that model $p(x|y)$, discriminative models that directly model $p(y|x)$ tend to give a lower prediction error rate and thus are preferable [65]. For named entity recognition, there has also been such a shift from generative models to discriminative models. A commonly used discriminative model for named entity recognition is the maximum entropy model [9] coupled with a Markovian assumption. Existing work using such a model includes [8, 27, 39, 30].

Specifically, with a Markovian assumption, the label y_i at position i is dependent on the observations within a neighborhood of position i as well as a number of previous labels:

$$p(\boldsymbol{y}|\boldsymbol{x}) \quad = \quad \prod_i p(y_i|\boldsymbol{y}_{i-k}^{i-1}, \boldsymbol{x}_{i-l}^{i+l}). \tag{2.2}$$

In the equation above, $\boldsymbol{y}_{i-k}^{i-1}$ refers to $(y_{i-k}, y_{i-k+1}, \ldots, y_{i-1})$ and $\boldsymbol{x}_{i-l}^{i+l}$ refers to $(x_{i-l}, x_{i-l+1}, \ldots, x_{i+l})$. And with maximum entropy models, the functional form of $p(y_i|\boldsymbol{y}_{i-k}^{i-1}, \boldsymbol{x}_{i-l}^{i+l})$ follows an exponential model:

$$p(y_i|\boldsymbol{y}_{i-k}^{i-1}, \boldsymbol{x}_{i-l}^{i+l}) \quad = \quad \frac{\exp\left(\sum_j \lambda_j f_j(y_i, \boldsymbol{y}_{i-k}^{i-1}, \boldsymbol{x}_{i-l}^{i+l})\right)}{\sum_{y'} \exp\left(\sum_j \lambda_j f_j(y', \boldsymbol{y}_{i-k}^{i-1}, \boldsymbol{x}_{i-l}^{i+l})\right)}. \tag{2.3}$$

In the equation above, $f_j(\cdot)$ is a feature function defined over the current label, the previous k labels as well as $2l+1$ observations surrounding the current observation, and λ_j is the weight for feature f_j. An example feature is below:

$$f(y_i, y_{i-1}, x_i) = \begin{cases} 1 & \text{if } y_{i-1} = \texttt{O} \text{ and } y_i = \texttt{B-PER} \text{ and } \texttt{word}(x_i) = \text{“Mr.”}, \\ 0 & \text{otherwise .} \end{cases}$$

The model described above can be seen as a variant of the maximum entropy Markov models (MEMMs), which were formally introduced by McCallum et al. for information extraction [48].

To train a maximum entropy Markov model, we look for the feature weights $\Lambda = \{\lambda_j\}$ that can maximize the conditional probability $p(\boldsymbol{Y}|\boldsymbol{X})$ where $\boldsymbol{X}$ denotes all the sentences in the training data and $\boldsymbol{Y}$ denotes their true label sequences. Just like for standard maximum entropy models, a number of optimization algorithms can be used to train maximum entropy Markov models, including Generalized Iterative Scaling (GIS), Improved Iterative Scaling (IIS) and limited memory quasi-Newton methods such as L-BFGS [15]. A comparative study of these optimization methods for maximum entropy models can be found in [46]. L-BFGS is a commonly used method currently.

2.2.3 Conditional Random Fields. Conditional random fields (CRFs) are yet another popular discriminative model for sequence labeling. They were introduced by Lafferty et al. to also address information extraction problems [41]. The major difference between CRFs

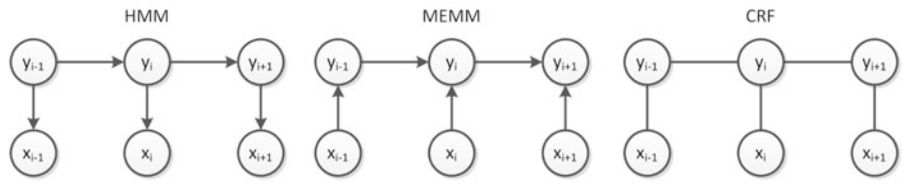

Figure 2.3. Graphical representations of linear-chain HMM, MEMM and CRF.

and MEMMs is that in CRFs the label of the current observation can depend not only on previous labels but also on future labels. Also, CRFs are undirected graphical models while both HMMs and MEMMs are directed graphical models. Figure 2.3 graphically depicts the differences between linear-chain (i.e. first-order) HMM, MEMM and CRF. Ever since they were first introduced, CRFs have been widely used in natural language processing and some other research areas.

Usually linear-chain CRFs are used for sequence labeling problems in natural language processing, where the current label depends on the previous one and the next one labels as well as the observations. There have been many studies applying conditional random fields to named entity recognition (e.g. [49, 59]). Specifically, following the same notation used earlier, the functional form of $p(\boldsymbol{y}|\boldsymbol{x})$ is as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_i \sum_j \lambda_j f_j(y_i, y_{i-1}, \boldsymbol{x}, i)\right), \tag{2.4}$$

where $Z(\boldsymbol{x})$ is a normalization factor of all possible label sequences:

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} \exp\left(\sum_i \sum_j \lambda_j f_j(y'_i, y'_{i-1}, \boldsymbol{x}, i)\right). \tag{2.5}$$

To train CRFs, again maximum likelihood estimation is used to find the best model parameters that maximize $p(\boldsymbol{Y}|\boldsymbol{X})$. Similar to MEMMs, CRFs can be trained using L-BFGS. Because the normalization factor $Z(\boldsymbol{x})$ is a sum over all possible label sequences for $\boldsymbol{x}$, training CRFs is more expensive than training MEMMs.

