Chapter VI
Automatic Semantic Annotation Using Machine Learning

Jie Tang
Tsinghua University, Beijing, China

Duo Zhang
University of Illinois, Urbana-Champaign, USA

Limin Yao
Tsinghua University, Beijing, China

Yi Li
Tsinghua University, Beijing, China

ABSTRACT

This chapter aims to give a thorough investigation of the techniques for automatic semantic annotation. The Semantic Web provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. However, lack of annotated semantic data is a bottleneck to make the Semantic Web vision a reality. Therefore, it is indeed necessary to automate the process of semantic annotation. In the past few years, there was a rapid expansion of activities in the semantic annotation area. Many methods have been proposed for automating the annotation process. However, due to the heterogeneity and the lack of structure of the Web data, automated discovery of the targeted or unexpected knowledge information still present many challenging research problems. In this chapter, we study the problems of semantic annotation and introduce the state-of-the-art methods for dealing with the problems. We will also give a brief survey of the developed systems based on the methods. Several real-world applications of semantic annotation will be introduced as well. Finally, some emerging challenges in semantic annotation will be discussed.

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.
INTRODUCTION

Semantic annotation of the web documents is the only way to make the Semantic Web vision a reality. The current Semantic Web meets a bottleneck that there is not much of a Semantic Web due to the lack of annotated web pages. There is such a lack that the Semantic Web is still submerged in the sea of the un-meaningful (un-annotated) web pages.

Semantic annotations are to tag ontology class instance data and map it onto ontology classes. Manual annotation is more easily accomplished today, using authoring tools such as OntoMat (Handschuh, Staab, and Ciravegna, 2002) and SHOE (Heflin, Hendler, and Luke, 2003). However, the use of human annotators is often fraught with errors due to factors such as annotator familiarity with the domain, amount of training, and complex schemas. Manual annotation is also expensive and cannot be used to deal with the large volume of the existing documents on the Web. Automatic semantic annotation is an ideal solution to the problem. However, the fully automatic creation of semantic annotations is also an unsolved problem. Hence, semi-automatic creation of annotations is the method mostly used in current systems.

There are many automatic annotation methods have been proposed, including: (A) supervised machine learning based method, (B) unsupervised machine learning based method, and (C) ontology based method.

(A) The supervised machine learning based method consists of two stages: annotation and training. In annotation, we are given a document in either plain text or semi-structured (e.g. emails, web pages, forums, etc.), and the objective is to identify the entities and the semantic relations between the entities. In training, the task is to learn the model(s) that are used in the annotation stage. For learning the models, the input data is often viewed as a sequence of units, for example, a document can be viewed as a sequence of either words or text lines (depending on the specific applications). In the supervised machine learning based method, labeled data for training the model is required.

(B) The unsupervised machine learning based method tries to create the annotation without labeled data. For example, Crescenzi, Mecca, and Merialdo (2001) propose a method for automatically generalizing the extraction patterns from the web pages. The generalized patterns can then be used to extract the data from the Web.

(C) The ontology based method employs the other knowledge sources like thesaurus, ontology, etc. The basic idea is to first construct a pattern-based ontology, and then use the ontology to extract the needed information from the web page. Some systems also utilize the human general knowledge from common sense ontologies such as Cyc (Lenat and Guha, 1990) and WordNet (Fellbaum, 1998).

In this chapter, we will focus on the first topic: how to create semantic annotation by using supervised machine learning. Figure 1 shows our perspective on semantic annotation. It consists of three layers: Theoretical layer, Annotation layer, and Advanced application layer. The bottom layer is the basic theories including machine learning, statistical learning, and natural language processing; based on these theories, the annotation layer (the middle layer) is mainly comprised of four subtasks: entity extraction, relation extraction, relation discovery, and annotation; based on the annotated results (i.e. semantic data), different advanced applications can be developed (the top layer), for example: semantic integration, semantic search, semantic mining, and reasoning. In semantic annotation, by entity extraction, we aim at identifying and pulling out a sub-sequence that we are interested in from a web page. The identified sub-sequence is viewed as an instance (Appelt, 1999; MUC, 1999). By relation extraction, given a pair of entities, the objective is to decide whether a particular relation holds between the entities (ACE, 2003; Culotta and Sorensen, 2004).
Automatic Semantic Annotation Using Machine Learning

By relation discovery, we aim at discovering unknown relations between instances (Grefenstette, 1994; Maedche and Staab, 2000). The discovered relations can again be used to populate the ontology. The task of annotation is to describe the identified entities and relations according to the ontology.

There are still many challenges in this research area. Web pages of different characteristics (its size, redundancy and the lack of semantics in most plain texts) require different kinds of methods (sometimes even vary largely) to deal with. For example, for a template-based web page in which the data may be generated from a database, one may achieve good results using a rule based method; however for a web page containing a large portion of free text, the rule based method might not work well while a classification based method can be more appropriate. The previous methods to entity extraction, such as those applied to the Message Understanding Conferences (MUC) during the 1990s, usually induce extraction rules on small collections of documents. However, the characteristics of the web require more effective algorithms being able to learn more efficiently. Furthermore, new types of web content such as web forums, blogs and wikis (some of them included in the so-called Web 2.0), provide rich data sources for conducting semantic annotation, at the same time, bring big challenges to the field.

The existing machine-learning based approaches rely on the assumption that documents have either the similar structure or the similar content, an assumption which seems unrealistic due to the heterogeneity of the Web.

This chapter tries to give a comprehensive investigation of the methods of automatic entity extraction and relation extraction using supervised machine learning. Specifically, for entity extraction we classify the methods into four categories: rule learning based extraction, classification based extraction, sequential learning based extraction, and non-linear Markov random fields based extraction. For relation extraction, we also classify the methods into four categories: classification based method, kernel based method, sequential labeling based method, and the other methods. All these methods have immediate real-life applications. Semantic annotation has been applied to, for example, social networking data annotation (Dingli, Ciravegna, and Wilks, 2003), researcher profile annotation (Mika, 2005; Tang, Zhang, and Yao, 2007c), Knowledge and Information Management (KIM) (Popov et al., 2003), image annotation (Bloehdorn et al., 2005), and company reports annotation (Tang, Li, Lu, Liang, and Wang, 2005).

In the rest of the chapter, we will describe the state-of-the-art methods for entity extraction and relation extraction. This is followed by a brief
introduction of existing systems based on the methods. We then present several applications to better understand how the methods can be utilized to help businesses. The chapter will have a mix of research and industry flavor, addressing research concepts and looking at the technologies from an industry perspective. After that, we will discuss future research directions on semantic annotation. Finally, we will give the concluding remarks.

METHODOLOGIES

The Semantic Web promises to make web content machine understandable. In this context, one of the most important things is the annotation of the existing Web, called semantic annotation.

In the past years, several conferences, for example Message Understanding Conferences (MUC) and Automatic Content Extraction (ACE) provided a benchmark for evaluating the effectiveness of different automatic content extraction technologies developed to support automatic processing of human language in text form. Recently, Pattern Analysis Statistical Modeling and Computational Learning (PASCAL) Challenge also provides a rigorous evaluation of various machine learning techniques for extracting the information from documents. In both of the contests, the situations can be described as: given a standardized corpus of annotated and pre-processed documents, the participants are expected to perform a number of subtasks, with each examining a different aspect of the learning process (in addition, subtasks will look at the effect of limiting the availability of training data, the ability to select the most appropriate training data (i.e. active learning) and the use of un-annotated data to aid learning).

In this section, we present a survey of the current techniques that can be used to perform automatic entity extraction and relation extraction.

Entity Extraction

Entity extraction, as one of the most important problems in semantic annotation, is aimed at identifying a sub-sequence that we are interested in from the documents like web pages, emails, and PDF files and giving meaning to the identified text. Considerable research work has been conducted for entity extraction. Among these work, rule learning based method, classification based method, and sequential labeling based method are the three state-of-the-art methods. Recently, non-linear Markov random fields also attract much attention, aiming at improving the performance of semantic annotation by incorporating different types of dependencies (e.g. hierarchical laid-out) rather than traditional linear-chain dependencies.

Rule Based Entity Extraction

In this section, we review the rule based algorithms for entity extraction. Numerous extraction systems have been developed based on the method, for instance: AutoSlog (Riloff, 1993), Crystal (Soderland, Fisher, Aseltine, and Lehnert, 1995), (LP)\(^2\) (Ciravegna, 2001), iASA (Tang et al., 2005), Whisk (Soderland, 1999), Rapier (Califf and Mooney, 1998), SRV (Freitag, 1998), WIEN (Kushmerick, Weld, and Doorenbos, 1997), Stalker (Muslea, Minton, and Knoblock, 1998; Muslea, 1999a), and BWI (Freitag and Kushmerick, 2000). See (Muslea, 1999b; Peng, 2001; Siefkes and Siniakov, 2005) for an overview. In general, the methods can again be grouped into two categories: dictionary based method and wrapper induction. We give a detailed introduction in (Tang, Hong, Zhang, Liang, and Li, 2007a). Here we use (LP)\(^2\) (Ciravegna, 2001) as an example to introduce the methods.

(LP)\(^2\) is one of the typical rule based extraction methods, which conducts rule learning in a bottom-up fashion (Ciravegna, 2001). It learns two types of rules that respectively identify the start
boundary and the end boundary of an entity to be extracted. The learning is performed in two steps: initially a set of tagging rules is learned from a user-defined corpus (training data set); then additional rules are induced to correct mistakes in extraction.

Three types of rules are defined in (LP): tagging rules, contextual rules, and correction rules. A tagging rule is composed of a pattern of conditions on a sequence of words and an action of determining whether or not the current position is a boundary of an instance. Table 1 shows an example of the tagging rule. The first column represents a sequence of words. The second to the fifth columns represent Part-Of-Speech, Word type, Lookup results in a dictionary, and Name Entity Recognition results of the word sequence respectively. The last column represents the action. In Table 1, the action “<Speaker>” indicates that if the text match the pattern, the word “Patrick” will be identified as the start boundary of a speaker.

The tagging rules are induced as follows: (1) First, a tag in the training corpus is selected, and a window of \( w \) words to the left and \( w \) words to the right is used as constraints in the initial rule pattern. (2) Then all the initial rules are generalized. The generalization algorithm could be various. For example, based on NLP knowledge, the two rules “at 4 pm” and “at 5 pm” can be generalized to be “at \( \text{DIGIT} \) pm”. Each generalized rule is tested on the training corpus and an error score \( E = \text{wrong/matched} \) is calculated. (3) Finally, the \( k \) best generalizations for each initial rule are kept in a so called best rule pool. This induction algorithm is also used for the other two types of rules discussed below. Table 2 indicates

**Table 1. Example of initial tagging rule**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>; : Punctuation</td>
<td></td>
</tr>
<tr>
<td>Patrick NNP Word Person’s first name Person &lt;Speaker&gt;</td>
<td></td>
</tr>
<tr>
<td>Stroh NNP Word</td>
<td></td>
</tr>
<tr>
<td>, , Punctuation</td>
<td></td>
</tr>
<tr>
<td>assistant NN Word Job title</td>
<td></td>
</tr>
<tr>
<td>professor NN Word</td>
<td></td>
</tr>
<tr>
<td>, , Punctuation</td>
<td></td>
</tr>
<tr>
<td>SDS NNP Word</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2. Example of generalized tagging rule**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>; : Punctuation</td>
<td></td>
</tr>
<tr>
<td>; Word Person’s first name Person &lt;Speaker&gt;</td>
<td></td>
</tr>
<tr>
<td>Word</td>
<td></td>
</tr>
<tr>
<td>Punctuation</td>
<td></td>
</tr>
<tr>
<td>assistant NN Word Job title</td>
<td></td>
</tr>
<tr>
<td>professor NN Word</td>
<td></td>
</tr>
</tbody>
</table>
Automatic Semantic Annotation Using Machine Learning

A generalized tagging rule for the start boundary identification of the Speaker.

