Information Extraction: Methodologies and Applications

Jie Tang, Mingcai Hong, Duo Zhang, Bangyong Liang, and Juanzi Li

Jie Tang (corresponding author)
Affiliation: Department of Computer Science, Tsinghua University
Telephone: +8610-62788788-20 Fax number: +8610-62789831
E-mail: jietang@tsinghua.edu.cn
Post mail address: 10-201, East Main Building, Tsinghua University, Beijing, 100084. China.

Mingcai Hong
Affiliation: Department of Computer Science, Tsinghua University
Telephone: +8610-62788788-20 Fax number: +8610-62789831
E-mail: hmc@keg.cs.tsinghua.edu.cn
Post mail address: 10-201, East Main Building, Tsinghua University, Beijing, 100084. China.

Duo Zhang
Affiliation: Department of Computer Science, Tsinghua University
Telephone: +8610-62788788-20 Fax number: +8610-62789831
E-mail: zhangduo@keg.cs.tsinghua.edu.cn
Post mail address: 10-201, East Main Building, Tsinghua University, Beijing, 100084. China.

Bangyong Liang
Affiliation: NEC Labs China
Telephone: +86013601002822 Fax number: +8610-62789831
E-mail: liangbangyong@research.nec.com.cn
Post mail address: 11th Floor, Innovation Plaza, Tsinghua Science Park, Beijing, 100084, China

Juanzi Li
Affiliation: Department of Computer Science, Tsinghua University
Telephone: +8610-62781461 Fax number: +8610-62789831
E-mail: ljz@keg.cs.tsinghua.edu.cn
Post mail address: 10-201, East Main Building, Tsinghua University, Beijing, 100084. China.

Keyword List

- Computer Science
  - Data Resource Management
    - Data Extraction
    - Data Management
    - Data Mining
    - Knowledge Discovery
  - Software
    - Natural Language Processors

- Information Systems
  - Information Theory
    - Information Processing

Information Extraction: Methodologies and Applications

Jie Tang[1], Mingcai Hong[1], Duo Zhang[1], Bangyong Liang[2], and Juanzi Li[1]

[1]Department of Computer Science, Tsinghua University
10-201, East Main Building, Tsinghua University, Beijing, 100084. China

[2]NEC Labs China
11th Floor, Innovation Plaza, Tsinghua Science Park, Beijing, 100084. China
liangbangyong@research.nec.com.cn

ABSTRACT

This chapter is concerned with the methodologies and applications of information extraction. Information is hidden in the large volume of web pages and thus it is necessary to extract useful information from the web content, called Information Extraction. In information extraction, given a sequence of instances, we identify and pull out a sub-sequence of the input that represents information we are interested in. In the past years, there was a rapid expansion of activities in the information extraction area. Many methods have been proposed for automating the process of extraction. However, due to the heterogeneity and the lack of structure of Web data, automated discovery of targeted or unexpected knowledge information still presents many challenging research problems. In this chapter, we will investigate the problems of information extraction and survey existing methodologies for solving these problems. Several real-world applications of information extraction will be introduced. Emerging challenges will be discussed.

INTRODUCTION

Information Extraction (IE), identifying and pulling out a sub-sequence from a given sequence of instances that represents information we are interested in, is an important task with many practical applications. Information extraction benefits many text/web applications, for example, integration of product information from various websites, question answering, contact information search, finding the proteins mentioned in a biomedical journal article, and removal of the noisy data.

Our focus will be on methodologies of automatic information extraction from various types of documents (including plain texts, web pages, and emails, etc.). Specifically, we will discuss three of the most popular methods: rule learning based method, classification model based method, and sequential labeling based method. All these methods can be viewed as supervised machine learning approaches. They all consist of two stages: extraction and training.

In extraction, the sub-sequence that we are interested in are identified and extracted from given data using learned model(s) by different methods. Then the extracted data are annotated as specified information on the basis of the predefined metadata.

In training, the model(s) are constructed to detect the sub-sequence. In the models, the input data is viewed as a sequence of instances, for example, a document can be viewed as either a sequence of words or a sequence of text lines (it depends on the specific application).

All these methodologies have immediate real-life applications. Information extraction has been applied, for instance, to part-of-speech tagging (Ratnaparkhi, 1998), named entity recognition (Zhang, 2004), shallow parsing (Sha, 2003), table extraction (Ng, 1999; Pinto, 2003; Wang, 2002), and contact information extraction (Kristjansson, 2004).

In the rest of the chapter, we will describe the three types of the state-of-the-art methods for information extraction. This is followed by presenting 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 the challenges the information extraction community faced. Finally, we will give the concluding remark.

METHODOLOGIES

Information extraction is an important research area, and many research efforts have been made so far. Among these research work, rule learning based method, classification based method, and sequential labeling based method are the three state-of-the-art methods.

Rule Learning based Extraction Methods

In this section, we review the rule based algorithms for information extraction. Numerous information systems have been developed based on the method, including: AutoSlog (Riloff, 1993), Crystal (Soderland, 1995), $(LP)^2$ (Ciravegna, 2001), iASA (Tang, 2005b), Whisk (Soderland, 1999), Rapier (Califf, 1998), SRV (Freitag, 1998), WIEN (Kushmerick, 1997), Stalker (Muslea, 1998; Muslea, 1999a), BWI (Freitag, 2000), etc. See (Muslea, 1999b; Siefkes, 2005; Peng, 2001) for an overview. In general, the methods can be grouped into three categories: dictionary based method, rule based method, and wrapper induction.

Dictionary based method

Traditional information extraction systems first construct a pattern (template) dictionary, and then use the dictionary to extract needed information from the new untagged text. These extraction systems are called as dictionary based systems (also called pattern based systems) including: AutoSlog (Riloff, 1993), AutoSlog-TS (Riloff, 1996), and CRYSTAL (Soderland, 1995). The key point in the systems is how to learn the dictionary of patterns that can be used to identify the relevant information from a text.

AutoSlog (Riloff, 1993) was the first system to learn text extraction dictionary from training examples. AutoSlog builds a dictionary of extraction patterns that are called *concept nodes*. Each AutoSlog concept node has a conceptual anchor that activates it and a linguistic pattern, which, together with a set of enabling conditions, guarantees its applicability. The conceptual anchor is a triggering word, while the enabling conditions represent constraints on the components of the linguistic pattern.

For instance, in order to extract the target of the terrorist attack from the sentence

The Parliament was bombed by the guerrillas.

One can use a concept that consists of the triggering word *bombed* together with the linguistic pattern <subject> passive-verb. Applying such an extraction pattern is straightforward: first, the concept is activated because the sentence contains the triggering word *bombed*; then the linguistic pattern is matched against the sentence and the subject is extracted as the *target* of the terrorist attack.

AutoSlog uses a predefined set of 13 linguistic patterns; the information to be extracted can be one of the following syntactic categories: *subject*, *direct object*, or *noun phrase*. In general, the triggering word is a verb, but if the information to be extracted is a noun phrase, the triggering word may also be a noun.

In Figure 1, we show a sample concept node. The slot “Name” is a concise, human readable description of the concept. The slot “Trigger” defines the conceptual anchor, while the slot “Variable Slots” represents that the information to be extracted is the subject of the sentence. Finally, the subject must be a physical target (see “Constraints:”), and the enabling conditions require the verb to be used in its passive form.

Concept Node:

Name:	target-subject-passive-verb-bombed
Trigger:	bombed
Variable Slots:	(target (*S *1))
Constraints:	(class phys-target *S*)
Constant Slots:	(type bombing)
Enabling Conditions:	((passive))

Figure 1. Example of AutoSlog concept node

AutoSlog needs to parse the natural language sentence using a linguistic parser. The parser is used to generate syntax elements of a sentence (such as subject, verb, preposition phrase). Then the output syntax elements are matched against the linguistic pattern and fire the best matched pattern as the result pattern to construct a pattern dictionary.

AutoSlog need tag the text before extracting patterns. This disadvantage has been improved by AutoSlog-TS (Riloff, 1996). In AutoSlog-TS, one does not need to make a full tag for the input data and only needs to tag the data whether it is relevant to the domain or not. The procedure of AutoSlog is divided into two stages. In the first stage, the sentence analyzer produces a syntactic analysis for each sentence and identifies the noun phrases using heuristic rules. In the second stage, the pre-classified text is inputted to the sentence analyzer again with the pattern dictionary generated in the first stage. The sentence analyzer activates all the patterns that are applicable in each sentence. The system then computes relevance statistics for each pattern and uses a rank function to rank the patterns. In the end, only the top patterns are kept in the dictionary.

Riloff et al (1999) propose using bootstrapping to generate the dictionary with a few tagged texts (called seed words). The basic idea is to use a mutual bootstrapping technique to learn extraction patterns using the seed words and then exploit the learned extraction patterns to identify more seed words that belong to the same category. In this way, the pattern dictionary can be learned incrementally as the process continues. See also Crystal (Soderland, 1995).

Rule based method

Different from the dictionary based method, the rule based method use several general rules instead of dictionary to extract information from text. The rule based systems have been mostly used in information extraction from semi-structured web page.

A usual method is to learn syntactic/semantic constraints with delimiters that bound the text to be extracted, that is to learn rules for boundaries of the target text. Two main rule learning algorithms of these systems are: bottom-up method which learns rules from special cases to general ones, and top-down method which learns rules from general cases to special ones. There are proposed many algorithms, such as $(LP)^2$ (Ciravegna, 2001), iASA (Tang, 2005b), Whisk (Soderland, 1999), Rapier (Califf, 1998), and SRV (Freitag, 1998). Here we will take $(LP)^2$ and iASA as examples in our explanation.

$(LP)^2$

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

Three types of rules are defined in $(LP)^2$: tagging rules, contextual rules, and correction rules.

A tagging rule is composed of a pattern of conditions on a connected 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 in a dictionary, and Name Entity Recognition results of the word sequence respectively. The last column represents the action. In the example of 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.

Table 1. Example of initial tagging rule

Pattern					Action
Word	**POS**	**Kind**	**Lookup**	**Name Entity**	
;	:	Punctuation			
Patrick	NNP	Word	Person's first name	Person	<Speaker>
Stroh	NNP	Word			
,	,	Punctuation			
assistant	NN	Word	Job title		
professor	NN	Word			
,	,	Punctuation			
SDS	NNP	Word			

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 extracted 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 DIGIT pm). Each generalized rule is tested on the training corpus and an error score *E=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 a generalized tagging rule for the start boundary identification of the Speaker.

Table 2. Example of generalized tagging rule

Pattern					Action
Word	**POS**	**Kind**	**Lookup**	**Name Entity**	
;	:	Punctuation			
		Word	Person's first name	Person	<Speaker>
		Word			
		Punctuation			
assistant	NN	Word	Jobtitle		
professor	NN	Word			

Another type of rules, contextual rules, is applied to improve the effectiveness of the system. The basic idea is that <tag$_x$> might be used as an indicator of the occurrence of <tag$_y$>. 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 <speaker>. Consequencely, 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 <time> 4 </time> 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.

Table 3. Example of correction rule

Pattern		Action
Word	**Wrong tag**	**Move tag to**
At		
4	</stime>	
pm		</stime>

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. <time>) without its corresponding close tag will be removed, and vice versa.