In linear-chain CRFs we cannot define long-range features. General CRFs allow long-range features but are too expensive to perform exact inference. Sarawagi and Cohen proposed semi-Markov conditional random fields as a compromise [58]. In semi-Markov CRFs, labels are assigned to segments of the observation sequence $\boldsymbol{x}$ and features can measure properties of these segments. Exact learning and inference on semi-Markov CRFs is thus computationally feasible. Sarawagi and Cohen applied Semi-Markov CRFs to named entity recognition and achieved better performance than standard CRFs.

3. Relation Extraction

Another important task in information extraction is relation extraction. Relation extraction is the task of detecting and characterizing the semantic relations between entities in text. For example, from the following sentence fragment,

Facebook co-founder Mark Zuckerberg

we can extract the following relation,

`FounderOf(`*Mark Zuckerberg*`,` *Facebook*`).`

Much of the work on relation extraction is based on the task definition from the Automatic Content Extraction (ACE) program [1]. ACE focuses on binary relations, i.e. relations between two entities. The two entities involved are also referred to as *arguments*. A set of major relation types and their subtypes are defined by ACE. Examples of ACE major relation types include `physical` (e.g. an entity is physically near another entity), `personal/social` (e.g. a person is a family member of another person), and `employment/affiliation` (e.g. a person is employed by an organization). ACE makes a distinction between relation extraction and relation mention extraction. The former refers to identifying the semantic relation between a pair of entities based on *all* the evidence we can gather from the corpus, whereas the latter refers to identifying individual mentions of entity relations. Because corpus-level relation extraction to a large extent still relies on accurate mention-level relation extraction, in the rest of this chapter we do not make any distinction between these two problems unless necessary.

Various techniques have been proposed for relation extraction. The most common and straightforward approach is to treat the task as a classification problem: Given a pair of entities co-occurring in the same sentence, can we classify the relation between the two entities into one of the predefined relation types? Although it is also possible for relation mentions to cross sentence boundaries, such cases are less frequent and hard to detect. Existing work therefore mostly focuses on relation extraction within sentence boundaries.

There have been a number of studies following the classification approach [38, 71, 37, 18, 19]. Feature engineering is the most critical step of this approach. An extension of the feature-based classification approach is to define kernels rather than features and to apply kernel machines such as support vector machines to perform classification. Kernels defined over word sequences [14], dependency trees [26], dependency paths [13] and parse trees [67, 68] have been proposed.

Both feature-based and kernel-based classification methods require a large amount of training data. Another major line of work on relation extraction is weakly supervised relation extraction from large corpora that does not rely on the availability of manually labeled training data. One approach is the bootstrapping idea to start with a small set of seed examples and iteratively find new relation instances as well as new extraction patterns. Representative work includes the Snowball system [3]. Another approach is distant supervision that makes use of known relation instances from existing knowledge bases such as Freebase [50].

3.1 Feature-based Classification

A typical approach to relation extraction is to treat the task as a classification problem [38, 71, 37, 18, 19]. Specifically, any pair of entities co-occurring in the same sentence is considered a candidate relation instance. The goal is to assign a class label to this instance where the class label is either one of the predefined relation types or *nil* for unrelated entity pairs. Alternatively, a two-stage classification can be performed where at the first stage whether two entities are related is determined and at the second stage the relation type for each related entity pair is determined.

Classification approach assumes that a training corpus exists in which all relation mentions for each predefined relation type have been manually annotated. These relation mentions are used as positive training examples. Entity pairs co-occurring in the same sentence but not labeled are used as negative training examples. Each candidate relation instance is represented by a set of features that are carefully chosen. Standard learning algorithms such as support vector machines and logistic regression can then be used to train relation classifiers.

Feature engineering is a critical step for this classification approach. Researchers have examined a wide range of lexical, syntactic and semantic features. We summarize some of the most commonly used features as follows:

- **Entity features:** Oftentimes the two argument entities, including the entity words themselves and the entity types, are correlated with certain relation types. In the ACE data sets, for example, entity words such as *father*, *mother*, *brother* and *sister* and the `person` entity type are all strong indicators of the `family` relation subtype.

- **Lexical contextual features:** Intuitively the contexts surrounding the two argument entities are important. The simplest way to incorporate evidence from contexts is to use lexical features. For example, if the word *founded* occurs between the two arguments, they are more likely to have the `FounderOf` relation.

- **Syntactic contextual features:** Syntactic relations between the two arguments or between an argument and another word can often be useful. For example, if the first argument is the subject of the verb *founded* and the second argument is the object of the verb *founded*, then one can almost immediately tell that the `FounderOf` relation exists between the two arguments. Syntactic features can be derived from parse trees of the sentence containing the relation instance.