Another type of rules, contextual rules, is applied to improve the effectiveness of the system. The basic idea is that $\langle \text{tag} \rangle$ can be used as an indicator of the occurrence of $\langle \text{tag} \rangle$. For example, consider a rule recognizing an end boundary between a capitalized word and a lowercase word. This rule does not belong to the best rule pool as its low precision on the corpus, but it is reliable if used only when closing to a tag $\langle \text{speaker} \rangle$. Consequently, some non-best rules are recovered, and the ones which result in acceptable error rate will be preserved as the contextual rules.

The correction rules are used to reduce the imprecision of the tagging rules. For example, a correction rule shown in Table 3 is used to correct the tagging mistake “at $\langle \text{time} \rangle$ 4 $\langle \text{time} \rangle$ pm” since “pm” should have been part of the time expression. So, correction rules are actions that shift misplaced tags rather than adding new tags.

After all types of rules are induced, information extraction is carried out in the following steps:

- The learned tagging rules are used to tag the texts.
- Contextual rules are applied in the context of introduced tags in the first step.
- Correction rules are used to correct mistaken extractions.
- All the identified boundaries are to be validated, e.g. a start tag (e.g. $\langle \text{time} \rangle$) without its corresponding close tag will be removed, and vice versa.

### Table 3. Example of correction rule

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>Wrong tag</td>
</tr>
<tr>
<td>At</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$\langle /\text{time} \rangle$</td>
</tr>
<tr>
<td>pm</td>
<td></td>
</tr>
</tbody>
</table>
Automatic Semantic Annotation Using Machine Learning

A feature vector) and \( y_i \in \{-1,+1\} \) denotes a classification label. In learning, one attempts to find a model from the labeled data that can separate the training data, while in prediction the learned model is used to identify whether an unlabeled instance should be classified as +1 or -1.

Support Vector Machines (SVMs) is one of the most popular methods for classification (Vapnik, 1998). Now, we use SVM as example to introduce the classification model.

Support vector machines (SVMs) are linear functions of the form \( f(x) = w^T x + b \), where \( w^T x \) is the inner product between the weight vector \( w \) and the input vector \( x \). The main idea of SVM is to find an optimal separating hyper-plane that maximally separates the two classes of training instances (more precisely, maximizes the margin between the two classes of instances). The hyper-plane then corresponds to a classifier (linear SVM). The problem of finding the hyper-plane can be stated as the following optimization problem:

\[
\text{Minimize } \frac{1}{2} w^T w \\
\text{subject to } y_i (w^T x_i + b) \geq 1, i = 1, 2, \ldots, n
\]  

(1)

To deal with cases where there may be no separating hyper-plane due to noisy labels of both positive and negative training instances, the soft margin SVM is proposed, which is formulated as:

\[
\text{Minimize } \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \\
\text{subject to } y_i (w^T x_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, n
\]  

(2)

where \( C \geq 0 \) is the cost parameter that controls the amount of training errors allowed.

It is theoretically guaranteed that the linear classifier obtained in this way has small generalization errors. Linear SVM can be further extended into non-linear SVMs by using kernel functions such as Gaussian and polynomial kernels (Boser, Guyon, and Vapnik, 1992; Schölkopf, Burges, and Smola, 1999; Vapnik, 1999). When there are more than two classes, we can adopt the “one class versus all others” approach, i.e., take one class as positive and the other classes as negative.

Boundary Detection Using Classification Model

We are using a supervised machine learning approach to entity extraction, so our system consists of two distinct phases: learning and extracting. In the learning phase the system uses a set of labeled documents to generate models which we can use for future predictions. The extraction phase takes the learned models and applies them to new unlabelled documents using the learned models to generate extractions.

The method aims at detecting the boundaries (start boundary and end boundary) of an instance. For entity extraction from text, the basic unit that we are dealing with can be a token or a text-line. (Hereafter, we will use token as the basic unit in our explanation.) We try to learn two classifiers that are respectively used to identify the boundaries. The instances are all tokens in the document. All tokens that begin with a start-label are positive instances for the start classifier, while all the other tokens become negative instances for this classifier. Similarly, the positive instances for the end classifier are the last tokens of each end-label, and the other tokens are negative instances.

Figure 2 gives an example of entity extraction as classification. There are two classifiers – one to identify starts of target text fragments and the other to identify ends of text fragments. Here, the classifiers are based on token only (however other patterns, e.g. syntax, can also be incorporated into). Each token is classified as being a start or non-start and an end or non-end. When we classify a token as a start, and also classify one of the closely following token as an end, we view the tokens between these two tokens as a target instance.
In the example, the tokens “Dr. Trinkle’s” is annotated as a “speaker” and thus the token “Dr.” is a positive instance and the other tokens are as negative instances in the speaker-start classifier. Similarly, the token “Trinkle’s” is a positive instance and the other tokens are negative instances in the speaker-end classifier. The annotated data is used to train the two classifiers in advance. In the extracting stage, the two classifiers are applied to identify the start token and the end token of the speaker. In the example, the tokens “Professor”, “Steve”, and “Skiena” are identified as two start tokens by the start classifier and one end token by the end classifier. Then, we combine the identified results and view tokens between the start token and the end token as a speaker. (i.e. “Professor Steve Skiena” is outputted as a speaker)

In the extracting stage, we apply the two classifiers to each token to identify whether the token is a “start”, “end”, neither, or both. After the extracting stage, we need to combine the starts and the ends predicted by the two classifiers. We need to decide which of the starts (if there exist more than one starts) to match with which of the ends (if there exist more than one ends). For the combination, a simple method is to search for an end from a start and then view the tokens between the two tokens as the target. If there exist two consecutive starts and only one end (as the example in Figure 2), then we start the search progress from the first start and view the tokens between the first token and the end token (i.e. “Professor Steve Skiena”) as the target. However, in some applications, the simple combination may not yield good results.

Several works have been conducted to enhance the combination. For example, Finn et al. propose a histogram model (Finn and Kushmerick, 2004; Finn, 2006). In Figure 2, there are two possible extractions: “Professor Steve Skiena” and “Steve Skiena”. The histogram model estimates confidence as $C_s \cdot C_e \cdot P(e \mid s)$. Here $C_s$ is the confidence of the start prediction and $C_e$ is the confidence of the end prediction. (For example, in Naïve Bayes, we can use the posterior probability as the confidence; in SVM, we can use the distance of the instance to the hyper-plane as the confidence.) $P(e \mid s)$ is the probability of a text fragment of that length which we estimate from the training data. Finally, we select the text with the highest confidence as the output.

To summarize, this classification approach simply learns to detect the start and the end of text fragments to be extracted. It treats entity extraction as a standard classification task, augmented with a simple mechanism to combine the predicted start and end tags. Experiments indicate that this approach generally has high precision but low recall. This approach can be viewed as that of one-level boundary classification (Finn and Kushmerick, 2004).

Many approaches can be used to training the classification models, for example, Support Vec-
Automatic Semantic Annotation Using Machine Learning


Sequential Labeling Based Entity Extraction

Entity extraction can be cast as a task of sequential labeling. In sequential labeling, a document is viewed as a sequence of tokens, and a label is assigned to each token to indicate the property of the token. For example, consider the researcher profiling problem, the task is to label a sequence of tokens with their corresponding profile attributes (e.g. position, affiliation, etc.), called tags. Thus the inputting sentence “Lars Arge, Associate Professor, Department of Computer Science Duke University” will result in an output as:

[Lars / Firstname] [Arge / Lastname] [, / Other] [Associate / Position]
[Professor / Position] [, / Other] [Department / Affiliation] [of / Affiliation]
[Computer Affiliation] [Science Affiliation] [Duke Affiliation] [University Affiliation]

Formally, given an observation sequence \( x = (x_1, x_2, ..., x_n) \), the entity extraction task as sequential labeling is to find a label sequence \( y^* = (y_1, y_2, ..., y_n) \) that maximizes the conditional probability \( p(y|x) \), that is:

\[
 y^* = \text{argmax}_y \ p(y|x) \quad (3)
\]

Different from the rule learning and the classification based methods, sequential labeling enables describing the dependencies between target information. The dependencies can be utilized to improve the accuracy of the extraction. Hidden Markov Model (Ghahramani and Jordan, 1997), Maximum Entropy Markov Model (McCallum, Freitag, and Pereira, 2000), and Conditional Random Field (Lafferty, McCallum, and Pereira, 2001) are widely used sequential labeling models. In this section, we will briefly introduce the linear sequential labeling based models (for details, please refer to Tang et al., 2007a).

Generative Model

Generative models define a joint probability distribution \( p(X, Y) \) where \( X \) and \( Y \) are random variables respectively ranging over observation sequences and their corresponding label sequences. In order to calculate the conditional probability \( p(y|x) \), Bayesian rule is employed:

\[
 y^* = \text{argmax}_y \ p(y|x) = \text{argmax}_y \ \frac{p(x,y)}{p(x)} \quad (4)
\]

Hidden Markov Models (HMMs) (Ghahramani and Jordan, 1997) are one of the most common generative models. In HMMs, each observation sequence is considered to be generated by a sequence of state transitions, beginning in some start state and ending when some pre-designated final state is reached. At each state an element of the observation sequence is stochastically generated, before moving to the next state. In the case of researcher profile annotation, each state of the HMM is associated with a profile attribute or “Other”. Although profile attributes do not generate words, the attribute tag associated with any given word can be considered to account for that word in some fashion. It is, therefore, possible to find the sequence of attribute tags that best accounts for any given sentence by identifying the sequence of states most likely to have been traversed when “generating” that sequence of words. Figure 3 shows the structure of a HMM.

These conditional independence relations, combined with the probability chain rule, can be used to factorize the joint distribution over a state sequence \( y \) and observation sequence \( x \) into the product of a set of conditional probabilities:
Automatic Semantic Annotation Using Machine Learning

Figure 3. Graphic structure of first-order HMMs

\[ p(y,x) = p(y_1)p(x_1 | y_1) \prod_{t=2}^{n} p(y_t | y_{t-1}) p(x_t | y_t) \] (5)

In supervised learning, the conditional probability distribution \( p(y_t | y_{t-1}) \) and observation probability distribution \( p(x_t | y_t) \) can be gained with maximum likelihood. While in unsupervised learning, there is no analytic method to gain the distributions directly. Instead, Expectation Maximization (EM) algorithm is employed to estimate the distributions.