See also Rapier (Califf, 1998; Califf, 2003) for another IE system which adopts the bottom-up learning strategy.

iASA

Tang et al (2005b) propose an algorithm for learning rules for information extraction. The key idea of iASA is that it tries to induce the 'similar' rules first. In iASA, each rule consists of three patterns: body pattern, left pattern, and right pattern, respectively representing the text fragment to be extracted (called target instance), the *w* words previous to the target instance, and *w* words next to the target instance. Thus, the rule learning tries to find patterns not only in the context of a target instance, but also in the target instance itself. Tang et al define similarity between tokens (it can be word, punctuation, and name entity), similarity between patterns, and similarity between rules. In learning, iASA creates an initial rule set from the training data set. Then it searches for the most similar rules from the rule set and generalizes a new rule using the two rules. The new rule is evaluated on the training corpus and a score of the rule is calculated. If its score exceeds a threshold, it would be put back to the rule set. The processing continues until no new rules can be generalized.

The other type of strategy for learning extraction rules is the top-down fashion. The method starts with the most generalized patterns and then gradually adds constraints into the patterns in the learning processing. See SRV (Freitag, 1998) and Whisk (Soderland, 1999) as examples.

Wrapper induction

Wrapper induction is another type of rule based method which is aimed at structured and semi-structured documents such as web pages. A wrapper is an extraction procedure, which consists of a set extraction rules and also program codes required to apply these rules. Wrapper induction is a technique for automatically learning the wrappers. Given a training data set, the induction algorithm learns a wrapper for extracting the target information. Several research works have been studied. The typical wrapper systems include WIEN (Kushmerick, 1997), Stalker (Muslea, 1998), and BWI (Freitag, 2000). Here, we use WIEN and BWI as examples in explaining the principle of wrapper induction.

WIEN

WIEN (Kushmerick, 1997) is the first wrapper induction system. An example of the wrapper defined in WIEN is shown in Figure 2, which aims to extract "Country" and "Area Code" from the two HTML pages: D1 and D2.

```
D1: <B>Congo</B> <I>242</I><BR>
D2: <B>Egypt</B> <I>20</I><BR>

Rule: *'<B>'(*)'</B>'*'<I>'(*)'</I>'
Output: Country_Code {Country@1}{AreaCode@2}
```

Figure 2. Example of wrapper induction

The rule in Figure 2 has the following meaning: ignore all characters until you find the first occurrence of '<B>' and extract the country name as the string that ends at the first '</B>'. Then ignore all characters until '<I>' is found and extract the string that ends at '</I>'. In order to extract the information about the other country names and area codes, the rule is applied repeatedly until it fails to match. In the example of Figure 2, we can see that the WIEN rule can be successfully to be applied to both documents D1 and D2.

The rule defined above is an instance of the so called LR class. A LR wrapper is defined as a vector $<l_1, r_1, \ldots, l_k, r_k>$ of $2K$ delimiters, with each pair $<l_i, r_i>$ corresponding to one type of information. The LR wrapper requires that resources format their pages in a very simple manner. Specifically, there must exist delimiters that reliably indicate the left- and right-hand boundaries of the fragments to be extracted. The classes HLRT, OCLR, and HOCLRT are extensions of LR that use document *head* and *tail* delimiters, tuple delimiters, and both of them, respectively. The algorithm for learning LR wrappers (i.e. learn_{LR}) is shown in Figure 3.

In Figure 3, E represents the example set; notation $\text{cands}_l(k, E)$ represent the candidates for delimiter l_k given the example set E. The candidates are generated by enumerating the suffixes of the shortest string occurring to the left of each instance of attribute k in each example; $\text{valid}_l(u, k, E)$ refers to the constraints to validate a candidate u for delimiter l_k.

```
procedure learn_LR(examples $E$)
{
    for each $1 \leq k \leq K$
        for each $u \in \text{cands}_l(k, E)$
            if $\text{valid}_l(u, k, E)$ then $l_k \leftarrow u$ and terminate this loop
    for each $1 \leq k \leq K$
        for each $u \in \text{cands}_r(k, E)$
            if $\text{valid}_r(u, k, E)$ then $r_k \leftarrow u$ and terminate this loop
    return LR wrapper $<l_1, r_1, \ldots, l_k, r_k>$
}
```

Figure 3. The learn$_{\text{LR}}$ algorithm

LR wrapper class is the simplest wrapper class. See (Kushmerick, 2000) for variant wrapper classes.

Stalker (Muslea, 1998; Muslea, 1999a) is another wrapper induction system that performs hierarchical information extraction. It can be used to extract data from such documents with multiple levels. In Stalker, rules are induced by a covering algorithm which tries to generate rules until all instances of an item are covered and returns a disjunction of the found rules. A Co-Testing approach has been also proposed to support active learning in Stalker. See (Muslea, 2003) for details.

BWI

The Boosted Wrapper Induction (BWI) system (Freitag, 2000) targets at making wrapper induction techniques suitable for free text, which uses boosting to generate and combine the predictions from numerous extraction patterns.

In BWI, a document is treated as a sequence of tokens, and the IE task is to identify the boundaries of different type of information. Let indices i and j denote the boundaries, we can use $<i, j>$ to represent an instance.

A wrapper $W = <F, A, H>$ learned by BWI consists of two sets of patterns that are used respectively to detect the start and the end boundaries of an instance. Here $F = \{F_1, F_2, \ldots, F_T\}$ identifies the start boundaries and $A = \{A_1, A_2, \ldots, A_T\}$ identifies the end boundaries; and a length function $H(k)$ that estimates the maximum-likelihood probability that the field has length k.

To perform extraction using the wrapper W, every boundary i in a document is first given a "start" score $F(i) = \sum_k C_{F_k} F_k(i)$ and an "end" score $A(i) = \sum_k C_{A_k} A_k(i)$. Here, C_{F_k} is the weight for F_k, and $F_k(i) = 1$ if i matches F_k, otherwise $F_k(i) = 0$. For $A(i)$, the definition is similar. W then classifies text fragment $<i, j>$ as follows:

$$W(i, j) = \begin{cases} 1 & \text{if } F(i)A(j)H(j-i) > \tau \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where τ is a numeric threshold.

Learning a wrapper W involves determining F, A, and H.

The function H reflects the prior probability of various field lengths. BWI estimates these probabilities by constructing a frequency histogram $H(k)$ recording the number of fields of length *k* occurring in the training set. To learn *F* and *A*, BWI boosts *LearnDetector*, an algorithm for learning a single detector. Figure 4 shows the learning algorithm in BWI.

```
procedure BWI (example sets S and E)
{
    $F \leftarrow$ AdaBoost(LearnDetector, S)
    $A \leftarrow$ AdaBoost(LearnDetector, E)
    $H \leftarrow$ field length histogram from S and E
    return wrapper $W = <F, A, H>$
}
```

Figure 4. The BWI algorithm

In BWI, AdaBoost algorithm runs in iterations. In each iteration, it outputs a weak learner (called hypotheses) from the training data and also a weight for the learner representing the percentage of the correctly classified instances by applying the weak learner to the training data. AdaBoost simply repeats this learn-update cycle *T* times, and then returns a list of the learned weak hypotheses with their weights. BWI invokes LearnDetector (indirectly through AdaBoost) to learn the “fore” detectors *F*, and then *T* more times to learn the “aft” detectors *A*. LearnDetector iteratively builds from a empty detector. At each step, LearnDetector searches for the best extension of length L (a lookahead parameter) or less to the prefix and suffix of the current detector. The procedure returns when no extension yields a better score than the current detector. More detailed experiments and results analysis about BWI is discussed in (Kauchak, 2004).

Classification based Extraction Methods

In this section, we introduce another principled approach to information extraction using supervised machine learning. The basic idea is to cast information extraction problem as that of classification. In this section, we will describe the method in detail. We will also introduce several improving efforts to the approach.

Classification model

Let us first consider a two class classification problem. Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training data set, in which x_i denotes an instance (a feature vector) and $y_i \in \{-1, +1\}$ denotes a classification label. A classification model usually consists of two stages: learning and prediction. 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. (In some cases, the prediction results may be numeric values, e.g. ranging from 0 to 1. Then an instance can be classified using some rules, e.g. classified as +1 when the prediction value is larger than 0.5.)

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

Support vector machines (SVMs) are linear functions of the form $f(x) = w^Tx + b$, where w^Tx 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:

$$\begin{gathered} Minimize: \frac{1}{2} w^T w \\ s.t.: y_i(w^T x_i + b) \geq 1, i = 1, 2, \ldots, n \end{gathered} \tag{2}$$

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

$$\begin{gathered} Minimize: \frac{1}{2} w^T w + C\sum_{i=1}^{n} \xi_i \\ s.t.: y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, n \end{gathered} \tag{3}$$

where $C \geqslant 0$ 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, 1992; Schölkopf, 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 IE, so our system consists of two distinct phases: learning and extracting. In the learning phase our 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 formalizes the IE problem as a classification problem. It is aimed at detecting the boundaries (start boundary and end boundary) of a special type of information. For IE from text, the basic unit that we are dealing with can be tokens or text-lines in the text. (Hereafter, we will use token as the basic unit in our explanation.) Then 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.

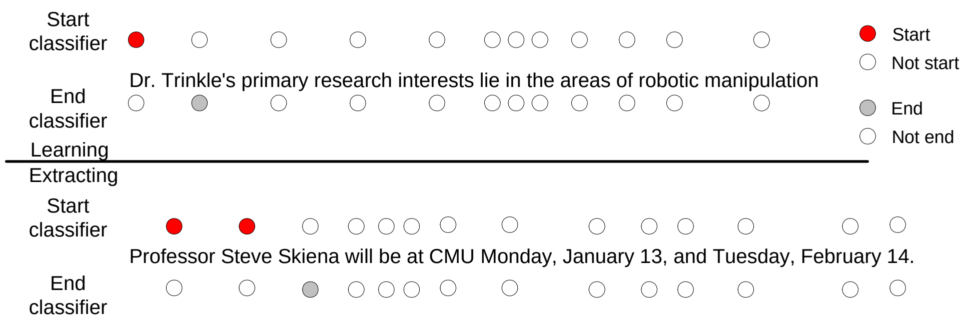

Start classifier
Dr. Trinkle's primary research interests lie in the areas of robotic manipulation
End classifier
Learning
Extracting
Start classifier
Professor Steve Skiena will be at CMU Monday, January 13, and Tuesday, February 14.
End classifier
Start
Not start
End
Not end

Figure 5. Example of Information Extraction as classification

Figure 5 gives an example of IE 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 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 starts and only one end (as the example in Figure 5), 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 method may not yield good results.

Several works have been conducted to enhance the combination. For example, Finn et al propose a histogram model (Finn, 2004; Finn, 2006). In Figure 5, there are two possible extractions: “Professor Steve Skiena” and “Steve Skiena”. The histogram model estimates confidence as $C_s * C_e * P(|e - s|)$. Here C_s is the confidence of the start prediction and C_e is the confidence of the end prediction. (For example, with Naïve Bayes, we can use the posterior probability as the confidence; with SVM, we can use the distance of the instance to the hyper-plane as the confidence.) $P(|e - s|)$ is the probability of a text fragment of that length which we get from the training data.

Finally, the method selects the text fragment with the highest confidence as output.

To summarize, this IE classification approach simply learns to detect the start and the end of text fragments to be extracted. It treats IE 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, 2004).

Many approaches can be used to training the classification models, for example, Support Vector Machines (Vapnik, 1998), Maximum Entropy (Berger, 1996), Adaboost (Shapire, 1999), and Voted Perceptron (Collins, 2002).