- **Background knowledge:** Chan and Roth studied the use of background knowledge for relation extraction [18]. An example is to make use of Wikipedia. If two arguments co-occur in the same Wikipedia article, the content of the article can be used to check whether the two entities are related. Another example is word clusters. For example, if we can group all names of companies such as *IBM* and *Apple* into the same word cluster, we achieve a level of abstraction higher than words and lower than the general entity type `organization`. This level of abstraction may help extraction of certain relation types such as `Acquire` between two companies.

Jiang and Zhai proposed a framework to organize the features used for relation extraction such that a systematic exploration of the feature space can be conducted [37]. Specifically, a relation instance is represented as a labeled, directed graph $G = (V, E, A, B)$, where V is the set of nodes in the graph, E is the set of directed edges in the graph, and A and B are functions that assign labels to the nodes.

First, for each node $v \in V$, $A(v) = \{a_1, a_2, \ldots, a_{|A(v)|}\}$ is a set of attributes associated with node v, where $a_i \in \Sigma$, and Σ is an alphabet that contains all possible attribute values. For example, if node v represents a token, then $A(v)$ can include the token itself, its morphological base form, its part-of-speech tag, etc. If v also happens to be the head word of arg_1 or arg_2, then $A(v)$ can also include the entity type. Next, function $B : V \to \{0, 1, 2, 3\}$ is introduced to distinguish argument nodes from non-argument nodes. For each node $v \in V$, $B(v)$ indicates how node v is related to arg_1 and arg_2. 0 indicates that v does not cover any argument, 1 or 2 indicates that v covers arg_1 or arg_2, respectively, and 3 indicates that v covers both arguments. In a constituency parse tree, a

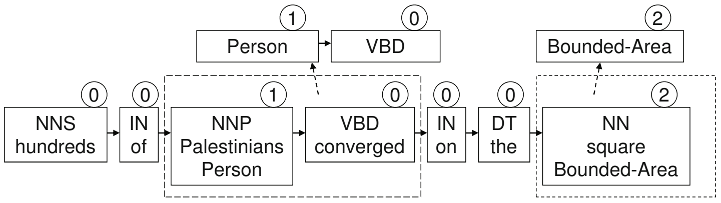

Figure 2.4. An example sequence representation. The subgraph on the left represents a bigram feature. The subgraph on the right represents a unigram feature that states the entity type of arg_2.

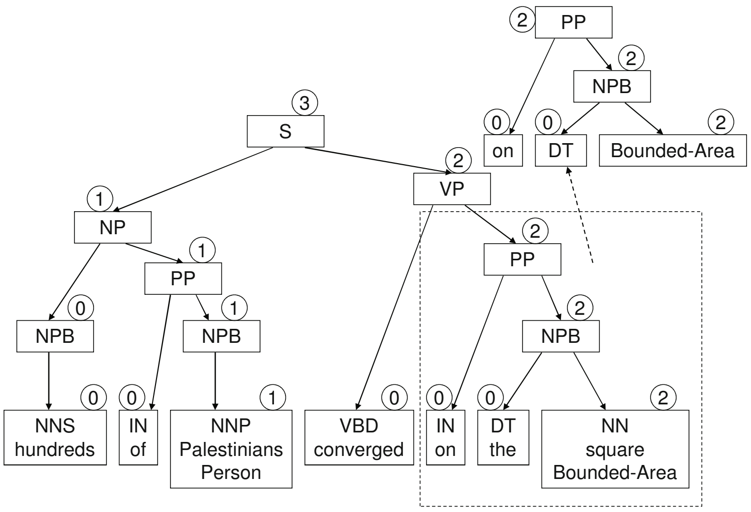

Figure 2.5. An example constituency parse tree representation. The subgraph represents a subtree feature (grammar production feature).

node v may represent a phrase and it can possibly cover both arguments. Figures 2.4, 2.5 and 2.6 show three relation instance graphs based on the token sequence, the constituency parse tree and the dependency parse tree, respectively.

Given the above definition of relation instance graphs, a feature of a relation instance captures part of the attributive and/or structural properties of the relation instance graph. Therefore, it is natural to define a feature as a subgraph of the relation instance graph. Formally, given a graph $G = (V, E, A, B)$, which represents a single relation instance, a feature that exists in this relation instance is a subgraph $G' = (V', E', A', B')$ that satisfies the following conditions: $V' \subseteq V$, $E' \subseteq E$, and $\forall v \in V', A'(v) \subseteq A(v), B'(v) = B(v)$.

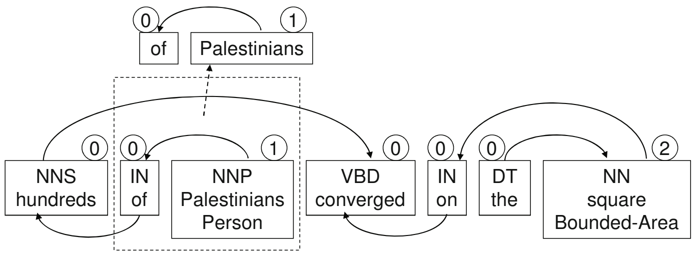

Figure 2.6. An example dependency parse tree representation. The subgraph represents a dependency relation feature between arg_1 *Palestinians* and *of*.