Finding the optimal state sequence can be efficiently performed using a dynamic programming such as Viterbi algorithm.

Generative models define a joint probability distribution \( p(X,Y) \) over observations and label sequences. This is useful if the trained model is used to generate data. However, for defining a joint probability over observations and label sequences, a generative model needs to enumerate all possible observation sequences, usually resulting into highly expensive cost. Therefore, generative models must make strict independence assumptions in order to make inference tractable. Consequently, it is not practical to represent complicated interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable.

Discriminative models provide a convenient way to overcome the strong independence assumption of generative models.

Discriminative Models

Instead of modeling joint probability distribution over observation and label sequences, discriminative models define a conditional distribution \( p(y|x) \) over observation and label sequences. This means that when identifying the most likely label sequence for a given observation sequence, discriminative models use the conditional distribution directly, without bothering to make any dependence assumption on observations or enumerate all the possible observation sequences to calculate the marginal probability \( p(x) \).

MEMMs (McCallum et al., 2000) are a form of discriminative models for labeling sequential data. MEMMs consider observation sequences to be conditioned upon rather than generated by the label sequence. Therefore, a MEMM has only a single set of separately trained distributions of the form:

\[ p(y_{t+1} | y_t, x) \] (6)

which represent the probability of transition from state \( y_t \) to \( y_{t+1} \) on observation \( x \). The fact that each of these functions is specific to a given state means that the choice of possible states at any given instant in time \( t+1 \) depends only on the state of the model at time \( t \). Figure 4 show the graphic structure of MEMMs.

Given an observation sequence \( x \), the conditional probability over label sequence \( y \) is given by:
Treating observations to be conditioned upon states rather than generated by means that the probability of each transition may depend on non-independent, interacting features of the observation sequence. Making use of maximum entropy framework and defining each state-observation transition function to be a log-linear model, equation (6) can be calculated as:

\[
p(y | x) = p(y_1 | x_1) \prod_{i=2}^{n} p(y_i | y_{i-1}, x_{1:i})
\]

Treating observations to be conditioned upon states rather than generated by means that the probability of each transition may depend on non-independent, interacting features of the observation sequence. Making use of maximum entropy framework and defining each state-observation transition function to be a log-linear model, equation (6) can be calculated as:

\[
p(y_{i+1} | y_i, x) = \frac{1}{Z(y_i, x)} \exp(\sum_{k} \lambda_k f_k(y_{i+1}, y_i, x))
\]

where \(Z(y_i, x) = \sum_{y_{i+1}} \exp(\sum_{k} \lambda_k f_k(y_{i+1}, y_i, x))\) is a normalization factor; \(\lambda_k\) are parameters to be estimated and \(f_k\) is a feature function. The parameters can be estimated using Generalized Iterative Scaling (GIS) (McCallum et al., 2000). Each feature function is a binary feature. For example, feature \(f(y', y, x)\) implies that if the current and the previous tags are \(y\) and \(y'\), and the observation is \(x\), then the feature value is 1; otherwise 0. Identifying the most likely label sequence given an observation sequence can be done efficiently by dynamic programming (McCallum et al., 2000).

Maximum Entropy Markov Models suffer from the Label Bias Problem (Lafferty et al., 2001), because MEMMs define a set of separately trained per-state probability distributions. Here we use an example to describe the label bias problem. The MEMM in Figure 5 is designed to shallow parse the sentences:

1. The robot wheels Fred round.
2. The robot wheels are round.

Consider when shallow parsing the sentence (1). Because there is only one outgoing transition from state 3 and 6, the per-state normalization requires that \(p(4|3, \text{Fred}) = p(7|6, \text{are}) = 1\). Also it’s easy to obtain that \(p(8|7, \text{round}) = p(5|4, \text{round}) = p(2|1, \text{robot}) = p(1|0, \text{The}) = 1\), etc. Now, given
Automatic Semantic Annotation Using Machine Learning

\[ p(3|2, \text{wheels}) = p(6|2, \text{wheels}) = 0.5, \]
by combining all these factors, we obtain:

\[ p(0123459|\text{The robot wheels Fred round.}) = 0.5, \]
\[ p(0126789|\text{The robot wheels Fred round.}) = 0.5. \]

Thus the MEMM ends up with two possible state sequences 0123459 and 0126789 with the same probability independently of the observation sequence. It’s impossible for the MEMM to tell which one is the more likely state sequence over the given sentence. Likewise, given \( p(3|2, \text{wheels}) < p(6|2, \text{wheels}) \), MEMM will always choose the bottom path despite what the preceding words and the following words are in the observation sequence.

The label bias problem occurs because a MEMM uses per-state exponential model for the conditional probability of the next states given the current state.

People, therefore, propose Conditional Random Fields (CRFs) to benefit the advantages from modeling conditional probability, at the same time to avoid the label bias problem. CRFs are undirected graphical model trained to maximize a conditional probability. CRFs can be defined as follows:

A CRF is a random field globally conditioned on the observation. Linear-chain CRFs were first introduced by Lafferty et al. (2001). Figure 6 shows the graphical structure of the linear-chain CRFs.

By the fundamental theorem of random fields (Hammersley and Clifford, 1971), the conditional distribution of the labels \( y \) given the observations data \( x \) has the form:

\[
p(y|x) = \frac{1}{Z(x)} \exp \left( \sum_{t=1}^{T} \sum_{k} \lambda_k \cdot f_k(y_{t-1}, y_t, x, t) \right)
\]

(9)

where \( Z(x) \) is a normalization factor, also known as partition function, which has the form:

\[
Z(x) = \sum_y \exp \left( \sum_{t=1}^{T} \sum_{k} \lambda_k \cdot f_k(y_{t-1}, y_t, x, t) \right)
\]

(10)

where \( f_k(y_{t-1}, y_t, x, t) \) is a feature function which can be either real-valued or binary-valued. The feature functions can measure any aspect of a state transition, \( y_{t-1} \rightarrow y_t \), and the observation sequence, \( x \), centered at the current time step \( t \). \( \lambda_k \) corresponds to the weight of the feature \( f_k \).

The most probable labeling sequence for an input \( x \):

\[
y^* = \arg \max_y p_y(y|x)
\]

(11)

can be efficiently calculated by dynamic programming using Viterbi algorithm.

We can train the parameters \( \lambda = (\lambda_1, \lambda_2, \ldots) \) by maximizing the likelihood of a given training set \( T = \{(x^{(k)}, y^{(k)})\}_{k=1}^{N} \).
Many methods can be used to do the parameter estimation. The traditional maximum entropy learning algorithms, such as GIS, IIS can be used to train CRFs (Darroch and Ratcliff, 1972). In addition, preconditioned conjugate-gradient (CG) (Shewchuk, 1994) or limited-memory quasi-Newton (L-BFGS) (Nocedal and Wright, 1999) have been found to perform better than the traditional methods (Sha and Pereira, 2003). The voted perceptron algorithm (Collins, 2002) can also be utilized to train the models efficiently and effectively.

To avoid overfitting, log-likelihood is often penalized by some prior distribution over the parameters. Empirical distributions such as Gaussian prior, exponential prior, and hyperbolic-L\textsubscript{1} prior can be used, and empirical experiments suggest that Gaussian prior is a safer prior to use in practice (Chen and Rosenfeld, 1999).

CRFs avoid the label bias problem because it has a single exponential model for the conditional probability of labels of the entire sequence given the observation. Therefore, weights of different features at different states can be traded off against each other.

Alternative Conditional Random Fields

Lafferty, Zhu, and Liu (2004) investigated employing kernel in conditional random fields. Kernel can be considered as a function comparing the cliques of different graphs. The dual parameters (like those in SVM) depend on all potential assignments of the cliques in the graph. Therefore, clique selection becomes important. They also argued that kernel enables semi-supervised learning.

Taskar, Guestrin, and Koller (2003) employed the idea of large margin under the framework of Markov random fields for assigning labels to structured data.

Jiao, Wang, and Lee (2006) proposed an alternative objective function for linear-chain CRFs using labeled and unlabeled data. The experiments on protein prediction showed the model taking advantage of unlabeled data gained accuracy compared with the model without unlabeled data.

Using Sequential Labeling for Entity Extraction

By casting entity extraction as sequential labeling, a set of labels need to be predefined based on the extraction task. For example, in the annotation of researcher profile (Tang et al., 2007c), labels such as “Researcher Name”, “Position”, “Affiliation”, “Email”, “Address”, and “Telephone” were defined. Then a document is viewed as a sequence x of observation unit. The observation unit can be a word, a text line, or any other granularity of linguistic information. Then the task is to find a label sequence y that maximizes the conditional probability \( p(y|x) \) using the models described above.

In generative models, only features on the current observation unit can be defined. Due to the conditional nature, discriminative models provide the flexibility of incorporating non-independent or even arbitrary features of input to improve the performance. For example, in the task of researcher profile annotation, with CRFs we can use as features not only text content, but also layout and external lexicon. Empirical experiments show that incorporating non-independent and arbitrary features can significantly improve the performance.

On the other hand, incorporation of non-independent and arbitrary features of discriminative models may also lead to too many features and some of the features are of little contributions to the model. The method of feature induction can be used to obtain the most useful features for efficiently performing training the model (McCallum, 2003).
Non-Linear Markov Random Fields Based Semantic Annotation

Markov random field models, for instance HMMs (Ghahramani and Jordan, 1997), MEMMs (McCallum et al., 2000), and CRFs (Lafferty et al., 2004) are widely used for semantic annotation including both entity extraction and relation extraction (we will introduce the methods for relation extraction in the next sub section). However, most of the previous methods based on the three models are linear-chain models, which can only describe the linear-dependencies, and cannot describe non-linear dependencies (Lafferty et al., 2001; Zhu, Nie, Wen, Zhang, and Ma, 2005; Tang, Hong, Li, and Liang, 2006). In this section, we will discuss several non-linear Markov random fields for semantic annotation. We will also introduce the inference on the non-linear Markov model.

Hierarchical Hidden Markov Models

A Hierarchical Hidden Markov Model (HHMM) is a structured multi-level discrete stochastic process. The HHMM generalizes the familiar HMM by making each of its hidden node states a similar stochastic model on its own, i.e. each state is an HMM as well (Skounakis, Craven, and Ray, 2003). The graphical structure of a two-level HHMM can be represented as Figure 7.

In the figure, we use the white circle to denote the observation $x_i$ and the gray circle the hidden variable $y_i$. The HMMM model has two levels. In the inner level, each node $y_{ij}$ represents a unit of fine granularity, for example a unique token. Each node $y_{ij}$ generates an observation node. In the outer level, each node $y_{i}^{'}$ represent a unit of coarse granularity, for example a text line or several tokens. Each node $y_{i}^{'}$ generates a state sequence rather than a single observation. The two-level HHMM can be easily extended to a multi-level model (c.f., e.g., (Fine, Singer, and Tishby, 1998)).

Hierarchical Conditional Random Fields

A Hierarchical Conditional Random Field (HCRF) model is a tree structured version of the Conditional Random Fields. Information on web pages is usually organized as hierarchy. The conventional linear-chain CRF model cannot describe dependencies across the hierarchically laid-out information. To better incorporate dependencies across the hierarchically laid-out information, proposal of new CRF models is necessary.