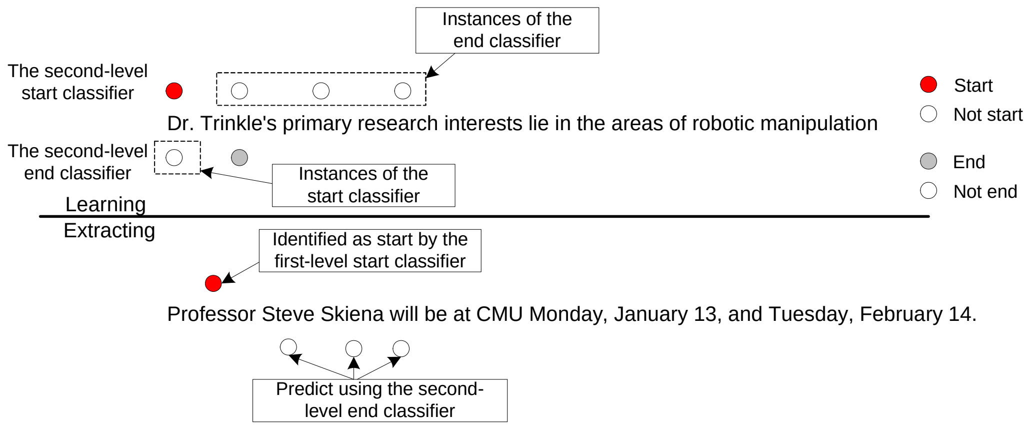
Enhancing IE by a two-level boundary classification model

Experiments on many data sets and in several real-world applications show that the one-level boundary classification approach can competitive with the start-of-the-art rule learning based IE systems. However, as the classifiers are built on a very large number of negative instances and a small number of positive instances, the prior probability that an arbitrary instance is a boundary is very small. This gives a model that has very high precision. Because the prior probability of predicting a tag is so low, and because the data is highly imbalanced, when we actually do prediction for a tag, it is very likely that the prediction is correct. The one-level model is therefore much more likely to produce false negatives than false positives (high precision).

To overcome the problem, a two-level boundary classification approach has been proposed by (Finn, 2004). The intuition behind the two-level approach is as follows. At the first level, the start and end classifiers have high precision. To make a prediction, both the start classifier and the end classifier have to predict the start and end respectively. In many cases where we fail to extract a fragment, one of these classifiers made a prediction, but not the other. The second level assumes that these predictions by the first level are correct and is designed to identify the starts and ends that we failed to identify at the first level.

The second-level models are learned from training data in which the prior probability that a given instance is a boundary is much higher than for the one-level learner. This “focused” training data is constructed as follows. When building the second-level start model, we take only the instances that occur a fixed distance before an end tag. Similarly, for the second-level end model, we use only instances that occur a fixed distance after a start tag. For example, an second-level window of size 10 means that the second-level start model is built using only 10 instances that occur before an end-tag in the training data, while the second-level end model is built using only those instances that occur in the 10 instances after a start tag in the training data. Note that these second-level instances are encoded in the same way as for the first-level; the difference is simply that the second-level learner is only allowed to look at a small subset of the available training data. Figure 6 shows an example of the IE using the two-level classification models.

In the example of Figure 6, there are also two stages: learning and extracting. In learning, the tokens “Dr.” and “Trinkle’s” are the start and the end boundaries of a speaker respectively. For training the second-level start and end classifiers. We use window size as three and thus three instances after the start are used to train the end classifier and three instances before the end are used to train the start classifier. In the example, the three tokens “Trinkle’s”, “primary”, and “research” are instances of the second-level end classifier and the token “Dr.” is an instance of the second-level start classifier. Note, in this way, the instances used for training the second-level classifiers are only a subset of the instances for training the first-level classifiers. These second-level instances are encoded in the same way as for the first-level. When extracting, the second-level end classifier is only applied to the three tokens following the token which the first-level classifier predicted as a start and the token itself. Similarly the second-level start classifier is only applied to instances predicted as an end by the first-level classifier and the three preceding tokens.



Instances of the end classifier
The second-level start classifier
Start
Not start
Dr. Trinkle's primary research interests lie in the areas of robotic manipulation
The second-level end classifier
Instances of the start classifier
End
Not end
Learning
Extracting
Identified as start by the first-level start classifier
Professor Steve Skiena will be at CMU Monday, January 13, and Tuesday, February 14.
Predict using the second-level end classifier

Figure 6. Example of Information Extraction by the two-level classification models

In the exacting stage of the example, the token “Professor” is predicted as a start by the first-level start classifier and no token is predicted as the end in the first-level model. Then we can use the second-level end classifier to make prediction for the three following tokens.

This second-level classification models are likely to have much higher recall but lower precision. If we were to blindly apply the second-level models to the entire document, it would generate a lot of false positives. Therefore, the reason we can use the second-level models to improve performance is that we only apply it to regions of documents where the first-level models have made a prediction. Specifically, during extraction, the second-level classifiers use the predictions of the first-level models to identify parts of the document that are predicted to contain targets.

Figure 7 shows the extracting processing flow in the two-level classification approach. Given a set of documents that we want to extract from, we convert these documents into a set of instances and then apply the first-level models for start and end to the instances and generate a set of predictions for starts and ends. The first-level predictions are then used to guide which instances we need apply the second-level classifiers to. We use the predictions of the first-level end model to decide which instances to apply the second-level start model to, and we use the predictions of the first-level start model to decide which instances to apply the second-level end model to. Applying the second-level models to the selected instances gives us a set of predictions which we pass to the combination to output our extracted results.

The intuition behind the two-level approach is that we use the unmatched first-level predictions (i.e. when we identify either the start or the end but not the other) as a guide to areas of text that we should look more closely at. We use more focused classifiers that are more likely to make a prediction on areas of text where it is highly likely that an unidentified fragment exists. These classifiers are more likely to make predictions due to a much lower data imbalance so they are only applied to instances where we have high probability of a fragment existing. As the level of imbalance falls, the recall of the model rises while precision falls. We use the second-level classifiers to lookahead/lookback instances in a fixed windows size and obtain a subset of the instances in the first-level classifier.

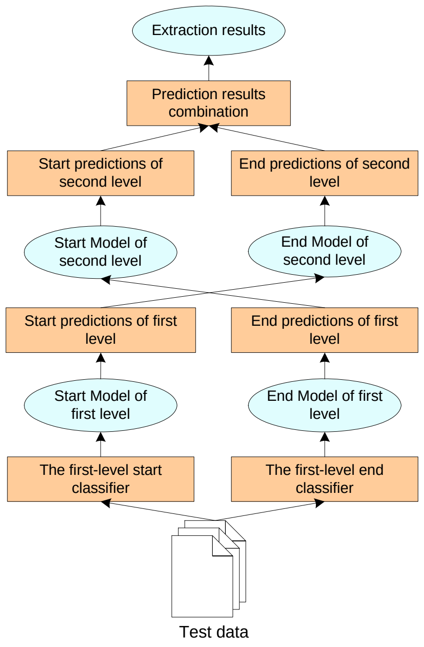

Extraction results
Prediction results combination
Start predictions of second level
End predictions of second level
Start Model of second level
End Model of second level
Start predictions of first level
End predictions of first level
Start Model of first level
End Model of first level
The first-level start classifier
The first-level end classifier
Test data

Figure 7. Extracting processing flow in the two-level classification approach

This enables us to improve recall without hurting precision by identifying the missing complementary tags for orphan predictions. If we have 100% precision at first-level prediction then we can improve recall without any corresponding drop in precision. In practice, the drop in precision is proportional to the number of incorrect prediction at the first-level classification.

Enhancing IE by unbalance classification model

Besides the two-level boundary classification approach, we introduce another approach to deal with the problem so as to improve performance of the classification based method.

As the classifiers are built on a very large number of negative instances and a small number of positive instances, the prior probability that an arbitrary instance is a boundary is very small. This gives a model that has very high precision, but low recall. In this section, we introduce an approach to the problem using an unbalanced classification model. The basic idea of the approach is to design a specific classification method that is able to learn a better classifier on the unbalanced data.

We have investigated the unbalanced classification model of SVMs (Support Vector Machines). Using the same notations in Section 2.2.1, we have the unbalanced classification model:

$$Minimize: \frac{1}{2} w^T w + C_1 \sum_{i=1}^{n_+} \xi_i + C_2 \sum_{i=1}^{n_-} \xi_i \qquad (4)$$
$$s.t.: y_i (w^T x_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots, n$$

here, C_1 and C_2 are two cost parameters used to control the training errors of positive examples and negative examples respectively. For example, with a larger C_1 and a small C_2, we can obtain a classification model that attempts penalize false positive examples more than false negative examples. The model can actually increase the probability of examples to be predicted as positive, so that we can improve the recall while likely hurting the precision, which is consistent with the method of two-level classification. Intuition shows that in this way we can control the trade-off between the problem of high precision and low recall and the training errors of the classification model. The model obtained by this formulation can perform better than the classical SVM model in the case of a large number of negative instances and a small number of positive instances. To distinguish this formulation from the classical SVM, we call the special formulation of SVM as Unbalanced-SVM. See also (Morik, 1999; Li, 2003) for details.

Unbalanced-SVM enables us to improve the recall by adjusting the two parameters. We need to note that the special case of SVM might hurt the precision while improving the recall. The most advantage of the model is that it can achieve a better trade-off between precision and recall.

Sequential Labeling based Extraction Methods

Information extraction can be cast as a task of sequential labeling. In sequential labeling, a document is viewed as a sequence of tokens, and a sequence of labels are assigned to each token to indicate the *property* of the token. For example, consider the nature language processing task of labeling words of a sentence with their corresponding Part-Of-Speech (POS). In this task, each word is labeled with a tag indicating its appropriate POS. Thus the inputting sentence "Pierre Vinken will join the board as a nonexecutive director Nov. 29." will result in an output as:

[NNP Pierre] [NNP Vinken] [MD will] [VB join] [DT the] [NN board]
[IN as] [DT a] [JJ nonexecutive] [NN director] [NNP Nov.] [CD 29] [. .]

Formally, given an observation sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, the information extraction task as sequential labeling is to find a label sequence $\boldsymbol{y}^* = (y_1, y_2, \ldots, y_n)$ that maximizes the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$, i.e.,

$$\boldsymbol{y}^* = \text{argmax}_{\boldsymbol{y}}\, p(\boldsymbol{y}|\boldsymbol{x}) \qquad (5)$$

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, 1997), Maximum Entropy Markov Model (McCallum, 2000), and Conditional Random Field (Lafferty, 2001) are widely used sequential labeling models.

For example, a discrete Hidden Markov Model is defined by a set of output symbols **X** (e.g. a set of words in the above example), a set of states **Y** (e.g. a set of POS in the above example), a set of probabilities for transitions between the states $p(y_i|y_j)$, and a probability distribution on output symbols for each state $p(x_i|y_i)$. An observed sampling of the process (i.e. the sequence of output symbols, e.g. "Pierre Vinken will join the board as a nonexecutive director Nov. 29." in the above example) is produced by starting from some initial state, transitioning from it to another state, sampling from the output distribution at that state, and then repeating these latter two steps. The best label sequence can be found using Viterbi algorithm.

Generative model

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

$$y^* = \arg\max_y p(y \mid x) = \arg\max_y \frac{p(x, y)}{p(x)} \qquad (6)$$

Hidden Markov Models (HMMs) (Ghahramani, 1997) are one of the most common generative models currently used. In HMMs, each observation sequence is considered to have been 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 POS tagging, each state of the HMM is associated with a POS tag. Although POS tags do not generate words, the 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 POS 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.

The states in an HMM are considered to be hidden because of the doubly stochastic nature of the process described by the model. For any observation sequence, the sequence of states that best accounts for that observation sequence is essentially hidden from an observer and can only be viewed through the set of stochastic processes that generate an observation sequence. The principle of identifying the most state sequence that best accounts for an observation sequence forms the foundation underlying the use of finite-state models for labeling sequential data.