It can be shown that many features that have been explored in previous work on relation extraction can be transformed into this graphic representation. Figures 2.4, 2.5 and 2.6 show some examples.

This framework allows a systematic exploration of the feature space for relation extraction. To explore the feature space, Jiang and Zhai considered three levels of small unit features in increasing order of their complexity: unigram features, bigram features and trigram features. They found that a combination of features at different levels of complexity and from different sentence representations, coupled with task-oriented feature pruning, gave the best performance.

3.2 Kernel Methods

An important line of work for relation extraction is kernel-based classification. In machine learning, a kernel or kernel function defines the inner product of two observed instances represented in some underlying vector space. It can also be seen as a similarity measure for the observations. The major advantage of using kernels is that observed instances do not need to be explicitly mapped to the underlying vector space in order for their inner products defined by the kernel to be computed. We will use the convolution tree kernel to illustrate this idea below.

There are generally three types of kernels for relation extraction: sequence-based kernels, tree-based kernels and composite kernels.

3.2.1 Sequence-based Kernels. Bunescu and Mooney defined a simple kernel based on the shortest dependency paths between two arguments [13]. Two dependency paths are similar if they have the same length and they share many common nodes. Here a node can be represented by the word itself, its part-of-speech tag, or its entity type. Thus the two dependency paths "protestors $\rightarrow$ seized $\leftarrow$ stations" and "troops $\rightarrow$ raided $\leftarrow$ churches" have a non-zero similarity value because they can both be represented as "Person $\rightarrow$ VBD $\leftarrow$ Facility," although they do not share any common word. A limitation of this kernel is that any two dependency paths with different lengths have a zero similarity.

In [14], Bunescu and Mooney introduced a subsequence kernel where the similarity between two sequences is defined over their similar subsequences. Specifically, each node in a sequence is represented by a feature vector and the similarity between two nodes is the inner product of their feature vectors. The similarity between two subsequences of the same length is defined as the product of the similarities of each pair of their nodes in the same position. The similarity of two sequences is then defined as a weighted sum of the similarities of all the subsequences of the same length from the two sequences. The weights are introduced to penalize long common subsequences. Bunescu and Mooney tested their subsequence kernel for protein-protein interaction detection.

3.2.2 Tree-based Kernels. Tree-based kernels use the same idea of using common substructures to measure similarities. Zelenko et al. defined a kernel on the constituency parse trees of relation instances [67]. The main motivation is that if two parse trees share many common subtree structures then the two relation instances are similar to each other. Culotta and Sorensen extended the idea to dependency parse trees [26]. Zhang et al. [68] further applied the convolution tree kernel initially proposed by Collins and Duffy [24] to relation extraction. This convolution tree kernel-based method was later further improved by Qian et al. [53] and achieved a state-of-the-art performance of around 77% of F-1 measure on the benchmark ACE 2004 data set.

We now briefly discuss the convolution tree kernels. As we explained earlier, a kernel function corresponds to an underlying vector space in which the observed instances can be represented. For convolution tree kernels, each dimension of this underlying vector space corresponds to a subtree. To map a constituency parse tree to a vector in this vector space, we simply enumerate all the subtrees contained in the parse tree. If a subtree i occurs k times in the parse tree, the value for the dimension corresponding to i is set to k. Only subtrees containing complete grammar production rules are considered. Figure 2.7 shows an example parse tree and all the subtrees under the NP "the company."

Formally, given two constituency parse trees T_1 and T_2, the convolution tree kernel K is defined as follows:

$$K(T_1, T_2) = \sum_{n_1 \in \mathcal{N}_1} \sum_{n_2 \in \mathcal{N}_2} \sum_{i} I_i(n_1) I_i(n_2). \tag{2.6}$$

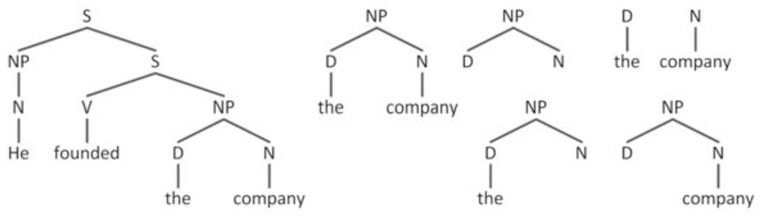

Figure 2.7. Left: The constituency parse tree of a simple sentence. Right: All the subtrees of the NP “the company” considered in convolution tree kernels.

Here $\mathcal{N}_1$ and $\mathcal{N}_2$ are the sets of all nodes in T_1 and T_2 respectively. i denotes a subtree in the feature space. $I_i(n)$ is 1 if subtree i is seen rooted at node n and 0 otherwise.

It is not efficient to directly compute K as defined in Equation 2.6. Instead, we can define $C(n_1, n_2) = \sum_i I_i(n_1)I_i(n_2)$. $C(n_1, n_2)$ can then be computed in polynomial time based on the following recursive property:

- If the grammar productions at n_1 and n_2 are different, then the value of $C(n_1, n_2)$ is 0.

- If the grammar productions at n_1 and n_2 are the same and n_1 and n_2 are pre-terminals, then $C(n_1, n_2)$ is 1. Here pre-terminals are nodes directly above words in a parse tree, e.g. the `N`, `V` and `D` in Figure 2.7.