For example, Zhu et al. (2007) propose a Hierarchical Conditional Random Field for simultaneous record detection and attribute labeling in web data. The task is to extract product information from web pages. A data record, describing a product, is a block in the web page and product attributes are information in the data record and are used to describe different aspects of the product. Instead of using traditional methods that attempt to do data record detection and attribute labeling in two separate phases the authors propose using a Hierarchical Conditional Random Field Model (HCRF) to conduct record extraction and attribute labeling simultaneously. The basic idea

![Figure 7. Graphic structure of a two-level HHMM](image-url)
of using HCRF is that record detection and attribute labeling from web pages can benefit from each other. The structure of the proposed HCRF model is shown in Figure 8. The hidden variables (indicated as gray nodes) represent a product record or an attribute of the product record and the observations (indicated as white nodes) represent an observation unit on the web page. The hidden variables are organized hierarchically. In the bottom level the hidden variables represent the label of attributes (for example, product image, product name) and in the upper level the hidden variables represent the label of product records or part of product records (for example, a product name-image block that contains the product name and the product image).

We also propose Tree-structured Conditional Random Fields (TCRFs), which can incorporate dependencies across the hierarchically laid-out information (Tang et al., 2006). Here we use an example to introduce the problem of hierarchical semantic annotation. Figure 9 (a) give an example document, in which the underlined text are what we want to extract including two telephone numbers and two addresses. The information can be organized as a tree structure (cf. Figure 9 (b)). In this case, the linear-chain CRFs cannot model the hierarchical dependencies and thus cannot distinguish the office telephone number and the home telephone number from each other. Likewise for the office address and the home address.

We present the graphical structure of the TCRF model as a tree and reformulate the conditional distribution by defining three kinds of transition features respectively representing the parent-child dependency, child-parent dependency, and sibling dependency. As the tree structure can be cyclical, exact inference is expensive. We propose using the Tree-based Reparameterization (TRP) algorithm (Wainwright, Jaakkola, and Willsky, 2001) to compute the approximate marginal probabilities for edges and vertices. We conducted experiments on company annual reports collected from Shang Stock Exchange. Experimental results indicate that the TCRFs can significantly outperform the existing linear-chain CRF model (+7.67% in terms of F1-measure). See (Tang et al., 2007a) for details.

**Skip-Chain Conditional Random Fields**

In some specific application, it might be helpful to incorporate long-distant dependencies. Skip-
Skip-chain Conditional Random Fields are proposed to address this problem (Sutton and McCallum, 2006; Bunescu and Mooney, 2005b). Skip-chain Conditional Random Fields can incorporate long-distance dependencies. For example, in entity extraction, a person name (e.g. Robert Booth) is mentioned more than one time in a document. All mentions might have the same label, such as SEMINAR-SPEAKER. An IE system can take advantage of this fact by labeling repeated words identically. Furthermore, identifying all mentions of an entity can be useful in itself, because each mention might contain different useful information. The skip-chain CRF, by identifying all mentions identically, combine features from all occurrences so that the extraction can be made based on global information. The skip-chain CRF is proposed to address this (Sutton and McCallum, 2006; Bunescu and Mooney, 2005b).

The skip-chain CRF is essentially a linear-chain CRF with additional long-distance edges between similar words. These additional edges are called **skip edges**. The features on skip edges can incorporate information from the context of both endpoints, so that strong evidence at one endpoint can influence the label at the other endpoint.

Formally, the skip-chain CRF is defined as a general CRF with two clique templates: one for the linear-chain portion, and one for the skip edges. For an input \( x \), let \( C = \{(u,v)\} \) be the set of all pairs of sequence positions for which there are skip edges. The probability of a label sequence \( y \) given an \( x \) is modeled as:

\[
P(y \mid x) = \frac{1}{Z(x)} \exp \left( \sum_{i=1}^{T} \sum_{k} \lambda_k \cdot f_k(y_{t-1}, y_t, x, t) + \right.
\]

\[
\left. \sum_{(u,v) \in C} \lambda_l \cdot f_l(y_u, y_v, x, u, v) \right)
\]

where \( Z(x) \) is a normalization factor, \( f_k \) is the feature function similar to that in the linear-chain CRF model and \( f_l \) is the feature function on a skip edge. \( \lambda_k \) and \( \lambda_l \) are weights of the features.

Because there might have a loop in the skip-chain CRF, exact inference is intractable. The running time required by exact inference is exponential in the size of the largest clique in the graph's junction tree. Instead, approximate inference using loopy belief propagation is performed, such as TRP (Wainwright et al., 2001).

Richer kinds of long-distance factor than just over pairs of words can be considered to augment the skip-chain model. These factors are useful for modeling exceptions to the assumption that similar words tend to have similar labels. For example, in Named Entity Recognition, the word “China” is as a location name when it appears alone, but when it occurs within the phrase The China Daily, it should be labeled as an organization (Finkel, Grenager, and Manning, 2005).
Dynamic Conditional Random Fields

Sutton, Rohanimanesh, and McCallum (2004) proposed Dynamic Conditional Random Fields (DCRFs). It generalizes the linear-chain conditional random fields, in which each slice is a Bayesian network, the interaction between slices can be seen as chain conditional random fields. Figure 10 shows the graph structure of a two-dimensional grid CRF model, which can be regarded as a special kind of DCRF.

The two-dimensional CRF has practical applications. For example, it can perform POS (Part-Of-Speech) tagging and NER (Named Entity Recognition), two typical tasks in NLP, simultaneously. The model can describe dependencies between the two subtasks.

Some other CRF models or Markov models are also proposed for addressing different types of special cases. For example 2D Conditional Random Fields (2D CRFs) (Zhu et al., 2005).

Inference on the Non-Linear CRFs

Non-linear graphical models can capture the long-range interaction between different labels. However, the difficulty of this kind of model lies in inference.

Given a set of observations, inference in graphical model has two tasks: (a) to estimate the marginal distribution of each random hidden variable or to estimate the most likely configuration of the hidden variables, that is maximum a posteriori (MAP) estimation. Both tasks can be solved under the framework of belief propagation (BP) (Yedidia, Freeman, and Weiss, 2003). The basic process is to set a root node and then collect messages from all nodes until root node by starting from leave nodes and send back the messages to the leave nodes from the root node. The process continues in the whole graph until convergence. BP generates correct results if the graph has no loops. If the graph contains loops, we can carry out approximate inference. The proposed algorithms include Tree-based reparameterization (TRP) (Wainwright et al., 2001) and Junction tree based inference, also called Generalized Belief Propagation (GBP) (Yedidia et al., 2003).

Here we would like to give some definitions of belief at node and edge, and message between nodes. The belief at a node is proportional to the product of the local potential of that node \( \psi(y_i) \), and all the messages coming into node \( i \) is:

\[
b_i(y_i) = k \phi(y_i) \prod_{j \in N(i)} m_{ij}(y_i) \quad (14)
\]

In the formula, \( k \) is a normalization factor, and \( N(i) \) denotes the neighboring nodes of \( i \). In fact, the belief at a node \( i \) is the marginal probability of that corresponding variable \( (y_i) \). The message can be computed using the following formula:

\[
m_{ij}(y_j) = \sum_{y_i} \phi(y_i) \psi_j(y_i, y_j) \prod_{k \in N(i) \setminus j} m_{ik}(y_i) \quad (15)
\]

Considering we also need the belief at an edge in solving the Markov random field, we define:

\[
b_{ij}(y_j) = k \phi(y_i) \phi(y_j) \prod_{k \in N(i) \setminus j} m_{ik}(y_i) \prod_{l \in N(j) \setminus i} m_{lj}(y_j) \quad (16)
\]

We will take the tree structure as an example to explain the message passing process. In Figure 11, we omit observation nodes since we concentrate on message passing through hidden nodes, \( m_{ij}(y_j) \)
is a “message” from a hidden node $i$ to the hidden node $j$ about what state node $j$ should be in, just as we defined above. We schedule message passing in two stages by choosing one node as root, collecting message from all nodes at leaves and sending message to all nodes from root. In the figure, suppose we choose $y_1$ as root, collecting messages in orders: $m_{43}$, $m_{33}$, $m_{32}$ and $m_{23}$, then sending messages in orders: $m_{32}$, $m_{33}$, $m_{35}$, $m_{34}$. After these two stages, we can get the belief at one node and at one edge, as well as marginal probabilities of a variable and two joint variables.

For structures with loops, we can convert the graph to a tree, through pruning some edges randomly (TRP) or through triangulating the graph and generating junction tree (Junction tree inference), see (Wainwright et al., 2001) and (Yedidia et al., 2003) for details.

**Relation Extraction**

Relation extraction is another important issue in semantic annotation. It is aimed at finding semantic relations between entities. That is to say, given a pair of entities, the objective is to decide whether a particular relation holds between the entities. It can be also viewed as a step following the entity extraction. For example, consider the following sentence:

“Andrew McCallum is an Associate Professor at University of Massachusetts, Amherst.”

An entity extraction system should recognize that “Andrew McCallum” is a person, “University of Massachusetts” is an organization, and “Amherst” is a location. In the above example, relations between these entities, such as the relation “work-for” (“Andrew McCallum” works for “University of Massachusetts”) and the relation “located-in” (“University of Massachusetts” is located in “Amherst”), will be found in an ideal relation extraction system. Table 4 shows the extracted relations from the sentence.

Let us start from some background knowledge of relation extraction. The problem of relation extraction was formulated as a part of Message Understanding Conferences (MUC). Systems attending the conference were tested and evaluated on New York Times News Service data and the task is limited to relations with organizations: “employee_of”, “product_of”, “location_in”.

In another program, the NIST Automatic Content Extraction (ACE) program, this task is defined as Relation Detection and Characterization (RDC). ACE program defines three main objectives for information extraction: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). The EDT task entails the detection of entity mentions and chain them together by identifying their coreference. In ACE vocabulary, entities are objects, mentions references to them, and relations are semantic relationships between entities. The RDC task detects and classifies implicit and explicit relations between entities identified by the EDT task. RDC is broadly comparable with the relation extraction task in MUC. In ACE 2004 a type and sub-type hierarchy for both entities and

<table>
<thead>
<tr>
<th>Relation</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee_of</td>
<td>Andrew McCallum is employee of University of Massachusetts</td>
</tr>
<tr>
<td>Located_in</td>
<td>University of Massachusetts is located in Amherst</td>
</tr>
</tbody>
</table>
relations was introduced, which is regarded as an important step towards ontology-based Semantic Annotation. ACE continues the competition from 1999 up to now.

The benefit of relation extraction technology is obvious. With this technology, we can integrate large databases of relational information and generate new information for data mining, question answering systems, and information retrieval. Therefore, numerous studies on relation extraction have been conducted. In this section, we will introduce four typical methods for relation extraction: classification based approach, kernel based approach, sequential labeling based approach, and unsupervised approach.

Classification Based Methods

The classification based method is one of the most popular methods for relation extraction. It formalizes the problem as classification, i.e. identifying whether an entity pair has a specific relation using a classifier.