Formally, an HMM is fully defined by

- A finite set of states Y.
- A finite output alphabet X.
- A conditional distribution $p(y'|y)$ representing the probability of moving from state y to state y', where $y, y' \in$ Y.

- An observation probability distribution $p(x|y)$ representing the probability of emitting observation x when in state y, where $x \in X$, $y \in Y$.
- An initial state distribution $p(y)$, $y \in Y$.

From the definition of HMMs, we can see that the probability of the state at time t depends only on the state at time t-1, and the observation generated at time t only depends on the state of the model at time t. Figure 8 shows the structure of a HMM.

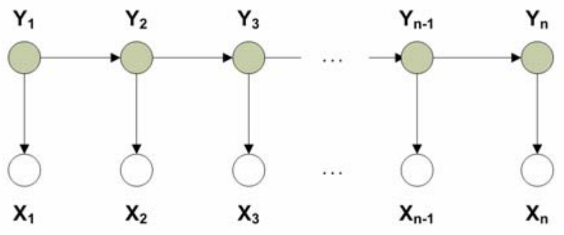


Figure 8. Graphic structure of first-order HMMs

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:

$$p(y,x) = p(y_1)p(x_1 \mid y_1)\prod_{t=2}^{n} p(y_t \mid y_{t-1})p(x_t \mid y_t) \qquad (7)$$

In supervised learning, the conditional probability distribution $p(y_t|y_{t-1})$ and observation probability distribution $p(x|y)$ 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.

Limitations of generative models. Generative models define a joint probability distribution $p(\mathbf{X}, \mathbf{Y})$ over observation and label sequences. This is useful if the trained model is to be used to generate data. However, to define a joint probability over observation and label sequences, a generative model needs to enumerate all possible observation sequences, typically requiring a representation in which observations are task-appropriate atomic entities, such as words or nucleotides. In particular, it is not practical to represent multiple interacting features or long-range dependencies of the observations, since the inference problem for such models is intractable. Therefore, generative models must make strict independence assumptions in order to make inference tractable. In the case of an HMM, the observation at time t is assumed to depend only on the state at time t, ensuring that each observation element is treated as an isolated unit, independent from all other elements in the sequence (Wallach, 2002).

In fact, most sequential data cannot be accurately represented as a set of isolated elements. Such data contain long-distance dependencies between observation elements and benefit from being represented in by a model that allows such dependencies and enables observation sequences to be represented by non-independent overlapping features. For example, in the POS task, when tagging a word, information such as the words surrounding the current word, the previous tag, whether the word begins with a capital character, can be used as complex features and help to improve the tagging performance.

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 $p(\boldsymbol{y}|\boldsymbol{x})$ 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 $p(\boldsymbol{x})$.

Maximum Entropy Markov Models (MEMMs)

MEMMs (McCallum, 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, instead of defining two types of distribution, a MEMM has only a single set of separately trained distributions of the form:

$$p(y'|x) = p(y'|y,x) \tag{8}$$

which represent the probability of moving from state y to y' on observation x. The fact the 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 9 show the graphic structure of MEMMs.

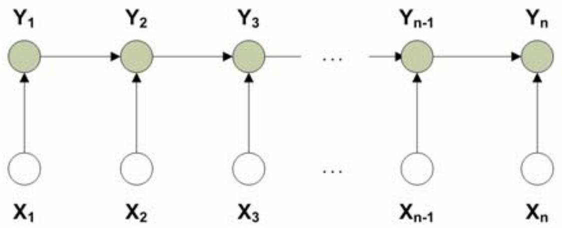


Figure 9. Graphic structure of first-order MEMMs

Given an observation sequence x, the conditional probability over label sequence y is given by

$$p(y|x) = p(y_1|x_1)\prod_{t=2}^{n} p(y_t|y_{t-1},x_{t-1}) \tag{9}$$

Treating observations as events to be conditioned upon rather than generated means that the probability of each transition may depend on non-independent, interacting features of the observation sequence. Making use of maximum entropy frame work and defining each state-observation transition function to be a log-linear model, equation (8) can be calculated as

$$p(y'|x) = \frac{1}{Z(y,x)} \exp(\sum_k \lambda_k f_k(y',x)) \tag{10}$$

where $Z(y, x)$ is a normalization factor; λ_k are parameters to be estimated and f_k are feature functions. The parameters can be estimated using Generalized Iterative Scaling (GIS) (McCallum, 2000). Each feature function can be represented as a binary feature. For example:

$$f(y',x) = \begin{cases} 1 & \text{if } b(x) \text{ is true and } y = y' \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Despite the differences between MEMMs and HMMs, there is still an efficient dynamic programming solution to the classic problem of identifying the most likely label sequence given an observation sequence. A variant Viterbi algorithm is given by (McCallum, 2000).

Label bias problem. Maximum Entropy Markov Models define a set of separately trained per-state probability distributions. This leads to an undesirable behavior in some situations, named *label bias problem* (Lafferty, 2001). Here we use an example to describe the label bias problem. The MEMM in Figure 10 is designed to shallow parse the sentences:

(1) The robot wheels Fred round.

(2) The robot wheels are round.

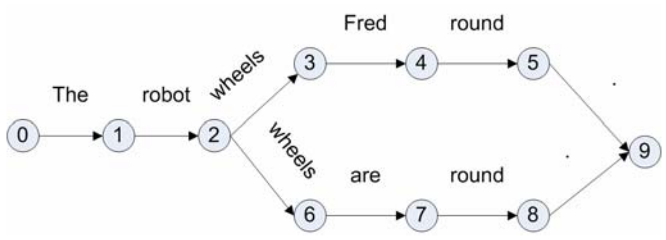


Figure 10. MEMM designed for shallow parsing

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 $p(3|2, \text{wheels}) = p(6|2, \text{wheels}) = 0.5$, 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 most 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.

Conditional Random Fields (CRFs)

CRFs are undirected graphical model trained to maximize a conditional probability. CRFs can be defined as follows:

CRF Definition. *Let $G = (V, E)$ be a graph such that $Y=(Y_v)_{v\in V}$, so that Y is indexed by the vertices of G. Then (X, Y) is a conditional random field in case, when conditioned on X, the random variable Y_v obey the Markov property with respect to the graph: $p(Y_v|X, Y_w, w\neq v) = p(Y_v|X, Y_w, w\backsim v)$, where $w\backsim v$ means that w and v are neighbors in G.*

A CRF is a random field globally conditioned on the observation X. Linear-chain CRFs were first introduced by Lafferty et al (2001). Figure 11 shows the graphic structure of the linear-chain CRFs.

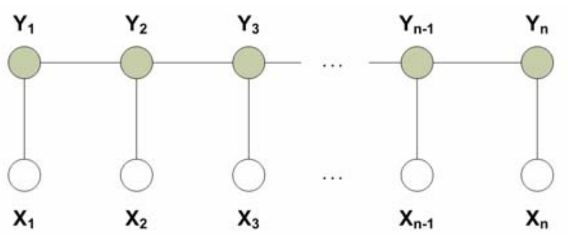


Figure 11. Graphic structure of linear-chain CRFs

By the fundamental theorem of random fields (Harmmersley, 1971), the conditional distribution of the labels y given the observations data x has the form

$$p_\lambda(y\mid x)=\frac{1}{Z_\lambda(x)}\exp(\sum_{t=1}^{T}\sum_k \lambda_k\cdot f_k(y_{t-1},y_t,x,t)) \tag{12}$$

where $Z_\lambda(x)$ is the normalization factor, also known as partition function, which has the form

$$Z_\lambda(x)=\sum_y \exp(\sum_{t=1}^{T}\sum_k \lambda_k\cdot f_k(y_{t-1},y_t,x,t)) \tag{13}$$

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

The most probable labeling sequence for an input $\boldsymbol{x}$

$$y^* = \arg\max_y p_\lambda(y\mid x) \tag{14}$$

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 $T=\{(x_k,y_k)\}_{k=1}^{N}$:

$$L_\lambda=\sum_{i=1}^{N}(\sum_{t=1}^{T}\sum_k \lambda_k\cdot f_k(y_{t-1},y_t,x_i,t)-\log Z_\lambda(x_i)) \tag{15}$$

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, 1972). In addition to the traditional methods, preconditioned conjugate-gradient (CG) (Shewchuk, 1994) or limited-memory quasi-Newton (L-BFGS) (Nocedal, 1999) have been found to perform better than the traditional methods (Sha, 2004). The voted perceptron algorithm (Collins, 2002) can also be utilized to train the models efficiently and effectively.

To avoid overfitting[1], log-likelihood is often penalized by some prior distribution over the parameters. Empirical distributions such as Gaussian prior, exponential prior, and hyperbolic-L_1 prior can be used, and empirical experiments suggest that Gaussian prior is a safer prior to use in practice (Chen, 1999).

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

Sequential labeling based extraction methods

By casting information extraction as sequential labeling, a set of labels need to be defined first according to the extraction task. For example, in metadata extraction from research papers (Peng, 2004), labels such as TITLE, AUTHOR, EMAIL, and ABSTRACT are defined. A document is viewed as an observation sequence x. The observation unit can be a word, a text line, or any other unit. Then the task is to find a label sequence y that maximize the conditional probability $p(y|x)$ using the models described above.

In generative models, there is no other features can be utilized except the observation itself. Due to the conditional nature, discriminative models provide the flexibility of incorporating non-independent, arbitrary features as input to improve the performance. For example, in the task of metadata extraction from research papers, with CRFs we can use as features not only text content, but also layout and external lexicon. Empirical experiments show that the ability to incorporate non-independent, arbitrary features can significantly improve the performance.

On the other hand, the ability to incorporate non-independent, arbitrary features of discriminative models may sometimes lead to too many features and some of the features are of little contributions to the model. A feature induction can be performed when training the model to obtain the features that are most useful for the model (McCallum, 2003).

Non-linear Conditional Random Fields

Conditional Random Fields (CRFs) are the state-of-the-art approaches in information extraction taking advantage of the dependencies to do better extraction, compared with HMMs (Ghahramani, 1997) and MEMMs (McCallum, 2000). However, the previous linear-chain CRFs only model the linear-dependencies in a sequence of information, and is not able to model the other kinds of dependencies (e.g.

non-linear dependencies) (Lafferty, 2001; Zhu, 2005). In this section, we will discuss several non-linear conditional random field models.

Condition random fields for relational learning

HMMs, MEMMs and linear-chain CRFs can only model dependencies between neighboring labels. But sometimes it is important to model certain kinds of long-range dependencies between entities. One important kind of dependency within information extraction occurs on repeated mentions of the same field. For example, when the same entity is mentioned more than once in a document, such as a person name *Robert Booth*, in many cases, all mentions have the same label, such as SEMINAR-SPEAKER. An IE system can take advantage of this fact by favoring labelings that treat repeated words identically, and by combining feature from all occurrences so that the extraction decision can be made based on global information. Furthermore, identifying all mentions of an entity can be useful in itself, because each mention might contain different useful information. The skip-chain CRF is proposed to address this (Sutton, 2005; Bunescu, 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

$$p_\lambda(y \mid x) = \frac{1}{Z(x)} \exp\Big(\sum_{t=1}^{T} \sum_{k} \lambda_k \cdot f_k(y_{t-1}, y_t, x, t) + \sum_{(u,v)\in C} \sum_{l} \lambda_l \cdot f_l(y_u, y_v, x, u, v)\Big) \qquad (16)$$

where Z(x) is the normalization factor, f_k is the feature function similar to that in equation (12) and f_l is the feature function of the skip edges. λ_k and λ_l are weights of the two kinds of feature functions.