- If the grammar productions at n_1 and n_2 are the same and n_1 and n_2 are not pre-terminals,

$$C(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j))), \qquad (2.7)$$

  where $nc(n)$ is the number of child-nodes of n, and $ch(n, j)$ is the j-th child-node of n. Note that here $nc(n_1) = nc(n_2)$.

With this recursive property, convolution tree kernels can be efficiently computed in $O(|\mathcal{N}_1||\mathcal{N}_2|)$ time.

3.2.3 Composite Kernels. It is possible to combine different kernels into a composite kernel. This is when we find it hard to include all the useful features into a single kernel. Zhao and Grishman defined several syntactic kernels such as argument kernel and dependency path kernel before combing them into a composite kernel [70]. Zhang et al. combined an entity kernel with the convolution tree kernel to form a composite kernel [69].

3.3 Weakly Supervised Learning Methods

Both feature-based and kernel-based classification methods for relation extraction rely on a large amount of training data, which is expensive to obtain. A solution to this problem is weakly supervised learning methods that work with much less training data. The most notable weakly supervised method for relation extraction is bootstrapping, which starts from a small set of seed relation instances and iteratively learns more relation instances and extraction patterns. It has been widely explored [12, 3]. More recently, another learning paradigm called distant supervision has been proposed to make use of a large number of known relation instances in existing large knowledge bases to create training data [50]. For both bootstrapping and distant supervision, noisy training data is automatically generated. To achieve good performance, careful feature selection and pattern filtering need to be carried out.

3.3.1 Bootstrapping. A representative work on bootstrapping for relation extraction is the Snowball system developed by Agichtein and Gravano [3], which improved over an earlier system called DIPRE developed by Brin [12]. The idea behind Snowball is simple. We start with a set of seed entity pairs that are related through the target relation. For example, if the target relation is `HeadquarteredIn`, we may use seed pairs such as $\langle \textit{Microsoft}, \textit{Redmond} \rangle$, $\langle \textit{Google}, \textit{Mountain View} \rangle$ and $\langle \textit{Facebook}, \textit{Palo Alto} \rangle$. Given a large corpus, we then look for co-occurrences of these entity pairs within close proximity. The assumption is that if two entities related through the target relation co-occur closely, the context in which they co-occur is likely to be a pattern for the target relation. For example, we may find sentence fragments such as "Google's headquarters in Mountain View" and "Redmond-based Microsoft" and extract patterns like "`ORG`*'s headquarters in* `LOC`" and "`LOC`-*based* `ORG`." With these patterns, we can search the corpus and find more $\langle \texttt{ORG}, \texttt{LOC} \rangle$ entity pairs that have the `HeadquarteredIn` relation. We add these entity pairs to the set of seed relation instances and repeat the process. More patterns and entity pairs are added to the results until a certain condition is satisfied.

An important step in bootstrapping methods is to evaluate the quality of extraction patterns so as not to include many noisy patterns during the extraction process. For example, from the seed entity pair $\langle \textit{Google}, \textit{Mountain View} \rangle$ we may also find "Google, Mountain View" in the corpus. However, the pattern "`ORG`*,* `LOC`" is not a reliable one and thus should not be used. Heuristic methods have been proposed to judge the quality of an extraction pattern. Usually two factors are considered, coverage and precision. Coverage is related to the percentage of true relation instances that can be discovered by the pattern. Precision is related to the percentage of correct relation instances among all the relation instances discovered by the pattern.

3.3.2 Distant Supervision. In bootstrapping only a small set of seed entity pairs is used. With the growth of the social Web, much human knowledge has been contributed by a large crowd of users and stored in knowledge bases. A well-known example is Wikipedia. Another example is Freebase, a knowledge base that stores structured human knowledge such as entity relations [11]. With such freely available knowledge, it becomes possible to use a large set of entity pairs known to have a target relation to generate training data. Mintz et al. proposed distant supervision for relation extraction based on this idea [50]. They assume that if two entities participate in a relation, any sentence that contain these two entities express that relation. Because this assumption does not always hold, Mintz et al. use features extracted from different sentences containing the entity pair to create a richer feature vector that is supposed to be more reliable. They define lexical, syntactic and named entity tag features. They use standard multi-class logistic regression as the classification algorithm. Their experiments show that this method can reach almost 70% of precision based on human judgment. Nguyen and Moschitti further used knowledge from both YAGO and Wikipedia documents for distant supervision and achieved around 74% F-1 measure [51].

4. Unsupervised Information Extraction

In Section 2 and Section 3, we discussed named entity recognition and relation extraction where the entity types and relation types are well defined in advance based on the application. A large amount of labeled training data is also required in order to learn a good named entity recognizer or relation extractor. However, both defining the structures for the information to be extracted and annotating documents according to the defined structures require human expertise and are time consuming. To alleviate this problem, recently there has been an increasing amount of interest in unsupervised information extraction from large corpora.

In this section we review some recent studies along this line. We first discuss relation discovery and template induction where the goal is to discover salient relation types or templates for a given domain. The key idea is to cluster entities or entity pairs based on their lexico-syntactic contextual features. We then discuss open information extraction where the goal is to extract *any* type of relation from a large, diverse corpus such as the Web.