For example, Kambhatla (2004) proposes combining diverse lexical, syntactic, and semantic features for each entity pairs. Then, they use Maximum Entropy models as classification models and apply their method to the dataset of ACE RDC task. Here is an example:

The American Medical Association voted yesterday to install the heir apparent as its president-elect, rejecting a strong, upstart challenge by a District doctor who argued that the nation’s largest physicians’ group needs stronger ethics and new leadership.

In electing Thomas R. Reardon, an Oregon general practitioner who had been the chairman of its board, ...

For each pair of mentions (references to entities), we can define the following features:

- **Words.** The features represent the words of both the mentions and all the words in between.
- **Entity Type.** The feature represents the entity type (including PERSON, ORGANIZATION, LOCATION, FACILITY, Geo-Political Entity or GPE) of both the mentions.
- **Mention Level.** The feature represents the mention level (one of NAME, NOMINAL, or PRONOUN) of both the mentions.
- **Overlap.** The features represent the number of words (if any) separating the two mentions, the number of other mentions in between, flags indicating whether the two mentions are in the same noun phrase, verb phrase or prepositional phrase.
- **Dependency.** The features represent the words and part-of-speech and chunk labels of the words on which the mentions are dependent in the dependency tree derived from the syntactic parse tree.
- **Parse Tree.** The feature represents the path of non-terminals (removing duplicates) connecting the two mentions in the parse tree, and the path annotated with head words.

Compared with (Kambhatla, 2004), Zhou, Su, Zhang, and Zhang (2005) separately incorporate the base phrase chunking information, which contributes to most of the performance improvement from syntactic aspect. See also (Jiang and Zhai, 2007)

Kernel Based Methods

Kernel based method is an emerging method in these years for relation extraction. In many cases, it is used as an alternative to feature vector based method, especially when it is infeasible to create a feature vector for an instance due to the high dimensionality of the feature space. Generally, kernel methods are non-parametric density estimation techniques that compute a kernel function
between data instances, where a kernel function can be thought of as a similarity measure. Given a set of labeled instances, a kernel based method determines the label of a novel instance by comparing it to the labeled training instances using this kernel function. Formally, a kernel function \( K \) is a mapping \( K : X \times X \rightarrow [0, \infty] \) from instance space \( X \) to a similarity score:

\[
K(x, y) = \sum_i \phi_i(x) \phi_i(y) = \phi(x) \cdot \phi(y)
\]  

(17)

Here, \( \phi_i(x) \) is a feature function over the instance \( x \). The kernel function must be symmetric and positive semi-definite, which means the \( n \times n \) Gram matrix \( G \) defined by \( G_{ij} = K(x_i, x_j) \) is positive semi-definite. Given a training set \( S = \{x_1, \ldots, x_n\} \), Gram matrix \( G \) will be computed. Then, the classifier finds a hyperplane which separates instances of different classes. To classify an unseen instance \( x \), the classifier first projects \( x \) into the feature space defined by the kernel function and then determine which side of the separating hyperplane \( x \) lies.

Culotta and Sorensen (2004) investigate a rich sentence representation and propose a general framework to allow feature weighting, as well as the use of composite kernels to reduce kernel scarcity. In (Culotta and Sorensen, 2004), the task is defined to generate potential relation instances by iterating over all pairs of entities occurring in the same sentence. For each entity pair, they create an augmented dependency tree to represent this instance. A dependency tree is a representation that denotes grammatical relations between words in a sentence. A set of rules are used to map a parse tree to a dependency tree. For example, subjects are dependent on their verbs and adjectives are dependent on the nouns they modify. Then, they define a tree kernel function \( K(T_1, T_2) \) which returns a normalized, symmetric similarity score in the range \((0, 1)\) for two trees \( T_1 \) and \( T_2 \).

Formally, a relation instance is a dependency tree \( T \) with nodes \( \{t_0, \ldots, t_n\} \). The features of node \( t_i \) are given by \( \phi(t_i) = \{v_i, \ldots, v_j\} \). Also, \( t_j[c] \) denotes the \( j \)-th child of node \( t_j \), \( t[c] \) denotes the set of all children of node \( t \), \( t[j] \) denotes a subset of children of \( t \), and \( t_p \) denotes the parent of \( t_p \).

First, two functions over the features of tree nodes are defined: a matching function \( m(t_i, t_j) \in \{0, 1\} \) and a similarity function \( s(t_i, t_j) \in (0, \infty) \):

\[
m(t_i, t_j) = \begin{cases} 1 & \text{if } \phi_i(t_i) = \phi_j(t_j) \\ 0 & \text{otherwise} \end{cases}
\]

(18)

\[
s(t_i, t_j) = \sum_{v_i \in \phi_i(t_i)} \sum_{v_j \in \phi_j(t_j)} C(v_i, v_j)
\]

(19)

where \( C(v_i, v_j) \) is a compatibility function between two feature values. For example, in the simplest case where:

\[
C(v_i, v_j) = \begin{cases} 1 & \text{if } v_i = v_j \\ 0 & \text{otherwise} \end{cases}
\]

\( s(t_i, t_j) \) returns the number of feature values in common between feature vectors \( \phi_i(t_i) \) and \( \phi_j(t_j) \).

For two dependency trees \( T_1, T_2 \) with root nodes \( r_1 \) and \( r_2 \), we define the tree kernel \( K(T_1, T_2) \) as follows:

\[
K(T_1, T_2) = \begin{cases} 0 & \text{if } m(r_1, r_2) = 0 \\ s(r_1, r_2) + K_c(r_1[c], r_2[c]) & \text{otherwise} \end{cases}
\]

(20)

where \( K_c \) is a kernel function over children. Let \( a \) and \( b \) be sequences of indices such that \( a \) is a sequence \( a_1 \leq a_2 \leq \ldots \leq a_n \), and likewise for \( b \). Let \( d(a) = a_n - a_1 + 1 \) and \( l(a) \) be the length of \( a \). Then we have:

\[
K_c(t_i[c], t_j[c]) = \sum_{a, b : d(a) = d(b)} \lambda^{|a|} \lambda^{|b|} K(t_i[a], t_j[b])
\]

(21)

The constant \( 0 < \lambda < 1 \) is a decay factor that penalizes matching subsequences that are spread out within the child sequences.
Intuitively, in formula (18) and (19), \( m(t_i, t_j) \) and \( s(t_i, t_j) \) provide a way to discretize the similarity between two nodes. If \( \phi_m(t_i) \neq \phi_m(t_j) \), then two nodes are completely dissimilar. On the other hand, if \( \phi_m(t_i) = \phi_m(t_j) \), then \( s(t_i, t_j) \) is computed. Thus, restring nodes by \( m(t_i, t_j) \) is a way to prune the search space of matching subtrees, as shown in formula (20). In formula (21), the function means, whenever we want to find a pair of matching nodes, we search for all matching subsequences of the children of each node. A matching subsequence of children is a sequence of children \( a \) and \( b \) such that \( m(a_i, b_j) = 1(\forall i < n) \). For each matching pair of nodes \((a_i, b_j)\) in a matching subsequence, we accumulate the result of the similarity function \( s(a_i, b_j) \) and then recursively search for matching subsequences of their children \( a_{[c]}, b_{[c]} \).

Zelenko, Aone, and Richardella (2003) investigate identifying relations like person-affiliation and organization-location from text. They define kernels over shallow parse representations of text and design efficient algorithms for computing the kernels. Bunescu and Mooney (2005a) also present a new kernel method based on a generalization of subsequence kernels. This kernel uses three types of subsequence patterns that are typically employed in natural language to assert relationships between two entities. For more details, please refer (Zelenko et al., 2003; Bunescu and Mooney, 2005a; Zhang, Zhang, and Su, 2006; Zhao and Grishman, 2005)

**Sequential Labeling Based Methods**

Although classification based methods have been proved successful in various kinds of applications, there are still some disadvantages. First, for any two entities, the candidate relations might be numerous. This makes it inconvenient, even impossible, to train different models for each relation. Second, classification based methods build local classifier from labeled relations and context around them. They cannot model correlations between different entity-pairs, therefore cannot take advantage of dependencies between them.

In order to address the problem, sequential labeling based methods for relation extraction have been studied. Sequential Labeling methods, for example Conditional Random Fields (Lafferty et al., 2001) have been proved to be successful in entity extraction tasks such as Named Entity Recognition and Part-Of-Speech tagging. Culotta, McCallum, and Betz (2006) introduce CRFs into relation extraction. They propose formalizing relation extraction as a sequential labeling task and using a Conditional Random Field model to identify relations from entities.

The proposed method supposes that there is an identified principal entity and the task is to identify the relations between the secondary entities (defined below) and the principal entity. The authors concentrate their investigation on biographical text, e.g. encyclopedia articles. A biographical text mostly discusses one entity, which is referred as the principal entity. Other entities mentioned in the text are referred as secondary entities. Therefore, the problem is viewed as a tagging problem, that is, assigning a label to each observation unit in the sequence. The label indicates a relation between the principal entity and the current unit. For example, as shown in Figure 12, the principal entity in this biographical text is George W. Bush. Two secondary entities, “George H. W. Bush” and “Barbara Bush”, are labeled with their relation with George W. Bush.

*Figure 12. An example of sequence labeling for relation extraction*

**George W. Bush**

George is the son of George H. W. Bush and Barbara Bush.

father         mother
A linear-chain Conditional Random Field model is employed to find relation from a biographical data. See Section 2.1.3 for details of the CRFs.

Roth and Wen (2002) propose another sequential labeling based method for relation extraction. They first train different local classifiers for identifying entities and relations. Then, they perform global inference that accounts for the mutual dependencies across entities. They construct a belief network along with constraints induced among entity types and relations. At last, the most probable assignment for entities and relations are discovered by an efficient Viterbi-like algorithm. See (Roth and Wen, 2002) for details.

Other Methods

Besides the methods discussed above, other unsupervised approaches have also been studied, including clustering, semi-supervised learning, and rule based methods.

Clustering is one of the important methods for relation extraction. For instance, (Brody, 2007) created a simplified and generalized grammatical clause representation which utilized information-based clustering and inter-sentence dependencies to extract high level semantic relations. (Davidov, Rappoport, and Koppel, 2007) discovered and enhanced concept specific relations other than general relations by web mining. They utilized clustering patterns which contain concept words and words related to them to implement their methods. Their approach can be used to discover unknown relations.

As supervised learning requires a large number of training data which leads to expensive labor cost, many research works have been made to minimize the labeling cost. For example, (Bunescu and Mooney, 2007) presented a new approach requiring only handful training examples to extract relations between entities. They used web as the corpus. First, pairs of entities which exhibit relations or no relations are found. Then searching all the sentences which describe these entities and creating positive and negative bags (weak form of multiple instances learning), they extended an existing approach using Support Vector Machines and string kernel to handle this weak form of multiple instances learning. As many errors generated in unsupervised and semi-supervised learning for relation extraction were attributed to the entities in the relations were not extracted correctly. (Rosenfeld and Feldman, 2007) proposed incorporating corpus statistics to validate and correct the arguments of the extracted relation instances.