Because the loops in a skip-chain CRF can be long and overlapping, exact inference is intractable for the data considered. 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, 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 place name when it appears alone, but when it occurs within the phrase *The China Daily,* it should be labeled as an organization (Finkel, 2005).

2D CRFs for web information extraction

Zhu et al (2005) propose 2D Conditional Random Fields (2D CRFs). 2D CRFs are also a particular case of CRFs. They are aimed at extracting object information from two-dimensionally laid-out web pages. The graphic structure of a 2D CRF is a 2D grid, and it's natural to model the 2D laid-out information. If viewing the state sequence on diagonal as a single state, a 2D CRF can be mapped to a linear-chain CRF, and thus the conditional distribution has the same form as a linear-chain CRF.

Dynamic CRFs

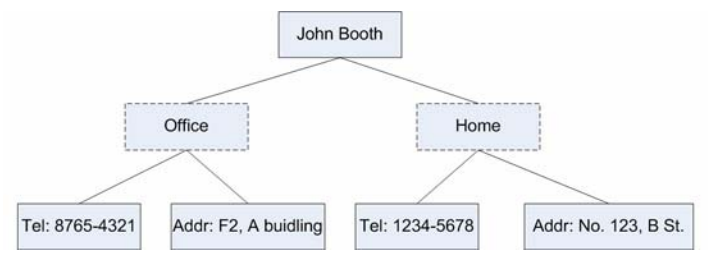
Sutton et al (2004) propose Dynamic Conditional Random Fields (DCRFs). As a particular case, a factorial CRF (FCRF) was used to jointly solve two NLP tasks (noun phrase chunking and Part-Of-Speech tagging) on the same observation sequence. Improved accuracy was obtained by modeling the dependencies between the two tasks.

Tree-structure CRFs for information extraction

We have investigated the problem of hierarchical information extraction and propose Tree-structured Conditional Random Fields (TCRFs). TCRFs can incorporate dependencies across the hierarchically laid-out information.

We here use an example to introduce the problem of hierarchical information extraction. Figure 12 (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 (ref. Figure 12 (b)). In this case, the existing 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 home address.

Contact Information:
John Booth
Office:
Tel: 8765-4321
Addr: F2, A building
Home:
Tel: 1234-5678
Addr: No. 123, B St.



(a) Example document (b) Organized the document in tree-structure

Figure 12. Example of tree-structured laid-out information

To better incorporate dependencies across hierarchically laid-out information, we propose a Tree-structured Conditional Random Field (TCRF) model. We present the graphical structure of the TCRF model as a tree and reformulate the conditional distribution by defining three kinds of edge features respectively representing the parent-child dependency, child-parent dependency, and sibling dependency. As the tree structure can be cyclable, exact inference in TCRFs is expensive. We propose to use the Tree-based Reparameterization (TRP) algorithm (Wainwright, 2001) to compute the approximate marginal probabilities for edges and vertices. We conducted experiments on company annual reports collected from Shang Stock Exchange. On the annual reports we defined ten extraction tasks. Experimental results indicate that the TCRFs can significantly outperform the existing linear-chain CRF model (+7.67% in terms of F1-measure) for hierarchical information extraction. See (Tang, 2006b) for details.

APPLICATIONS

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

Information Extraction 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 greatly 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 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 700,000 documents, primarily in the fields of computer and information science and engineering (Lawrence, 1999; Han, 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 extraction 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 and line-specific features are used. Each line is represented by a set of word 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.

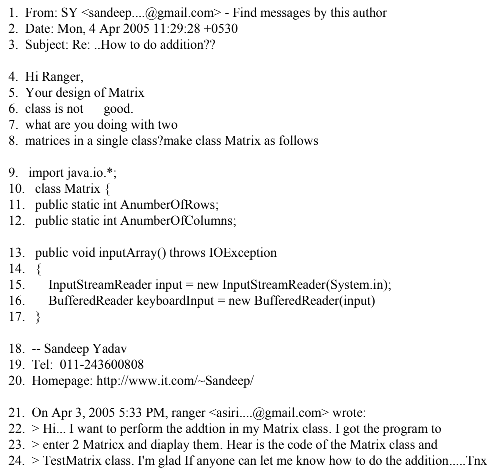
After classifying each line into one or more classes, meta-tag can be assigned to lines that have only one class label. For those that have more than one class label, a further identification is employed to extract metadata from each line. The task is cast as a chunk identification task. Punctuation 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 obtains an overall accuracy of 92.9%. It's adopted in the DL Citeseer and EbizSearch for automatic metadata extraction. It can be also generalized to other DL. See (Lawrence, 1999; Han, 2003) for details.

Information Extraction from Emails

We also make use of information extraction methods to email data (Tang, 2005a). Email is one of the commonest means for communication via text. It is estimated that an average computer user receives 40 to 50 emails per day (Ducheneaut, 2001). Many text mining applications need take emails as inputs, for example, email analysis, email routing, email filtering, information extraction from email, and newsgroup analysis.

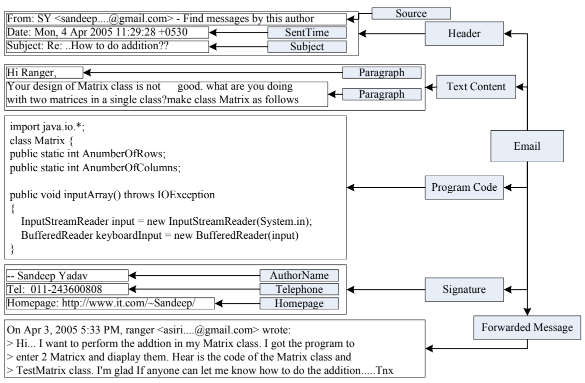
Unfortunately, information extraction from email has received little attention in the research community. Email data can contain different types of information. Specifically, it may contain headers, signatures, quotations, and text content. Furthermore, the text content may have program codes, lists, and paragraphs; the header may have metadata information such as sender, receiver, subject, etc.; and the signature may have metadata information such as author name, author's position, author's address, etc.

In this work, we formalize information extraction from email as that of text-block detection and block-metadata detection. Specifically, the problem is defined as a process of detection of different types of informative blocks (it includes header, signature, quotation, program code, list, and paragraph detections) and detection of block-metadata (it includes metadata detection of header and metadata detection of signature). We propose to conduct email extraction in a 'cascaded' fashion. In the approach, we perform the extraction on an email by running several passes of processing on it: first at email body level (text-block detection), next at text-content level (paragraph detection), and then at block levels (header-metadata detection and signature-metadata detection). We view the tasks as classification and propose a unified statistical learning approach to the tasks, based on SVMs (Support Vector Machines). Features used in the models have also been defined. See (Tang, 2005a) for details.



```
1.  From: SY <sandeep....@gmail.com> - Find messages by this author
2.  Date: Mon, 4 Apr 2005 11:29:28 +0530
3.  Subject: Re: ..How to do addition??

4.  Hi Ranger,
5.  Your design of Matrix
6.  class is not      good.
7.  what are you doing with two
8.  matrices in a single class?make class Matrix as follows

9.   import java.io.*;
10.   class Matrix {
11.   public static int AnumberOfRows;
12.   public static int AnumberOfColumns;

13.   public void inputArray() throws IOException
14.   {
15.      InputStreamReader input = new InputStreamReader(System.in);
16.      BufferedReader keyboardInput = new BufferedReader(input)
17.   }

18.  -- Sandeep Yadav
19.  Tel:  011-243600808
20.  Homepage: http://www.it.com/~Sandeep/

21.  On Apr 3, 2005 5:33 PM, ranger <asiri....@gmail.com> wrote:
22.  > Hi... I want to perform the addtion in my Matrix class. I got the program to
23.  > enter 2 Matricx and diaplay them. Hear is the code of the Matrix class and
24.  > TestMatrix class. I'm glad If anyone can let me know how to do the addition.....Tnx
```

Figure 13. Example of email message



```
From: SY <sandeep....@gmail.com> - Find messages by this author
Date: Mon, 4 Apr 2005 11:29:28 +0530
Subject: Re: ..How to do addition??

Hi Ranger,
Your design of Matrix class is not      good. what are you doing
with two matrices in a single class?make class Matrix as follows

import java.io.*;
class Matrix {
public static int AnumberOfRows;
public static int AnumberOfColumns;

public void inputArray() throws IOException
{
   InputStreamReader input = new InputStreamReader(System.in);
   BufferedReader keyboardInput = new BufferedReader(input)
}

-- Sandeep Yadav
Tel:  011-243600808
Homepage: http://www.it.com/~Sandeep/

On Apr 3, 2005 5:33 PM, ranger <asiri....@gmail.com> wrote:
> Hi... I want to perform the addtion in my Matrix class. I got the program to
> enter 2 Matricx and diaplay them. Hear is the code of the Matrix class and
> TestMatrix class. I'm glad If anyone can let me know how to do the addition.....Tnx
```

Figure 14. Annotation results of the email message

Figure 13 shows an example of email that includes many typical information. Lines from 1 to 3 are a header; lines from 18 to 20 are a signature; and a forwarded message lies from line 21 to line 24. Lines from 4 to 8 are the actual text content, which should be two paragraphs, but is mistakenly separated by extra line breaks. Moreover, the header has a sender (line 1), a sent time (line 2), and a subject (line 3); the signature has an author name (line 18), a telephone (line 19), and a homepage (line 20).

Figure 14 shows an ideal result of information extraction on the email in Figure 13. Within it, the text-blocks (the header, signature and the forwarded message) have been identified. The actual text content has been detected. In the text content, extra line breaks have been detected and the text has been annotated as two paragraphs. Metadata information is recognized in the identified header and the identified signature.

We propose a cascaded approach for information extraction from email and cast the extraction tasks as detection tasks of different types of information blocks. We employ a unified machine learning approach in the detection tasks.

The input is an email message. The implementation carries out extraction in the following steps. The identified text-blocks and other extraction results in each step will be saved for use in the later steps.

(1) Preprocessing. It uses patterns to recognize 'special words', including email address, IP address, URL, date, file directory, number (e.g. 5.42), money (e.g. $100), percentage (e.g. 92.86%), words containing special symbols (e.g. C#, .NET, .doc).

(2) Forwarded message detection. It identifies forwarded messages using hard-coded rules. It views lines starting with special characters (e.g. >, |, >>) as forwarded messages. It then eliminates the identified forwarded messages for later processing.

(3) Header and signature detection. It detects the header and signature (if there exist) in the email by using a classification model. It next eliminates the identified blocks (headers and signatures).

(4) Metadata detection in header and signature. It uses the identified headers and signatures as input and then detects the metadata information from the headers and signatures, respectively.

(5) List and program code detection. It detects list and program code (if there exist) in the email with the same approach as that in header and signature detection and removes them from the text content. After that, only natural language text remains.

(6) Paragraph annotation. It identifies whether or not each line break is a paragraph ending by using a classification model. If not, it removes the line break. As a result, the text is segmented into paragraphs. The step is based on paragraph ending detection.

We make use of Support Vector Machines (SVM) as the classification model (Vapnik, 1998). We use SVM-light, which is available at http://svmlight.joachims.org/. We obtain high performances in all detection tasks. (The F1-measuer scores range from 89.83% to 97.17%.)

The extracted information from email is applied to applications of email data cleaning (Tang, 2005a) and email classification.