4.1 Relation Discovery and Template Induction

In Section 3 we discussed relation extraction when the types of relations to be extracted are known in advance. There are also cases where we do not have any specific relation types in mind but would like to discover salient relation types from a given corpus. For example, given a set of articles reporting hurricane events, it would be useful if we could automatically discover that one of the most important relations for this domain is the `hit` relation between a hurricane and the place being hit.

Shinyama and Sekine first proposed to study this problem, which they referred to as Unrestricted Relation Discovery [60]. They started by collecting a large number of news articles from different news sources on the Web. They then used simple clustering based on lexical similarity to find articles talking about the same event. In this way they could enrich the feature representation of an entity using its multiple occurrences in different articles. Next they performed syntactic parsing and extracted named entities from these articles. Each named entity could then be represented by a set of syntactic patterns as its features. For example, a pattern may indicate that the entity is the subject of the verb *hit*. Finally, they clustered pairs of entities co-occurring in the same article using their feature representations. The end results were tables in which rows corresponded to different articles and columns corresponded to different roles in a relation. They were able to achieve around 75% of accuracy for the discovered tables.

Rosenfeld and Feldman formulated unsupervised relation discovery in a more general way [57]. It is assumed that the input of the problem consists of entity pairs together with their contexts. An unsupervised relation discovery algorithm clusters these entity pairs into disjoint groups where each group represents a single semantic relation. There is also a garbage cluster to capture unrelated entity pairs or unimportant relations. The contexts for each entity pair consist of the contexts of each entity and the contexts of the two entities' co-occurrences. An entity pair can be represented by a set of features derived from the contexts. Rosenfeld and Feldman considered only surface pattern features. For example, "$\texttt{arg}_1$, based in $\texttt{arg}_2$" is a pattern to capture a co-occurrence context between the two entities. For clustering, Rosenfeld and Feldman considered hierarchical agglomerative clustering and K-means clustering. Their method was able to discover relations such as `CityOfState` and `EmployedIn`.

While relation discovery considers binary relations only, a more complex task is to automatically induce an information extraction template, which may contain multiple slots playing different semantic roles. The most straightforward solution is to identify candidates of role fillers first and then cluster these candidates into clusters. However, this simplified clustering approach does not consider an important observation, which is that a single document tends to cover different slots. To remedy this problem, Marx et al. proposed a cross-component clustering algorithm for unsupervised information extraction [47]. The algorithm assigns a candidate from a document to a cluster based on the candidate's feature similarity with candidates from *other documents* only. In other words, the algorithm prefers to separate candidates from the same document into different clusters. Leung et al. proposed a generative model to capture the same intuition [43]. Specifically, they assume a prior distribution over the cluster labels of candidates in the same document where the prior prefers a diversified label assignment. Their experiments show that clustering results are better with this prior than without using the prior.

The aforementioned two studies assume a single template and do not automatically label the discovered slots. Chambers and Jurafsky presented a complete method that is able to discover multiple templates from a corpus and give meaningful labels to discovered slots [17]. Specifically, their method performs two steps of clustering where the first clustering step groups lexical patterns that are likely to describe the same type of events and the second clustering step groups candidate role fillers into slots for each type of events. A slot can be labeled using the syntactic patterns of the corresponding slot fillers. For example, one of the slots discovered by their method for the bombing template is automatically labeled as "Person/Organization who raids, questions, discovers, investigates, diffuses, arrests." A human can probably infer from the description that this refers to the `police` slot.

4.2 Open Information Extraction

Relation discovery and template induction usually work on a corpus from a single domain, e.g. articles describing terrorism events, because the goal is to discover the most salient relations from such a domain-specific corpus. In some cases, however, our goal is to find all the potentially useful facts from a large and diverse corpus such as the Web. This is the focus of open information extraction, first introduced by Banko et al. [6].

Open information extraction does not assume any specific target relation type. It makes a single pass over the corpus and tries to extract as many relations as possible. Because no relation type is specified in advance, part of the extraction results is a phrase that describes the relation extracted. In other words, open information extraction generates $\langle \texttt{arg}_1, \texttt{rel}, \texttt{arg}_2 \rangle$ tuples.

In [7], Banko and Etzioni introduced an unlexicalized CRF-based method for open information extraction. The method is based on the observation that although different relation types have very different semantic meanings, there exists a small set of syntactic patterns that cover the majority of semantic relation mentions. It is therefore possible to train a relation extraction model that extracts arbitrary relations. The key is not to include lexical features in the model.

Later work on open information extraction introduced more heuristics to improve the quality of the extracted relations. In [29], for example, Fader et al. proposed the following two heuristics: (1) A multi-word relation phrase must begin with a verb, end with a preposition, and be a contiguous sequence of words in the sentence. (2) A binary relation phrase ought to appear with at least a minimal number of distinct argument pairs in a large corpus. It is found that the two heuristics can effectively lead to better extraction results.