Rule based methods are also studied and implemented in several systems including DIPRE (Brin, 1998), Snowball (Agichtein and Gravano, 2000), and Espresso (Pennacchiotti and Pantel, 2006). All these systems use bootstrapping techniques which have been proved as a successful automatic text processing methods. Here, we use the Snowball system, which is based on the DIPRE algorithm, as an example to illustrate the rule based approach.

Given a handful of training examples from users, the Snowball system uses these examples to generate extraction patterns, which in turn result in new tuples (i.e. train examples) extracted from the document collection.

The main processing flow includes:

1. Start with a few certain relationship (e.g. <Microsoft, Redmond> for location relationship)
2. Locate occurrences of these examples (e.g. “Microsoft is at Redmond.”)
3. Generalize patterns of the relationship from these occurrences
4. Repeat (2) until no more examples can be extracted

The advantage of the method is that it does not need a large number of manually labeled training data. However, a strong limitation of the mutual bootstrapping based method is that a minor error
can introduce a large amount of errors during the following iterations. Therefore, in each iteration, the confidence of the extracted patterns and examples are estimated and the most reliable ones are selected for further consideration.

Another problem of this method is the low extraction recall, because the relation patterns produced by the bootstrapping based method may be specific to some examples. In order to improve recall, Snowball patterns are generalized by clustering similar examples using a simple single-pass clustering algorithm. After generalizing patterns, the system discovers new tuples that match the patterns in a certain degree. Each candidate tuple will then have a number of patterns that help generate it associated with a degree of match. This information helps Snowball to decide what candidate tuples to add to the final template.

Some other relation extraction systems also perform relation extraction by combining the linguistic patterns and statistical processing. For example, RelExt (Schutz and Buitelaar, 2005) is a system intending to automatically identify relations between concepts from an existing domain-specific ontology. RelExt works by extracting relevant verbs and their grammatical arguments (i.e. terms) from a domain-specific text collection and computing corresponding relations through a combination of linguistic and statistical processing. LEIL (Suchanek, Ifrim, and Weikum, 2006) argues that relation extraction can benefit significantly from deep natural language processing. Their strategy is to discover text patterns that express the semantic relation, generalize these patterns, and then apply them to the new text. They utilized parsing and statistical learning to discover and generalize the patterns. (Maedche and Staab, 2000) utilizes shallow text processing to discover non-taxonomic conceptual relations. They introduced association rule learning to discover more relations based on texts building on shallow parsing techniques.

**SEMANTIC ANNOTATION SYSTEMS**

There are a number of available systems that address semantic annotation from different aspects. A complete review of this subject is outside the scope of this chapter. We present some of them through their principles and availabilities. Many systems support manual annotation, for example: Protégé-2000 (Eriksson, Fergerson, Shahar, and Musen, 1999), WebKB (Martin and Eklund, 1999), SHOE (Heflin and Hendler, 2000), Artequakt (Alani et al., 2003), Annotea (Kahan and Koivunen, 2001), Ontobroker (Fensel, Decker, Erdmann, and Studer, 1998), and SEAN (Mukherjee, Yang, and Ramakrishnan, 2003). As manual annotation is not our focus here, we will concentrate on (semi-) automatic annotation systems.

**CREAM**

CREAM is a comprehensive framework for creating annotations, relational metadata in the Semantic Web, including tools for both manual and semi-automatic annotation of pages (Handschuh, Staab, and Ciravegna, 2001). Figure 13 shows the architecture of CREAM. The complete design of CREAM comprises a plug-in structure, which is flexible with regard to adding or replacing modules. We give a brief introduction of the main modules in the system as follows:

- **Document Viewer**: The document viewer visualizes the web page contents. The annotator may easily provide annotations by highlighting text.
- **Ontology Guidance**: The newly created annotations must be consistent with a community’s ontology. The ontology is used to guide annotators towards creating relational metadata.
- **Crawler**: The crawler collects the available relevant entities so that annotators can look for proper reference or recognize whether properties have already been instantiated.
• **Annotation Inference Server**: Relational metadata, proper reference and avoidance of redundant annotation require querying for instances. The annotation inference server reasons on crawled and newly annotated instances.

• **Document Management**: It stores annotated web pages together with their annotations. When the web page changes, the old annotations may still be valid or they may become invalid. The annotator can decide based on the old annotations and the changes of the web page.

• **Information Extraction**: CREAM uses two major techniques: First, “wrappers” can be learned from given markup in order to annotate similarly pages (cf., e.g., Kushmerick, 2000). Second, named entities, propose coreferences, and relations are recognized from texts (cf., e.g., MUC).

• **Storage and Replication**: CREAM stores annotations both inside the document in the document management and in the annotation inference server.

Based on CREAM framework, Handschuh, Staab, and Maedche (2001) have implemented a semantic annotation tool called OntoMat.

**KNOWITALL**

KNOWITALL system aims to automate the process of extracting large collections of facts, concepts, and relationships from the Web in an unsupervised, domain-independent, and scalable manner. KNOWITALL uses a generate-and-test architecture that extracts information in two stages. Inspired by Hearst (Etzioni et al., 2004), KNOWITALL is seeded with an extensible ontology and a small set of domain-independent extraction patterns from which it creates text extraction rules for each class and relation in its ontology. Next, KNOWITALL automatically tests the plausibility of the candidate facts it extracts using point-wise mutual information (PMI) statistics computed by treating the Web as a massive corpus of text. KNOWITALL leverages existing Web search engines to compute these statistics efficiently. Based on these PMI statistics, KNOWITALL associates a probability with every fact it extracts, enabling it to automatically manage the tradeoff between precision and recall.

**TEXTRUNNER**

Banko, Cafarella, Soderland, Broadhead, and Etzioni (2007) introduce Open Information Extraction (OIE) which is a novel extraction
Automatic Semantic Annotation Using Machine Learning

A paradigm that facilitates domain-independent discovery of relations extracted from text and readily scales to the diversity and size of the Web corpus. The sole input to an OIE system is a corpus, and its output is a set of extracted relations. An OIE system makes a single pass over its corpus guaranteeing scalability with the size of the corpus.

It also introduces TEXTRUNNER, a fully implemented, highly scalable OIE system where the tuples are assigned a probability and indexed to support efficient extraction and exploration via user queries. TEXTRUNNER consists of three key modules:

1. **Self-Supervised Learner**: Given a small corpus sample as input, the Learner outputs a classifier that labels candidate extractions as “trustworthy” or not. The Learner requires no hand-tagged data.
2. **Single-Pass Extractor**: The Extractor makes a single pass over the entire corpus to extract tuples for all possible relations. The Extractor does not utilize a parser. The Extractor generates one or more candidate tuples from each sentence, sends each candidate to the classifier, and retains the ones labeled as trustworthy.
3. **Redundancy-Based Assessor**: The Assessor assigns a probability to each retained tuple based on a probabilistic model of redundancy in text (Downey, Etzioni, and Soderland, 2005).

The Knowledge and Information Management (KIM) platform (Popov et al., 2003) contains an ontology, a knowledgebase, a semantic annotation, an indexing and retrieval server, as well as front-ends for interfacing with the server. For ontology and knowledgebase storage it uses the SESAME RDF repository (Brockstra, Kampman, and Harmelen, 2002), and for search it uses a modified version of Lucene, a keyword-based search engine. The semantic annotation process relies on a pre-built lightweight ontology called KIMO as well as an inter-domain knowledgebase. KIMO defines a base set of entity classes, relationships, and attribute restrictions. The knowledgebase is populated with 80,000 entities consisting of locations and organizations, gathered from a general News corpus. Named-entities found during the annotation process are matched to their type in the ontology and also to a reference in the knowledgebase. The dual mapping allows the information extraction process to be improved by providing disambiguation clues based on attributes and relations (Popov et al., 2003).

The information extraction component of semantic annotation is performed using components of the GATE toolkit (Cunningham, Maynard, Bontcheva, and Tablan, 2002). Some components of GATE have been modified to support the KIM server. Some other components of semantic annotation have been custom developed.

**MUSE**

MUSE (Maynard, 2003) was designed to perform named entity recognition and coreferencing. It has been implemented using the GATE framework. The IE components, called processing resources (PRs), form a processing pipeline used to discover named entities. MUSE executes PRs conditionally based on text attributes. Conditional processing is handled using a Switching Controller, which calls the appropriate PRs in the specified order. The use of conditional processing allows MUSE to obtain accuracies similar to machine learning systems. Semantic tagging is accomplished using the Java Annotations Pattern Engine (JAPE) (Cunningham, Maynard, and Tablen, 2000). Rules using the JAPE grammar are constructed to generate annotations. The Semantic Tagger can use tags generated by processing resources run earlier in the pipeline. For example, if the gazetteer recognizes a first name and the part-of-speech
tagger recognizes a proper noun, a JAPE rule can use both tags to annotate an entity of type Person. The MUSE system is more sophisticated than a gazetteer because a gazetteer cannot provide an exhaustive list of all potential named-entities, and cannot resolve entity ambiguities.

**AeroDAML**

AeroDAML (Kogut and Holmes, 2001) is a knowledge markup tool that applies information extraction techniques to automatically generate DAML annotations from web pages. AeroDAML uses a pattern-based approach to link the most proper nouns and common relationships with classes and properties in DAML (DARPA Agent Markup Language) ontologies (Hendler and McGuinness, 2000). AeroDAML consists of an information extraction module called AeroText™ and components for DAML generation. AeroText™ (Kogut and Holmes, 2001) is designed to support various text processing tasks, and is comprised of four major components: 1) Knowledge Base Compiler for converting linguistic data files into an efficient runtime knowledge base; 2) Knowledge Base Engine for applying the knowledge base to input documents; 3) an IDE for building, testing, and analyzing linguistic knowledge base; and 4) Common Knowledge Base containing general rules for extracting proper nouns and frequently occurring relations.

**Armadillo and MnM**

Armadillo (Dingli, Ciravegna, and Wilks, 2003) and MnM (Vargas-Vera et al., 2002) utilize the Amilcare IE system (Ciravegna, 2001) to perform wrapper induction on web pages. We use Armadillo as the example here. Armadillo uses a pattern-based approach to find entities. It finds its own initial set of seed-patterns rather than requiring an initial set of seeds (Brin, 1998). Manual patterns are used for the named entity recognizer. No manual annotation of corpus documents is required. Once the seeds are found, pattern expansion is then used to discover additional entities. Information redundancy, via queries to Web services such as Google and CiteSeer, is used to verify discovered entities by analyzing query results to confirm or deny the existence of an entity. The use-case implemented in Armadillo is extracting worker details from a university computer science department web site in order to find personal data, such as name, position, home page, email address, and other contact information. The seed-discovery and expansion finds worker names in the web pages. Since many names may be discovered, the Web services are queried to confirm whether a person actually works in the department. The names are then used to discover home pages, where detailed information about a person can often be found and extracted. Armadillo is also interesting in that it attempts to discover citations for each person discovered. The information redundancy approach was also applied to bibliographic entries, but with a lower success rate than discovering and extracting information about people from home pages.

**DIPRE**

Dual Iterative Pattern Expansion (DIPRE) (Brin, 1998) was proposed as an approach for extracting a structured relation (or table) from a collection of HTML documents. The method works well in an environment like the World-Wide-Web, where the table tuples to be extracted will tend to appear in uniform contexts repeatedly in the collection documents (i.e., in the available HTML pages). DIPRE exploits this redundancy and inherent structure in the collection to extract the target relation with minimal training from a user.