In email data cleaning, we try to remove 'noisy' (irrelevant) blocks for a specific application (e.g. term extraction, a task in which base noun phrases are extracted from documents) and transform relevant text into a canonical form as that in a newspaper article. For term extraction, we identify and remove the header, signature, program code, and forwarded message. We view the remaining text as the relevant text. In the relevant text, we identify and remove extra line breaks, remove extra punctuations, and restore badly cased words. Experimental results show that the extraction based email cleaning can significantly improve the accuracy of term extraction. The improvements on precision range from +49.90% to +71.15%. See (Tang, 2005a) for details.

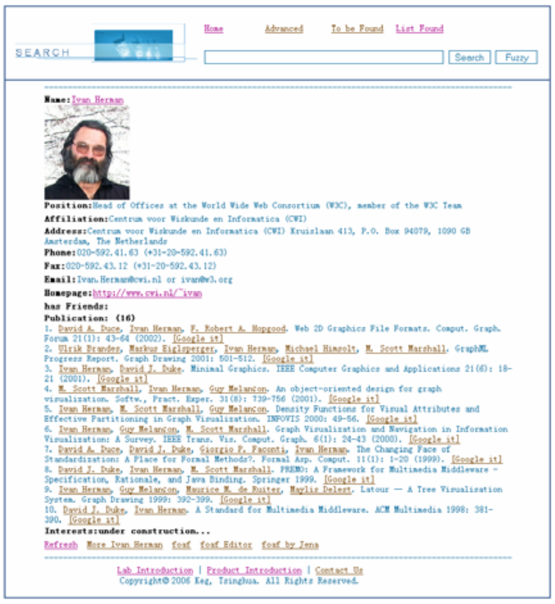
In email classification, we are aimed at taking advantage of the extracted information to improve the performance of email classification. We evaluated the classification results on Enron Email Dataset, which is available at http://www.cs.umass.edu/~ronb/enron_dataset.html. Experimental results show that the classification performance can be significantly improved (averagely +49.02% in terms of F1-measure) by making use of the extraction results from emails. The related issues are what we are currently researching, and will be reported elsewhere.

Person Profile Extraction

Person information management is an important topic in both research community and industrial community. A person can have different types of information: person profile (including portrait, homepage, position, affiliation, publications, and documents), contact information (including address, email, telephone, and fax number), and social network information (including person or professional relationships between persons, e.g. friend relationship). However, the information is usually hidden in heterogeneous and distributed web pages.

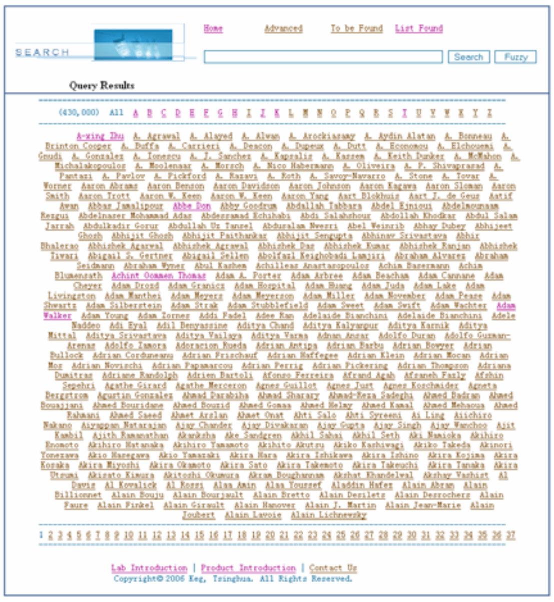
We have investigated the problem of person information extraction. We have found that the person information is mainly hidden in person homepage, person introduction page (web page that introduces the person), person list (e.g. a faculty list), and email message (e.g. in signature). We employed the classification based method to extract the person information from the different types of web pages.

More specifically, in extraction we convert a web page into a token sequence (the token can be word, punctuation, and space). Then we view each token as a candidate and define features for each candidate. Due to space limitation, we omit the details of the feature definition. After that, we use two classification models to respectively identify whether a token is the start position and whether the token is the end position for each type of information. We next view the tokens between the start token and the end token as the target. We can also use the text-line as candidate in extraction.



Name: Ivan Herman
Position: Head of Offices at the World Wide Web Consortium (W3C), member of the W3C Team
Affiliation: Centrum voor Wiskunde en Informatica (CWI)
Address: Centrum voor Wiskunde en Informatica (CWI) Kruislaan 413, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Phone: 020-592.41.63 (+31-20-592.41.63)
Fax: 020-592.43.12 (+31-20-592.43.12)
Email: Ivan.Herman@cwi.nl or ivan@w3.org
Homepage: http://www.cwi.nl/~ivan
has Friends:
Publication: (16)
1. David A. Duce, Ivan Herman, F. Robert A. Hopgood. Web 2D Graphics File Formats. Comput. Graph. Forum 21(1): 43-64 (2002). [Google it]
2. Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, M. Scott Marshall. GraphML Progress Report. Graph Drawing 2001: 501-512. [Google it]
3. Ivan Herman, David J. Duke. Minimal Graphics. IEEE Computer Graphics and Applications 21(6): 18-21 (2001). [Google it]
4. M. Scott Marshall, Ivan Herman, Guy Melançon. An object-oriented design for graph visualization. Softw., Pract. Exper. 31(8): 739-756 (2001). [Google it]
5. Ivan Herman, M. Scott Marshall, Guy Melançon. Density Functions for Visual Attributes and Effective Partitioning in Graph Visualization. INFOVIS 2000: 49-56. [Google it]
6. Ivan Herman, Guy Melançon, M. Scott Marshall. Graph Visualization and Navigation in Information Visualization: A Survey. IEEE Trans. Vis. Comput. Graph. 6(1): 24-43 (2000). [Google it]
7. David A. Duce, David J. Duke, Giorgio P. Faconti, Ivan Herman. The Changing Face of Standardization: A Place for Formal Methods?. Formal Asp. Comput. 11(1): 1-20 (1999). [Google it]
8. David J. Duke, Ivan Herman, M. Scott Marshall. PREMO: A Framework for Multimedia Middleware - Specification, Rationale, and Java Binding. Springer 1999. [Google it]
9. Ivan Herman, Guy Melançon, Maurice M. de Ruiter, Maylis Delest. Latour — A Tree Visualization System. Graph Drawing 1999: 392-399. [Google it]
10. David J. Duke, Ivan Herman. A Standard for Multimedia Middleware. ACM Multimedia 1998: 381-390. [Google it]
Interests: under construction...
Refresh More Ivan Herman foaf foaf Editor foaf by Jena

Lab Introduction | Product Introduction | Contact Us
Copyright© 2006 Keg, Tsinghua. All Rights Reserved.



Query Results

(430,000) All A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A-xing Zhu A. Agrawal A. Alayed A. Alwan A. Arockiasamy A. Aydin Alatan A. Bonneau A. Brinton Cooper A. Buffa A. Carrieri A. Deacon A. Dupeux A. Dutt A. Economou A. Elchouemi A. Gnudi A. Gonzalez A. Ionescu A. J. Sanchez A. Kapralis A. Kassem A. Keith Dunker A. McMahon A. Michalakopoulos A. Moolenaar A. Morsch A. Nico Habermann A. Oliveira A. P. Shivaprasad A. Pantazi A. Pavlov A. Pickford A. Razavi A. Roth A. Savoy-Navarro A. Stone A. Tovar A. Worner Aaron Abrams Aaron Benson Aaron Davidson Aaron Johnson Aaron Kagawa Aaron Sloman Aaron Smith Aaron Trott Aaron W. Keen Aaron W. Keen Aaron Yang Aart Blokhuis Aart J. de Geus Aatif Awan Abbas Jamalipour Abbe Don Abby Goodrum Abdallah Tabbara Abdel Ejnioui Abdelmounaam Rezgui Abdelnaser Mohammad Adas Abderramad Echihabi Abdi Salahshour Abdollah Khodkar Abdul Salam Jarrah Abdulkadir Gorur Abdullah Uz Tansel Abduralam Nwesri Abel Weinrib Abhay Dubey Abhijeet Ghosh Abhijit Ghosh Abhijit Paithankar Abhijit Sengupta Abhinav Srivastava Abhir Bhalerao Abhishek Agarwal Abhishek Agrawal Abhishek Das Abhishek Kumar Abhishek Ranjan Abhishek Tiwari Abigail S. Gertner Abigail Sellen Abolfazl Keighobadi Lamjiri Abraham Alvarez Abraham Seidmann Abraham Wyner Abul Kashem Achilleas Anastasopoulos Achim Basermann Achim Blumensath Achint Oommen Thomas Adam A. Porter Adam Arbree Adam Beacham Adam Cannane Adam Cheyer Adam Drozd Adam Granicz Adam Hospital Adam Huang Adam Juda Adam Lake Adam Livingston Adam Manthei Adam Meyers Adam Meyerson Adam Miller Adam November Adam Pease Adam Shwartz Adam Silberstein Adam Strak Adam Stubblefield Adam Sweet Adam Swift Adam Wachter Adam Walker Adam Young Adam Zorner Addi Fadel Adee Ran Adelaide Bianchini Adelaide Bianchini Adele Naddeo Adi Eyal Adil Benyassine Aditya Chand Aditya Kalyanpur Aditya Karnik Aditya Mittal Aditya Srivastava Aditya Vailaya Aditya Varma Adnan Ansar Adolfo Duran Adolfo Guzman-Arenas Adolfo Zamora Adoracion Rueda Adrian Antipa Adrian Barbu Adrian Bowyer Adrian Bullock Adrian Corduneanu Adrian Frischauf Adrian Haffegee Adrian Klein Adrian Mocan Adrian Mos Adrian Novischi Adrian Papamarcou Adrian Perrig Adrian Pickering Adrian Thompson Adriana Dumitras Adriane Randolph Adrien Bartoli Afonso Ferreira Afrand Agah Afsaneh Fazly Afshin Sepehri Agathe Girard Agathe Merceron Agnes Guillot Agnes Just Agnes Koschmider Agneta Bergstrom Agustin Gonzalez Ahmad Darabiha Ahmad Sharary Ahmad-Reza Sadeghi Ahmed Badran Ahmed Bouajjani Ahmed Bouridane Ahmed Bouzid Ahmed Gomaa Ahmed Helmy Ahmed Kamal Ahmed Mehaoua Ahmed Rahmani Ahmed Saeed Ahmet Arslan Ahmet Onat Ahti Salo Ahti Syreeni Ai Ling Aiichiro Nakano Aiyappan Natarajan Ajay Chander Ajay Divakaran Ajay Gupta Ajay Singh Ajay Wanchoo Ajit Kambil Ajith Ramanathan Akanksha Ake Sandgren Akhil Sahai Akhil Seth Aki Namioka Akihiro Enomoto Akihiro Hatanaka Akihiro Yamamoto Akihito Akutsu Akiko Kashiwagi Akiko Takeda Akinori Yonezawa Akio Hasegawa Akio Yamazaki Akira Hara Akira Ishikawa Akira Ishino Akira Kojima Akira Kosaka Akira Miyoshi Akira Okamoto Akira Sato Akira Takemoto Akira Takeuchi Akira Tanaka Akira Utsumi Akisato Kimura Akitoshi Okumura Akram Boughannam Akshat Khandelwal Akshay Vashist Al Davis Al Kovalick Al Rossi Alaa Amin Alaa Youssef Aladdin Hafez Alain Abran Alain Billionnet Alain Bouju Alain Bourjault Alain Bretto Alain Desilets Alain Desrochers Alain Faure Alain Finkel Alain Girault Alain Hanover Alain J. Martin Alain Jean-Marie Alain Joubert Alain Lavoie Alain Lichnewsky

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37

Lab Introduction | Product Introduction | Contact Us
Copyright© 2006 Keg, Tsinghua. All Rights Reserved.