5. Evaluation

To evaluate information extraction systems, manually annotated documents have to be created. For domain-specific information extraction systems, the annotated documents have to come from the target domain. For example, to evaluate gene and protein name extraction, biomedical documents such as PubMed abstracts are used. But if the purpose is to evaluate general information extraction techniques, standard benchmark data sets can be used. Commonly used evaluation data sets for named entity recognition include the ones from MUC [33], CoNLL-2003 [63] and ACE [1]. For relation extraction, ACE data sets are usually used.

The typical evaluation metrics for information extraction are precision, recall and F-1 scores. Precision measures the percentage of correct instances among the identified positive instances. Recall measures the percentage of correct instances that can be identified among all the positive instances. F-1 is the geometric mean of precision and recall.

For named entity recognition, strictly speaking a correctly identified named entity must satisfy two criteria, namely, correct entity boundary and correct entity type. Most evaluation is based on the exact match of entity boundaries. However, it is worth nothing that in some cases credit should also be given to partial matches, e.g. when the goal is only to tell whether an entity is mentioned in a document or a sentence [64].

For relation extraction, as we have mentioned, there are two levels of extraction, corpus-level and mention-level. While evaluation at mention level requires annotated relation mention instances, evaluation at corpus level requires only truly related entity pairs, which may be easier to obtain or annotate than relation mentions.

Currently, the state-of-the-art named entity recognition methods can achieve around 90% of F-1 scores when trained and tested on the same domain [63]. It is generally observed that person entities are easier to extract, followed by locations and then organizations. It is important to note that when there is domain change, named entity recognition performance can drop substantially. There have been several studies addressing the domain adaptation problem for named entity recognition (e.g. [36, 5]).

For relation extraction, the state-of-the-art performance is lower than that of named entity recognition. On the ACE 2004 benchmark data set, for example, the best F-1 score is around 77% for the seven major relation types [53].

6. Conclusions and Summary

Information extraction is an important text mining problem and has been extensively studied in areas such as natural language processing, information retrieval and Web mining. In this chapter we reviewed some representative work on information extraction, in particular work on named entity recognition and relation extraction. Named entity recognition aims at finding names of entities such as people, organizations and locations. State-of-the-art solutions to named entity recognition rely on statistical sequence labeling algorithms such as maximum entropy Markov models and conditional random fields. Relation extraction is the task of finding the semantic relations between entities from text. Current state-of-the-art methods use carefully designed features or kernels and standard classification to solve this problem.

Although supervised learning has been the dominating approach to information extraction, weakly supervised methods have also drawn much attention. Bootstrapping is a major technique for semi-supervised relation extraction. More recently, with large amounts of knowledge made available in online knowledge bases, distant supervision provides a new paradigm of learning without training data.

Unsupervised information extraction aims to automatically induce the structure of the information to be extracted such as the relation types and the templates. Clustering is the main technique used for unsupervised information extraction.

With the fast growth of textual data on the Web, it is expected that future work on information extraction will need to deal with even more diverse and noisy text. Weakly supervised and unsupervised methods will play a larger role in information extraction. The various user-generated content on the Web such as Wikipedia articles will also become important resources to provide some kind of supervision.

References

[1] Automatic content extraction (ACE) evaluation. http://www.itl.nist.gov/iad/mig/tests/ace/.

[2] BioCreAtIvE. http://www.biocreative.org/.

[3] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM Conference on Digital Libraries*, pages 85–94, 2000.

[4] Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, and Mabry Tyson. FASTUS: A finite-state processor for information extraction from real-world text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

[5] Andrew Arnold, Ramesh Nallapati, and William W. Cohen. Exploiting feature hierarchy for transfer learning in named entity recognition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 245–253, 2008.

[6] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2670–2676, 2007.

[7] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 28–36, 2008.

[8] Oliver Bender, Franz Josef Och, and Hermann Ney. Maximum entropy models for named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, 2003.

[9] Adam L. Bergert, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March 1996.

[10] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In

Proceedings of the 5th Conference on Applied Natural Language Processing, pages 194–201, 1997.

[11] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.

[12] Sergey Brin. Extracting patterns and relations from the World Wide Web. In *Proceedings of the 1998 International Workshop on the Web and Databases*, 1998.

[13] Razvan Bunescu and Raymond Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 724–731, 2005.

[14] Razvan Bunescu and Raymond Mooney. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems 18*, pages 171–178. 2006.

[15] Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Journal of Mathematical Programming*, 63(2):129–156, January 1994.

[16] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference*, pages 328–334, 1999.

[17] Nathanael Chambers and Dan Jurafsky. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 976–986, 2011.

[18] Yee Seng Chan and Dan Roth. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160, 2010.

[19] Yee Seng Chan and Dan Roth. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 551–560, 2011.

[20] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled F. Shaalan. A survey of Web information extraction sys-

tems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, October 2006.

[21] Tao Cheng, Xifeng Yan, and Kevin Chen-Chuan Chang. Supporting entity search: a large-scale prototype search engine. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 1144–1146, 2007.

[22] Hai Leong Chieu and Hwee Tou Ng. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning*, pages 160–163, 2003.

[23] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, pages 1251–1256, 2001.

[24] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 13*. 2001.

[25] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.

[26] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 423–429, 2004.

[27] James R. Curran and Stephen Clark. Language independent NER using a maximum entropy tagger. In *Proceedings of the 7th Conference on Natural Language Learning*, 2003.