**Snowball**

The techniques of Snowball build on the idea of DIPRE. Snowball (Agichtein and Gravano, 2000)
is a bootstrapping-based system that requires only a handful of training examples of interest. These examples are used to generate extraction patterns, which in turn results in new tuples being extracted from the document collection. During each iteration of the extraction process, Snowball evaluates the quality of these patterns and tuples without human intervention, and keeps only the most reliable ones for the next iteration.

**SemTag**

SemTag (Dill et al., 2003) is the semantic annotation component of a comprehensive platform, called Seeker, for performing large-scale annotation of web pages. SemTag performs annotation in three passes: Spotting, Learning, and Tagging. The Spotting examines tokenized words from source documents and finds label matches from the taxonomy. If a label match is found, a window of ten words to either side of the source match is kept. In the Learning pass, a sample of the corpus is examined to find the corpus-wide distribution of terms at each node of the taxonomy. The Tagging pass is then executed, scanning all of the windows from the Spotting pass and disambiguating the matches. Once a match is confirmed, the URL, text reference, and other metadata are stored. SemTag/Seeker is an extensible system, so new annotation implementations can replace the existing Taxonomy-based Disambiguation algorithm (TBD). The taxonomy used by SemTag is TAP. TAP is shallow and covers a range of lexical and taxonomic information about popular items such as music, movies, authors, sports, health and so forth. The annotations generated by SemTag are stored separately from the source document.

**C-PANKOW**

PANKOW (Pattern-based Annotation through Knowledge On the Web) (Cimiano, Handschuh, and Staab, 2004) uses globally available knowledge to annotate resources such as web pages. The core of PANKOW is a pattern generation mechanism which creates pattern strings out of a certain pattern schema conveying a specific semantic relation, an instance to be annotated and all the concepts from a given ontology. It counts the occurrences of these pattern strings on the Web using the Google API. The ontological instance in question is then annotated semantically according to a principle of maximal evidence, i.e. with the concept having the largest number of hits.

C-PANKOW (Context-driven PANKOW) (Cimiano, Ladwig, and Staab, 2005) alleviates several shortcomings of PANKOW. First, by downloading abstracts and processing them off-line, it avoids the generation of large number of linguistic patterns and correspondingly large number of Google queries. Second, by linguistically analyzing and normalizing the downloaded abstracts, it increases the coverage of pattern matching mechanism and overcome several limitations of the earlier pattern generation process. Third, it uses the annotation context in order to distinguish the significance of a pattern match for the given annotation task. C-PANKOW is implemented as a plug-in for OntOMat.

**Summary**

Table 5 gives the comparison of the semantic annotation systems. It shows the methods and the algorithms employed in the systems. We can see from the table that most of the annotation systems focus on dealing with one specific genre of documents or a specific application. For example, SEAN (Mukherjee, Yang, and Ramakrishnan, 2003) aims at annotating documents generated based on a specific template; AeroDAML (Kogut and Holmes, 2001) only supports annotation with the ontology description language DAML; systems like ALPHA (Li and Yu, 2001) and MUMIS (Buitelaar and Declerck, 2003) support annotation of only natural language text.

Based on methods employed in the system, we can see some system only support manual
annotation or rule based annotation (for example (LP) (Ciravegna, 2001) and GATE (Cunningham et al., 2002)); some systems can take advantage of natural language analysis techniques (e.g. ALPHA (Li and Yu, 2001) and MUMIS (Buitelaar and Declerck, 2003)) and statistical learning methods (e.g., SemTag (Dill et al., 2003) and SCORE (Hammond, Sheth, and Kochut, 2002)); and some other systems enable learning from users’ feedbacks or domain knowledge to improve the performance of annotation (e.g., KIM (Popov et al., 2003)).

We can also see from Table 5 that few systems utilize the dependencies between the annotated instances. Although Reeve has investigated the Hidden Markov Model for semantic annotation, the annotation systems have not employed the dependent models in practical applications.

<table>
<thead>
<tr>
<th>System</th>
<th>Method</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>AeroDAML</td>
<td>Manual Rules</td>
<td>AeroText, NLP</td>
</tr>
<tr>
<td>ALPHA</td>
<td>NLP</td>
<td>Linker grammar parser</td>
</tr>
<tr>
<td>Armadillo</td>
<td>Pattern Discovery</td>
<td>LP²</td>
</tr>
<tr>
<td>Artequakt</td>
<td>Manual Rules + NLP</td>
<td>GATE+ Apple Pie Parser</td>
</tr>
<tr>
<td>CREAM/OntoMat</td>
<td>Multiple</td>
<td>Multiple</td>
</tr>
<tr>
<td>Dome</td>
<td>Manual Rules</td>
<td>Remember User Operation</td>
</tr>
<tr>
<td>Esperonto</td>
<td>Rules Learning</td>
<td>Wrapper Induction</td>
</tr>
<tr>
<td>KIM</td>
<td>Manual Rules</td>
<td>GATE</td>
</tr>
<tr>
<td>Melia</td>
<td>Rules Learning</td>
<td>LP²</td>
</tr>
<tr>
<td>MnM</td>
<td>Rules Learning</td>
<td>LP², Badger, Marmot, Crystal</td>
</tr>
<tr>
<td>MUMIS</td>
<td>NLP</td>
<td>ShProT</td>
</tr>
<tr>
<td>MUSE</td>
<td>Manual Rule</td>
<td>GATE</td>
</tr>
<tr>
<td>Ontobroker</td>
<td>Manual</td>
<td>Manual</td>
</tr>
<tr>
<td>C-PANKOW</td>
<td>NLP + Unsupervised Pattern Discovery</td>
<td>Pattern Discovery + Statistical Learning</td>
</tr>
<tr>
<td>SCORE</td>
<td>Classification Model + Statistical Learning</td>
<td>Name entity and relation learning</td>
</tr>
<tr>
<td>S-CREAM</td>
<td>Manual Rules + Learning</td>
<td>LP²</td>
</tr>
<tr>
<td>SEAN</td>
<td>Rules + Webpage Template Analysis</td>
<td>Template Discovery + Semantic Analysis</td>
</tr>
<tr>
<td>SemTag</td>
<td>Manual Rules + Statistical Learning</td>
<td>TBD</td>
</tr>
<tr>
<td>SHOE</td>
<td>Manual</td>
<td>Manual</td>
</tr>
<tr>
<td>WebKB-1</td>
<td>Rules Learning</td>
<td>SRV</td>
</tr>
<tr>
<td>WebKB-2</td>
<td>Rules Learning</td>
<td>Unknown</td>
</tr>
<tr>
<td>KnowItAll</td>
<td>Rules Learning</td>
<td>Bootstrapping + PMI-IR</td>
</tr>
<tr>
<td>KnowItNow</td>
<td>Rules Learning</td>
<td>Binding Engine + URNS</td>
</tr>
<tr>
<td>TextRunner</td>
<td>Rules Learning</td>
<td>Statistical Learning Algorithm</td>
</tr>
<tr>
<td>Snowball</td>
<td>Pattern Discovery</td>
<td>Based on DIPRE</td>
</tr>
<tr>
<td>Gate</td>
<td>Rule Learning</td>
<td>Annie</td>
</tr>
<tr>
<td>ESpotter</td>
<td>Manual Rules + Pattern Discovery</td>
<td>Named Entity Recognition</td>
</tr>
<tr>
<td>T-Rex</td>
<td>Framework</td>
<td>Multiple</td>
</tr>
</tbody>
</table>
Generally speaking, it is still necessary to conduct a thorough investigation of the semantic annotation issue. Many real-world problems require to be solved as the first step for automatic semantic annotation. The major problems include:

1. Lack of analysis of the characteristics of the emerging Web documents. There are a lot of new types of documents, especially with the development of the Web 2.0. The traditional annotation method often focuses on one type of document or application in a specific domain. A comprehensive analysis of characteristics of the documents is thus necessary.

2. The current annotation methods still need improvements. Existing systems usually make use of the rule learning based annotation method. However, the proposed rule learning methods (e.g., LP\textsuperscript{2}; Ciravegna, 2001) still have some problems such as low efficiency and too many parameters needed to tune.

3. Lack of a theoretical model that can efficiently take advantage of dependencies between the annotated instances. The dependencies between the annotated instances can be used to improve the annotation performance. However, dependencies in different types of documents are different (sometimes even vary largely). For example, sometimes the dependencies are linear and sometimes are hierarchical. Therefore, a theoretical model is required for efficiently and effectively incorporating the dependencies.

**APPLICATIONS**

In this section, we introduce several semantic annotation applications that we experienced. We will also introduce some well-known applications in this area.

### Semantic Annotation in Digital Libraries

In digital libraries (DL), “metadata” is structured data for helping users find and process documents and images. With the metadata information, search engines can retrieve required documents more accurately. Scientists and librarians need use great manual efforts and lots of time to create metadata for the documents. To alleviate the hard labor, many efforts have been made toward the automatic metadata generation, based on the techniques of information extraction. Here we take Citeseer, a popular scientific literature digital library, as an example in our explanation.

Citeseer is a public specialty scientific and academic DL that was created in NEC Labs, which is hosted on the World Wide Web at the College of Information Sciences and Technology, The Pennsylvania State University, and has over 800,000 documents, primarily in the fields of computer and information science and engineering (Lawrence, Giles, and Bollacker, 1999; Han et al., 2003). Citeseer crawls and harvests documents on the web, extracts documents metadata automatically, and indexes the metadata to permit querying by metadata.

By extending Dublin Core metadata standard, Citeseer defines 15 different meta-tags for the document header, including Title, Author, Affiliation, and so on. They view the task of automatic document metadata generation as that of labeling the text with the corresponding meta-tags. Each meta-tag corresponds to a metadata class. The annotation task is cast as a classification problem and SVM is employed to perform the classification. They show that classifying each text line into one or more classes is more efficient for meta-tagging than classifying each word, and decompose the metadata extraction problem into two sub-problems: (1) line classification and (2) chunk identification of multi-class lines.

In line classification, both word-level and line-level features are used. Each line is represented
Automatic Semantic Annotation Using Machine Learning

by a set of words and line-specific features. A rule-based, context-dependent word clustering method is developed to overcome the problem of word sparseness. For example, an author line “Chungki Lee James E. Burns” is represented as “CapNonDictWord: :MayName: :MayName: :SingleCap: :MayName”, after word clustering. The weight of a word-specific feature is the number of times this feature appears in the line. And line-specific features are features such as “Number of the words in the line”, “The position of the line”, “The percentage of dictionary words in the line” and so on. The classification process is performed in two steps, an independent line classification followed by an iterative contextual line classification. Independent line classification use the features described above to assign one or more classes to each text line. After that, by making use of the sequential information among lines output by the first step, an iterative contextual line classification is performed. In each iteration, each line uses the previous N and next N lines’ class information as features, concatenates them to the feature vector used in step one, and updates its class label. The procedure converges when the percentage of line with new class labels is lower than a threshold. The principle of the classification based method is the Two-level boundary classification approach as described in Section 2.2.3.