(a) (b)

Figure 15. Personal Network Search system

For learning the classification models, we have human annotators conduct annotation on the web pages. We also convert the web page into a token sequence and view each token as the candidate. Features are defined for each candidate. Finally, we learn two classification models respectively for the start position identification and the end position identification for each type of information.

As models, we use SVMs (Support Vector Machines) (Vapnik, 1998). Features are defined in the SVM models respectively for each type of the information. The average F1-measure obtained in extraction is 91.18%.

We have developed a system based on the extracted person information, which is called 'Personal Network Search' (PNS shortly). In PNS, the user inputs a person name, and the system returns the information of the person. Given a person name, we first utilize Google API to get a list of relevant documents. Then a classification model is employed to identify whether or not a document in the list is really 'related' to the person. Next, we extract person information from the identified documents using the classification based method as described above.

Figure 15 shows the snapshots of the PNS system. In Figure 15 (a), the user types a person name, and he gets a detailed description of the person. Figure 15 (b) shows the list of gathered persons in our current system. See (Tang, 2006a) for details.

Table Extraction Using Conditional Random Fields

Tables — textual tokens laid out in tabular form — are often used to compactly communicate information in fields and records. They have been described as "databases designed for human eyes". Tables appear in the earliest writing on clay tablets, and in the most modern Web pages. Some make use of line-art, while others rely on white space only. They sometimes consist merely of two simple columns, other times of extremely baroque collections of headings, embedded subheadings, and varying cell sizes. They are used in everything from government reports, to magazine articles, to academic publications.

Pinto and McCallum (2003) propose a model of table extraction that richly integrates evidence from both content and layout by using Conditional Random Fields (CRFs). They describe a method that simultaneously locates tables in plain-text government statistical reports, and labels each of their constituent lines with tags such as header, sub-header, data, separator, etc. The features measure aspects of the input stream such as the percentage of alphabetic characters, the presence of regular expression matching months or years, and the degree to which white space in the current line aligns with white space in the previous line. In experiments on government reports, tables are located with 92% in terms of F1-measure, and lines are labeled with 94% accuracy — reducing error by 80% over a similarly configured hidden Markov model with the same features. See (Pinto, 2003) for details. See also (Wang, 2002).

Shallow Parsing with Conditional Random Fields

Shallow parsing identifies the non-recursive cores of various phrase types in text, possibly as a precursor to full parsing or information extraction (Abney, 1991). The paradigmatic shallow parsing problem is NP chunking, which finds the non-recursive cores of noun phrases called base NPs. The pioneering work of (Ramshaw, 1995) introduced NP chunking as a machine-learning problem, with standard datasets and evaluation metrics. The task was extended to additional phrase types for the CoNLL-2000 shared task (Tjong Kim Sang, 2000), which is now the standard evaluation task for shallow parsing.

Sha et al (2003) employ Conditional Random Fields (CRFs) into shallow parsing. They carried out an empirical study on different sequential labeling approaches in shallow parsing. Their experimental results show that CRFs outperform all reported single-model NP chunking results on the standard evaluation dataset. They also compared different kinds of parameter estimation methods for training CRF models that confirm and strengthen previous results on shallow parsing and training methods for maximum entropy models.

FUTURE RESEARCH DIRECTIONS

There are a variety of promising directions for future research in applying supervised machine learning to information extraction.

On the machine-learning side, it would be interesting to generalize the ideas of large-margin classification to sequence models, strengthening the results of (Collins, 2002) and leading to new optimal training algorithms with stronger guarantees against overfitting. For example, (Taskar, 2003) proposes a maximal Markov model for sequential labeling task using the maximal margin theory.

In information extraction, in addition to identifying entities, an important problem is extracting specific types of relations between entities. For example, in newspaper text, one can identify that an organization is located in a particular city or that a person is affiliated with a specific organization (Zelenko, 2003); in biomedical text, one can identify that a protein interacts with another protein or that a protein is located in a particular part of the cell (Bunescu, 2005a; Craven, 1999). The entities may occur in different parts of a sentence or paragraph. New principled methods are needed to such problems to identify both the entities while identify their relations. Bunescu and Mooney (2005b) propose to use a Statistical Relational Learning (SRL) method for the complex problem. They are trying to integrate decision at different levels (e.g. different kinds of entity identification and different kinds of relations identification) into the SRL model. Moreover, several recent projects have taken the first steps in this direction. For example, Sutton (2004) presents a dynamic version of CRF that integrates part-of-speech tagging and noun-phrase chunking into one coherent process. (Roth, 2004) presents an information extraction approach based on linear-programming that integrates recognition of entities with the identification of relations between these entities.

As another future work, more applications, especially practical applications, need to be investigated. The new applications can provide rich data sources for conducting information extraction, at the same time bring big challenges to the field. This is because various applications have various characteristics, needing to use different methods to deal with.

CONCLUSIONS

Aiming to apply methods and technologies from practical computer science such as compiler construction and artificial intelligence to the problem of processing unstructured textual data automatically, information extraction has become an important sub-discipline of language engineering, a branch of computer science. Nowadays, the significance of Information Extraction is determined by the growing amount of information available in unstructured (i.e. without metadata) form, for instance on the Internet.

In this chapter, we have reviewed the information extraction methods. Specifically, we focus on the three state-of-the-art methods: rule learning based method, classification based method, and sequential labeling base method. We have explained the principle of the three methods by using several developed systems as examples. We have also introduced our research work on the information methods and their applications. We also introduced several practical application of information extraction, ranging from natural language processing to information extraction from web pages and plain texts.

The rule learning based method try to exploit the regularity in language expressions of certain information to find common linguistic patterns that match these expressions. It is easy to understand by an average user. The method can obtain good performance when processing some semi-structured documents (e.g. template-based web page). Its disadvantage lies on that its rudimentary learning mechanisms cannot provide enough generalization capabilities. This makes it difficult to obtain good performance in complicated situations (e.g. extraction from natural language text).

The classification based method casts the IE task as a classification problem in terms of the statistical theory. It can incorporate different types of information (including words, syntax, a prior knowledge, etc.). Thus it has more generalization capabilities than the rule based method. In several real-world applications, it can outperform the rule based method. Its drawback is that its model is usually complex and it is difficult for the general user to understand (e.g. the feature definition). Thus the performances of extraction differ from application to application.

The sequential labeling based method can make use of dependencies between information to improve the extraction performance. It is also based on the statistical theory and thus has strong generalization capabilities. In many applications, in particular natural language processing, it can outperform the rule based method and the classification based method. As for the disadvantage, similar to the classification based method, it is not easy to be understood by a general user.

Information extraction suffers from uncertainty and implication of the natural language. Both of the two problems are difficult for machine to automatic extraction, sometimes even for human. For example, “It is likely that …”. In such sentence, it is difficult to determine the reliability degree of the information. Consider another example “After a furious fight, enemy raised the white flag”, here the white flag means a defeat. However, it would of course difficult for computer to conclude the implication.

Another interesting also important issue is how to make use of the prior knowledge in information extraction. So far, a usual method for incorporating the prior knowledge is to use some domain-specific dictionaries, thesauri 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.

In future work, research community has to face the rising challenges and focuses on how to enhance the practical usefulness of IE methods.

ACKNOWLEDGE

The work is funded by the Natural Science Foundation of China under Grant No. 90604025. Thanks to the anonymous reviewers for their constructive suggestions.

REFERENCES

Abney, S. (1991). Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny (Eds.), *Principle-based parsing*. Boston: Kluwer Academic Publishers.

Berger, A. L., Della Pietra, S. A., & Della Pietra, V. J. (1996). A maximum entropy approach to natural language processing. *In Computational Linguistics* (Vol.22, pp.39-71). MA: MIT Press.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *In* D. Haussler (Eds.) *5th Annual ACM Workshop on COLT* (pp.144-152). Pittsburgh, PA: ACM Press.

Bunescu, R., Ge, R., Kate, R. J., Marcotte, E. M., Mooney, R. J., Ramani, A. K., et al. (2005). Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine (special issue on Summarization and Information Extraction from Medical Documents).* 33(2), pp.139-155.

Bunescu, R. & Mooney, R. J. (2005). Statistical relational learning for natural language information extraction. In Getoor, L., & Taskar, B. (Eds.), *Statistical Relational Learning*, forthcoming book

Califf, M. E., & Mooney, R. J. (1998). Relational learning of pattern-match rules for information extraction. *In Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*. pp.6-11.

Califf, M. E., & Mooney, R. J. (2003). Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research.* Vol.4, pp.177-210.

Chen, S. F., & Rosenfeld, R. (1999). A Gaussian prior for smoothing maximum entropy models. *Technical Report CMU-CS-99-108*, Carnegie Mellon University.

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.

Collins, M. (2002). Discriminative training methods for Hidden Markov models: theory and experiments with Perceptron algorithms. *In Proceedings of the Conference on Empirical Methods in NLP (EMNLP'02).*

Craven, M., & Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. *In Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-1999).* pp.77-86. Heidelberg, Germany.

Darroch, J. N., & Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43 (5), pp.1470-1480.

Ducheneaut, N., & Bellotti, V. (2001). E-mail as Habitat: An exploration of embedded personal information management. *Interactions*, Vol.8, pp.30-38.

Finkel, J. R., Grenager, T., & Manning, C. D. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. *In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*. pp.363-370.

Finn, A., & Kushmerick, N. (2004). Information extraction by convergent boundary classification. *In AAAI-04 Workshop on Adaptive Text Extraction and Mining*. San Jose, USA.

Finn, A. (2006). A multi-level boundary classification approach to information extraction. Phd thesis, University College Dublin.

Freitag, D. (1998). Information extraction from HTML: Application of a general machine learning approach. *In Proceedings of the 15th Conference on Artificial Intelligence (AAAI'98).* pp.517-523

Freitag, D., & Kushmerick, N. (2000). Boosted wrapper induction. *In Proceedings of 17th National Conference on Artificial Intelligence*. pp.577-583

Ghahramani, Z., & Jordan, M. I. (1997). Factorial Hidden Markov Models. *Machine Learning*, Vol.29, pp.245-273

Hammersley, J., & Clifford, P. (1971). Markov fields on finite graphs and lattices. Unpublished manuscript.

Han, H., Giles, L., Manavoglu, E., Zha, H., Zhang, Z., & Fox, E.A. (2003). Automatic document metadata extraction using support vector machines. *In Proceedings of 2003 Joint Conference on Digital Libraries* (*JCDL'03*). pp.37-48

Kauchak, D., Smarr, J., & Elkan, C. (2004). Sources of success for boosted wrapper induction. *The Journal of Machine Learning Research*, Vol.5, pp.499-527. MA: MIT Press.

Kristjansson, T. T., Culotta, A., Viola, P. A., & McCallum, A. (2004). Interactive information extraction with constrained conditional random fields. *In Proceedings of AAAI'04*, pp.412-418

Kushmerick, N., Weld, D. S., & Doorenbos, R. (1997). Wrapper induction for information extraction. *In Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI'97)*. pp.729-737.

Kushmerick, N. (2000). Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, Vol.118, pp.15-68.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields:

Probabilistic models for segmenting and labeling sequence data. *In Proceedings of the 18th International Conference on Machine Learning (ICML'01)*. pp.282-289.

Lawrence, S., Giles, C.L., & Bollacker K. (1999). Digital libraries and autonomous citation indexing. *IEEE Computer*, Vol.32(6), pp.67-71.