[28] Gerald DeJong. Prediction and substantiation: A new approach to natural language processing. *Cognitive Science*, 3:251–173, 1979.

[29] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, 2011.

[30] Jenny Finkel, Shipra Dingare, Christopher D. Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. Exploring the boundaries: gene and protein identification in biomedical text. *BMC Bioinformatics*, 6(Suppl 1)(S5), 2005.

[31] Sergio Flesca, Giuseppe Manco, Elio Masciari, Eugenio Rende, and Andrea Tagarelli. Web wrapper induction: a brief survey. *AI Communications*, 17(2):57–61, April 2004.

[32] Ralph Grishman, John Sterling, and Catherine Macleod. New York University: Description of the PROTEUS system as used for MUC-3. In *Proceedings of the 3rd Message Understadning Conference*, pages 183–190, 1991.

[33] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 466–471, 1996.

[34] Guoping Hu, Jingjing Liu, Hang Li, Yunbo Cao, Jian-Yun Nie, and Jianfeng Gao. A supervised learning approach to entity search. In *Proceedings of the 3rd Asia Information Retrieval Symposium*, pages 54–66, 2006.

[35] Hideki Isozaki and Hideto Kazawa. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th International Conference on Computational Linguistics*, 2002.

[36] Jing Jiang and ChengXiang Zhai. Exploiting domain structure for named entity recognition. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–81, 2006.

[37] Jing Jiang and ChengXiang Zhai. A systematic exploration of the feature space for relation extraction. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 113–120, 2007.

[38] Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pages 178–181, 2004.

[39] Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. Named entity recognition with character-level models. In *Proceedings of the 7th Conference on Natural Language Learning*, 2003.

[40] Nicholas Kushmerick, Daniel S. Weld, and Robert Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.

[41] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and

labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[42] Wendy Lehnert, Claire Cardie, Divid Fisher, Ellen Riloff, and Robert Williams. University of Massachusetts: Description of the CIRCUS system as used for MUC-3. In *Proceedings of the 3rd Message Understadning Conference*, pages 223–233, 1991.

[43] Cane Wing-ki Leung, Jing Jiang, Kian Ming A. Chai, Hai Leong Chieu, and Loo-Nin Teow. Unsupervised information extraction with distributional prior knowledge. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 814–824, 2011.

[44] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, 2002.

[45] Liu Ling, Calton Pu, and Wei Han. XWRAP: An XML-enabled wrapper construction system for Web information sources. In *Proceedings of the 16th International Conference on Data Engineering*, pages 611–621, 2000.

[46] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, 2002.

[47] Zvika Marx, Ido Dagan, and Eli Shamir. Cross-component clustering for template learning. In *Proceedings of the 2002 ICML Workshop on Text Learning*, 2002.

[48] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning*, pages 591–598, 2000.

[49] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th Conference on Natural Language Learning*, 2003.

[50] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, 2009.

[51] Truc Vien T. Nguyen and Alessandro Moschitti. End-to-end relation extraction using distant supervision from external semantic

repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 277–282, 2011.

[52] Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. The GENIA corpus: an annotated research abstract corpus in molecular biology domain. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 82–86, 2002.

[53] Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 697–704, 2008.

[54] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *77*, 77(2):257–286, 1989.

[55] Lance A. Ramshaw and Mitch P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workship on Very Large Corpora*, pages 82–94, 1995.

[56] Lisa F. Rau. Extracting company names from text. In *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications*, pages 29–32, 1991.

[57] Benjamin Rosenfeld and Ronen Feldman. Clustering for unsupervised relation identification. In *Proceedings of the 16th ACM conference on Conference on Information and Knowledge Management*, pages 411–418, 2007.

[58] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17*, pages 1185–1192. 2005.

[59] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, pages 104–107, 2004.

[60] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 304–311, 2006.

[61] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, February 1999.

[62] Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. CRYSTAL inducing a conceptual dictionary. In *Proceed-*

ings of the 14th International Joint Conference on Artificial Intelligence, pages 1314–1319, 1995.

[63] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 142–147, 2003.

[64] Richard Tzong-Han Tsai, Shih-Hung Wu, Wen-Chi Chou, Yu-Chun Lin, Ding He, Jieh Hsiang, Ting-Yi Sung, and Wen-Lian Hsu. Various criteria in the evaluation of biomedical named entity recognition. *BMC Bioinformatics*, 7(92), 2006.

[65] Vladimir Vapnik. *Statistical Learning Theory.* John Wiley & Sons, 2008.

[66] Fei Wu and Daniel S. Weld. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, 2010.

[67] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, February 2003.

[68] Min Zhang, Jie Zhang, and Jian Su. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, 2006.

[69] Min Zhang, Jie Zhang, Jian Su, and GuoDong 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 the 44th Annual Meeting of the Association for Computational Linguistics*, pages 825–832, 2006.

[70] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 419–426, 2005.

[71] GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 427–434, 2005.

Springer

http://www.springer.com/978-1-4614-3222-7

Mining Text Data
(Eds.)C.C. Aggarwal; C. Zhai
2012, XI, 522 p. 37 illus., 10 in color., Hardcover
ISBN: 978-1-4614-3222-7