After classifying each line into one or more classes, meta-tag can be assigned to lines that have only one class label. For those that have more than one class label, a further identification is employed to extract metadata from each line. The task is cast as a chunk identification task. Punctuation marks and spaces between words are considered candidate chunk boundaries. A two-class chunk identification algorithm for this task was developed and it yields an accuracy of 75.5%. For lines that have more than two class labels, they are simplified to two-class chunk identification tasks by detecting natural chunk boundary. For instance, using the positions of email and URL in the line, the three-class chunk identification can be simplified as two-class chunk identification task. The position of the email address in the following three-class line “International Computer Science Institute, Berkeley, CA94704. Email: aberer@icsi.berkeley.edu.” is a natural chunk boundary between the other two classes. The method reaches an overall accuracy of 92.9%. See also (Lawrence et al., 1999; Han et al., 2003) for details.

Researcher Profile Extraction

We present a novel expertise oriented search system for web community, which is available at http://www.arnetminer.org (Tang, Zhang, Zhang, Yao, and Zhu, 2007b). Our objective in this system is to provide services for searching and mining the semantic-based web community.

We define a researcher profile ontology, which include basic information (e.g. photo, affiliation, and position), contact information (e.g. address, email, and telephone), educational history (e.g. graduated university and major), and publications. For each researcher, we intend to create a profile based on the ontology by extracting the profile information from his/her homepage or Web pages introducing him/her. Figure 14 shows a researcher’s homepage. It includes typical information in a researcher profile. The top section includes a photo, two addresses, and an email address; the middle section describes the educational history of the researcher; the bottom section provides the position and affiliation information. The ideal annotation result is shown in the right part of Figure 14.

We formalize the problem as that of sequential labeling. Next, we propose a unified approach on the basis of tagging. We view the problem as assigning tags to the input texts, with each tag representing one profile property. As the tagging model, we employ Conditional Random Fields (CRFs). The unified approach can achieve better performance in researcher profiling than the separated methods, because the approach can take advantage of the interdependencies between the
subtasks of profiling. Furthermore, there is no need to define specialized models to annotate different types of properties; all the properties can be extracted in one unified model.

There are three steps in our approach: relevant page finding, preprocessing, and tagging. In relevant page finding, given a researcher name, we first get a list of web pages by a web search engine (i.e. Google) and then identify the homepage or introducing page using a classifier. We view the URL of the identified web page as the value of the Homepage property in the profile.

In preprocessing, (A) we segment the text into tokens and (B) we assign possible tags to each token. The tokens form the basic units and the pages form the sequences of units in the tagging problem. In tagging, given a sequence of units, we determine the most likely corresponding sequence of tags using a trained tagging model. (The tags correspond to the properties defined in the ontology.)

(A). We identify tokens in the Web page heuristically. We define five types of tokens: ‘standard word’, ‘special word’, ‘<image>’ token, term, and punctuation mark. Standard words are unigram words in natural language. Special words (Sprat, Black, Chen, Kumar, Ostendorf, and Richards, 1999) include email address, IP address, URL, date, number, percentage, words containing special symbols (e.g. ‘Ph.D.’, ‘Prof.’), unnecessary tokens (e.g. ‘===’ and ‘####’), etc. We identify special words using regular expressions. ‘<image>’ tokens are ‘<image>’ tags in the HTML file. We identify them by parsing the HTML file. Terms are base noun phrases extracted from the Web pages. We employed the methods proposed in (Xun, Huang, and Zhou, 2000). Punctuation marks include period, question, and exclamation mark.

(B). We assign tags to each token based on their corresponding type. For standard word, we assign all possible tags. For special word, we assign tags: Position, Affiliation, Email, Address, Phone, Fax, and Bsmajor, Msmajor, Phdmajor, Bsuniv, Msuniv, and Phduniv. For ‘<image>’ token, we assign two tags: Photo and Email (it is likely that an email address is shown as an image). For term token, we assign Position, Affiliation, Address, Bsmajor, Msmajor, Phdmajor, Bsuniv, Msuniv, and Phduniv. In this way, each token can be assigned several possible tags. Using the tags, we can perform most of the profiling processing.

Experimental results show that our method can obtain high performance (83.37% in terms of F1-measure) and outperforms the separated clas-
Automatic Semantic Annotation Using Machine Learning

Semantic Annotation for Biomedical Information

Biomedical information, such as the protein or gene name and the biological relations between biomolecules, is presented in unstructured text of biomedical journal articles. Manual annotation requires lots of human efforts for interpreting molecules and the interactions between them. Thus, automatic annotation of entities and relations is extremely useful for biomedical information.

Identifying the names of proteins and genes is a Named Entity Recognition task. Rule-based, classification based, and sequential labeling based methods can all be applied to this task. For example, ABNER is a tool for automatic annotation of genes, proteins, and other entity names from text, which utilizes the Conditional Random Field model.

Extracting the biomedical relationship between proteins (i.e. interaction) is another important step toward understanding the biological organism. Uncovering the complex network of protein interactions is the key aspect of proteomic effort, which is also called PPI. The interactions are always presented in the abstracts of biomedical articles. Researchers have proposed different kinds of methods to solve the problem. Early solutions are rule-based methods, discovering useful patterns. Blaschke, Andrade, Ouzounis, and Valencia (1999) propose to extract patterns with restricted protein names and some verbs describing the interaction to detect protein-protein interaction. Rule-based method can only discover the subset of the interactions. Statistical method can solve the task more efficiently. (Donaldson et al., 2003) designed a system, PreBIND and Textomy for mining the biomedical literature for protein-protein interactions using Support Vector Machines. The abstracts of scientific articles which contain interactions are positive examples, and the abstracts without interactions are negative examples. Then the classifier detects abstracts containing the interactions. This will help reduce human efforts for building the database of protein-protein interactions. Soni (2006) utilized Conditional Random Fields for detecting protein-protein interactions. The interaction detection is cast as sequential labeling – assigning labels to each word. The labels include PAIR-1, PAIR-2 (the two proteins), PAIR-1-2 (the protein involved in two interactions) and the FILTER (the language between the protein pairs. In theory, it should be a phrase which describe the real interaction between the pairs).

FUTURE RESEARCH DIRECTIONS

There are varieties of promising directions for future research in semantic annotation using machine learning.

On the machine-learning side, it would be interesting to generalize the idea of large-margin classification to sequential labeling models, strengthening the extraction results and leading to new optimal learning algorithms with stronger guarantees against overfitting. For example, Taskar (2003) proposes a maximal Markov model for sequential labeling task using the maximal margin theory. It is also attractive to study the new model for annotating the multimedia documents. As the video and audio becomes more and more popular, it would be very useful to study a new model with higher accuracy by considering the text, audio, and video documents simultaneously.

Although much research work has been conducted for automating semantic annotation, there is still a long way before making the dream of “automatic” reality. Many problems needed to be investigated and solved. We here list several challenging problems:

- The traditional supervised machine learning based annotation methods require a
large number of annotated data to train the annotation model. Manually labeling the training data is expensive. How to learn a good annotation model from limited data is a challenging problem. Methods, such as active learning, bootstrapping, and semi-supervised learning, have been proposed. However, the problem is unsolved and still needs further investigation.

- How to deal with the problem of “domain adaptation” is another challenging problem. Many semantic annotation methods can learn good annotation models for a specific domain. However, the annotation models cannot be applied to the other domains. How to solve this problem is a key factor to the development of a real-world semantic annotation system.

- How to learn annotation models in some special cases is also a challenging problem. For example, in researcher profile annotation, a general method is to identify a person homepage first and conduct entity extraction from the identified pages, and then identify relations between the identified entities. The challenge is: Can we propose a unified approach that performs the three separated steps simultaneously? For example, sometimes one kind of entity (e.g. Time) may be easy to be identified while the other entities may be difficult (e.g. Address). How can we use the former one to help identify the annotation of the later ones?

- How to conduct complex semantic annotation is a critical problem for Semantic Web community as well. Conventional semantic annotation tasks are usually aimed at annotating specific type of web pages by a simple metadata or ontology. However, the ontology would be complex in practical applications. In such cases, how to conduct the annotation, especially, relation annotation is really a challenge.

- How to enrich the representation of the text needs to be considered as well. Exiting methods often view a document as a “bag of words”, which is obviously limited. The effectiveness of several attempts by making use of domain knowledge or background knowledge to represent the document is not satisfactory. Furthermore, how to represent the document so as to improve the performance of semantic annotation also need further studies.

- How to develop multimodal techniques for conducting semantic annotation from information encoded in different modalities (text, images, audio, video, and background knowledge. By integrating the processes on all kinds of modalities and conduct the annotations simultaneously can be more accurate and more efficient.

Another interesting also important issue is how to make use of the prior knowledge in semantic annotation. So far, a common method for incorporating the prior knowledge is to use some domain-specific dictionaries (e.g. ontology) in the extraction. The question is whether the simple method still works well when dealing with more complex extraction tasks. A further question is if we can incorporate the different types of prior knowledge into a unified model for extraction.

As another future work, more applications, especially practical applications, need to be investigated. The new applications can provide rich data sources for conducting semantic annotation, as well as bring new challenges to the field. This is because various applications have various characteristics, requiring different methods to deal with.

**CONCLUSION**

Aiming to provide Semantic Web with fundamental semantics, semantic annotation has become an
Automatic Semantic Annotation Using Machine Learning

important sub-discipline of artificial intelligence, language processing, text mining, and Semantic Web. Nowadays, the significance of Semantic Annotation is promoted by the fast growing amount of information available in the unstructured form, for example on the Web.

In this chapter, we have reviewed the existing principled methods for semantic annotation. Specifically, we focus on two most important issues in semantic annotation: entity extraction and relation extraction. For entity extraction, we introduce four state-of-the-art methods: rule based methods, classification based methods, sequential labeling based methods, and non-linear based Markov random fields based methods. For relation extraction, we also introduce four typical methods: classification based methods, kernel based method, sequential labeling based methods, and other methods. We have explained the principle of these methods by using several approaches as examples. We have described several annotation systems and compared the main features and the algorithms employed in the systems. Moreover, we have introduced several practical application of semantic annotation.

ACKNOWLEDGMENT

The work is supported by the National Natural Science Foundation of China (90604025, 60703059), Chinese National Key Foundation Research and Development Plan (2007CB310803), and Chinese Young Faculty Funding (20070003093).

Thanks to the anonymous reviewers for their constructive suggestions.

REFERENCES


Ciravegna, F. (2001). (LP)², an adaptive algorithm for information extraction from Web-related texts. In Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with 17th International Joint Conference on Artificial Intelligence (IJCAI), Seattle, USA.


Freitag, D. (1998). Information extraction from HTML: Application of a general machine learn-
Automatic Semantic Annotation Using Machine Learning


Automatic Semantic Annotation Using Machine Learning


Sutton, C., Rohanimanesh, K., & McCallum, A. (2004). Dynamic conditional random fields:
Automatic Semantic Annotation Using Machine Learning


**ADDITIONAL READING**


Automatic Semantic Annotation Using Machine Learning


the 23rd International Conference on Machine Learning (ICML2006)