Li, J., & Yu, Y. (2001). Learning to generate semantic annotation for domain specific sentences. *In Proceedings of the Knowledge Markup and Semantic Annotation Workshop in K-CAP'2001*, Victoria, BC.

Li, X., & Liu, B. (2003). Learning to classify texts using positive and unlabeled data. *In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI'2003)*. pp.587-592

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum Entropy Markov Models for information extraction and segmentation. *In Proceedings of the 17th International Conference on Machine Learning (ICML'00)*. pp.591-598.

McCallum, A. (2003). Efficiently inducing features of Conditional Random Fields. *In Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence.* pp.403-410.

Morik, K., Brockhausen, P., & Joachims, T. (1999). Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring. *In Proceedings of International Conference on Machine Learning (ICML*'99). pp.268-277.

Muslea, I., Minton, S., & Knoblock, C. (1998). STALKER: Learning extraction rules for semistructured, web-based information sources. *In AAAI Workshop on AI and Information Integration*. pp.74-81.

Muslea, I., Minton, S., & Knoblock, C. (1999). Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*. Vol.4, pp.93-114.

Muslea, I. (1999). Extraction patterns for information extraction tasks: A survey. *In Proceedings of AAAI-99: Workshop on Machine Learning for Information Extraction*. Orlando.

Muslea, I., Minton, S., & Knoblock, C. A. (2003). Active learning with strong and weak views: A case study on wrapper induction. *In Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI).* Acapulco, Mexico.

Ng, H. T., Lim, C. Y., Koo, J. L. T. (1999). Learning to Recognize Tables in Free Text. *In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics (ACL'99)*. pp. 443-450.

Nocedal, J., & Wright, S. J. (1999). Numerical optimization. New York, USA: Springer press.

Peng, F. (2001). Models for Information Extraction. Technique Report.

Peng, F., & McCallum, A. (2004). Accurate information extraction from research papers using Conditional Random Fields. *In Proceedings of HLT-NAACL*. pp. 329-336.

Pinto, D., McCallum, A., Wei, X., & Croft, W. B. (2003). Table Extraction Using Conditional Random Fields. *In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*.

pp. 235-242.
Ramshaw, L. A., & Marcus, M. P. (1995). Text chunking using transformation-based learning. *In Proceedings of Third Workshop on Very Large Corpora, ACL*. pp.67-73.
Ratnaparkhi, A. (1998). Unsupervised Statistical Models for Prepositional Phrase Attachment. *In Proceedings of COLING ACL'98*. pp.1079-1085. Montreal, Canada.
Riloff, E. (1993). Automatically Constructing a Dictionary for Information Extraction Tasks. *In Proceedings of the Eleventh National Conference on Artificial Intelligence*. pp.811-816.
Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text. *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*. pp.1044-1049.
Riloff, E., & Jones, R. (1999). Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. *In Proceedings of the Sixteenth National Conference on Artificial Intelligence*. pp.474-479.
Roth, D., & Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. *In Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)* . pp.1-8. Boston, MA.
Schölkopf B., Burges, C. J. C., & Smola A. J. (1999). *Advances in kernel methods: Support vector learning*. MA: MIT Press.
Sha, F., & Pereira, F. (2003). Shallow parsing with Conditional Random Fields. *In Proceedings of Human Language Technology, NAACL*. pp.188-191.
Shapire, R. E. (1999). A brief introduction to Boosting. *In Proceedings of the 16th International Joint Conference on Artificial Intelligence* (IJCAI-1999). pp.1401-1405.
Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. from http://www-2.cs.cmu.edu/.jrs/jrspapers.html#cg.
Siefkes, C., & Siniakov, P. (2005). An overview and classification of adaptive approaches to information extraction. *Journal on Data Semantics IV*. Berlin, Germany: Springer.
Soderland, S., Fisher, D., Aseltine, J., & Lehnert, W. (1995). CRYSTAL: Inducing a conceptual dictionary. *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI'95)*. pp.1314-1319.
Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. *Machine Learning*. Boston: Kluwer Academic Publishers
Sutton, C., Rohanimanesh, K., & McCallum, A. (2004). Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. *In Proceedings of ICML'2004*. pp.783-790.
Sutton, C., & McCallum, A. (2005). An introduction to Conditional Random Fields for relational learning. In Getoor, L., & Taskar, B. (Eds.), *Statistical Relational Learning*, forthcoming book.
Tang, J., Li, H., Cao, Y., & Tang, Z. (2005). Email Data Cleaning. *In Proceedings of SIGKDD'2005*. pp.489-499. Chicago, Illinois, USA.
Tang, J., Li, J., Lu, H., Liang, B., & Wang, K. (2005). iASA: Learning to Annotate the Semantic Web. *Journal on Data Semantic IV* (pp. 110-145). New York, USA: Springer Press.

Tang. J., Hong, M., Zhang, J., Liang, B., and Li, J. (2006). A New Approach to Personal Network Search based on Information Extraction. *In Proceedings of the first International Conference of Asian Semantic Web (ASWC)*. To appear.
Tang, J., Hong, M., Li, J., & Liang, B. (2006). Tree-structured conditional random fields for semantic annotation. *In Proceedings of 5th International Conference of Semantic Web* (*ISWC'2006*), pp.640-653.
Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *In Neural Information Processing Systems 2003*.
Tetko, I.V., Livingstone, D.J., & Luik, A.I. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. *Journal of Chemical Information and Computer Sciences*, Vol.35, pp.826-833.
Tjong Kim Sang, E. F., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. *In Proceedings of CoNLL-2000*, pp.127-132.
Vapnik, V. (1998). Statistical Learning Theroy. Springer Verlage, New York, 1998
Vapnik V. (1999). The Nature of Statistical Learning Theory. Springer Verlag, New York, 1999.
Wallach, H. (2002). Efficient training of Conditional Random Fields. Master thesis. University of Edinburgh, USA.
Wang, Y., & Hu, J. (2002). A Machine Learning based Approach for Table Detection on the Web. *In Proceedings of the 11th International World Wide Web Conference (WWW'02)*. pp. 242-250. Honolulu, Hawaii, USA.
Wainwright, M., Jaakkola, T., & Willsky, A. (2001). Tree-based reparameterization for approximate estimation on graphs with cycles. *In Proceedings of Advances in Neural Information Processing Systems (NIPS'2001)*. pp.1001-1008.
Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*, Vol.3, 1083-1106.
Zhang, L., Pan, Y., & Zhang, T. (2004). Recognizing and Using Named Entities: Focused Named Entity Recognition Using Machine Learning. *In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'04)*. pp.281-288.
Zhu, J., Nie, Z., Wen, J., Zhang, B., & Ma, W. (2005). 2D Conditional Random Fields for Web information extraction. *In Proceedings of 22nd International Conference on Machine Learning (ICML2005)*. pp.1044-1051. Bonn, Germany.

ADDITIONAL READING

Adwait, R. (1996). Maximum Entropy Model for POS tagging. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pp.133-142. Somerset, New Jersey, 1996.
Ahn, D. (2006). The Stages of Event Extraction. *In Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. pp. 1–8. Sydney, July 2006.
Allen, J. (1994). Natural Language Understanding (2nd Edition). Addison Wesley. 1994
Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden Markov Support Vector

Machines. *In Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*.

Appelt, D. & Israel, D. (1999). Introduction to Information Extraction Technology. *In Proceedings of IJCAI'99 Tutorial.*

Baum, L. E. & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Annual of Mathematical statistics*, 37:1554-1563, 1966.

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. *In Proceedings of the Sixth Workshop on Very Large Corpora New Brunswick*, New Jersey.

Bunescu, R.C. & Mooney, R.J. (2004). Collective Information Extraction with Relational Markov Networks. *In Proceedings of ACL'2004*.

Cafarella, M.J., Downey, D., Soderland, S., & Etzioni, O. (2005). KnowItNow: Fast, Scalable Information Extraction from the Web. *In Proceedings of HLT/EMNLP'2005.*

Chieu, H.L. (2002). A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. *In Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI'2002)*. pp.786-791.

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP'2002)*. pp.1-8, July 06, 2002.

Dietterich, T. (2002). Machine Learning for Sequential Data: A Review. *In Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. pp. 15–30. 2002. Springer-Verlag.

Downey, D., Etzioni, O., & Soderland, S. (2005). A Probabilistic Model of Redundancy in Information Extraction. *In Proceedings of 22th International Joint Conference on Artificial Intelligence (IJCAI'2005)*. pp. 1034-1041.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press, 1998.

Eikvil, L. (1999). Information Extraction from World Wide Web - A Survey. Rapport Nr. 945, July, 1999.

Embley, D.W. (2004). Toward Semantic Understanding - An Approach Based on Information Extraction. *In Proceedings of the Fifteenth Australasian Database Conference*, 2004.

Freitag, D. (1998). Machine Learning for Information Extraction from Online Documents. PhD thesis, School of Computer Science. Carnegie Mellon University.

Freitag, D. & McCallum, A. (2000). Information Extraction with HMM Structures Learned by Stochastic Optimization. *In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'2000)*.

Grishman, R. & Sundheim, B. (1996). Message Understanding Conference –6: A Brief History. *In Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, June 1996.

Hu, Y., Li, H., Cao, Y., Meyerzon, D., Teng, L., & Zheng, Q. (2006). Automatic Extraction of Titles from General Documents using Machine Learning. *Information*

Processing and Management. pp.1276-1293, 2006

Huffman, S.B. (1995). Learning Information Extraction Patterns from Examples. *In Proceedings of Learning for Natural Language Processing'1995*. pp. 246-260.

Jackson, P. & Moulinier, I. (2002). Natural Language Processing for Online Applications. John Benjamins, 2002.

Klein, D. & Manning, C. (2002). Conditional Structure Versus Conditional Estimation in NLP Models. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'2002)*, Philadelphia.

Laender, A.H.F., Ribeiro-Neto, B.A., da Silva, A.S., & Teixeira, J.S. (2002). A Brief Survey of Web Data Extraction Tools . *Journal of ACM SIGMOD Record*, 2002.

Leek, T.B. (1997). Information Extraction Using Hidden Markov Models. M.S. thesis.

Moens, M. (2006). Information Extraction: Algorithms and Prospects in a Retrieval Context. Springer press

Li, Y., Bontcheva, K., & Cunningham, H. (2005). Using Uneven-Margins SVM and Perceptron for Information Extraction. *In Proceedings of Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*. pp.72-79

Manning, C., & Schutze, H. (1999). Markov Models. In Book: Foundations of Statistical Natural Language Processing. The MIT Press. 1999.

Pazienza, M.T. (1999). Information Extraction : Towards Scalable, Adaptable Systems. Springer press.

Punyakanok, V. & Roth, D. (2001). The Use of Classifiers in Sequential Inference. In Proceedings of NIPS'01. pp.995-1001.

Rabiner, L. A. (1989). Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *In Proceedings of the IEEE'1989.*

Shawe-Taylor, J. & Cristianini, N. (2000). Introduction to Support Vector Machines. Cambridge University Press, 2000

Skounakis, M., Craven, M., & Ray, S. (2003). Hierarchical Hidden Markov Models for Information Extraction. *In Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico. Morgan Kaufmann,2003*.

Z. Zhang. (2004). Weakly-Supervised Relation Classification for Information Extraction. *In Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (CIKM'2004).*pp581-588.

---

[1] In machine learning, usually a learning algorithm is trained using some set of training examples. The learner is assumed to reach a state where it will also be able to predict the correct output for other examples. However, especially in cases where learning was performed too long or where training examples are rare, the learner may adjust to very specific random features of the training data, that have no causal relation to the target function. In this process of overfitting, the performance on the training examples still increases while the performance on unseen data becomes worse (Tetko, 1995).