3 Information Extraction Techniques

3.1 General Techniques Issues

In this section we will discuss various techniques in processing of Information Extraction. Currently, researchers try to use almost all artificial intelligent methods and machine learning algorithms to achieve high performance and automatic Information Extraction from documents. Under all used techniques, the most basic techniques are syntactic rules and basic Nature Language Processing (NLP) techniques. With the first technique some syntactic rules and patterns at the word level (such as regular expressions, token-based rules etc.) are used to extract fine information from text. Another widely used technique is based on Natural Language Processing (NLP). NLP was initially used for machine translation and speech recognition (Galliers 1993). The basis idea of using NLP in IE is analysing grammatical structure at sentence level and then constructing grammatical rules for some useful information within sentence (Cunningham 1997).

Other advanced technologies such as bayesian model, Hidden Markov Model (HMM), Decision Tree etc. are based on basic technologies mentioned above. Furthermore, other machine learning researchers find that machine-learning algorithms can be used to extract information by given examples automatically. Evaluations showed that some well-known machine learning algorithms (such as rule induction, statistical approaches, spatial model analyse, etc.) gain successful results in some defined domains (see also Section 3.3). There are also many experiments carried out in order to use various artificial intelligent methods (such as Case-based reasoning, neural network etc.) in IE. Instead of common techniques mentioned above such algorithms are selected for some extreme cases (such as documents with high noises levels etc.).

Also Information Retrieval methods are used in this field. Although we mentioned that IE and IR have different tasks and goal, some IR methods (such as Keywords with relevance) could be used for IE. For instance, such IR methods can be used to get short description with some significant keywords. Furthermore, information retrieval methods are also very suitable for classification of documents.

Besides traditional extraction techniques for normal documents, some special extraction techniques are developed for online HTML documents. IE Systems, which are developed for extracting information from HTML, are called generally HTML-Wrapper (Freitag 1998). Unlike traditional techniques appied in normal documents, wrappers use additional formatting information (e.g.

tags) in HTML documents. Wrapper rules can be written manually or generated automatically by machine learning algorithms.

It is difficult to evaluate and compare which methods or techniques are better than the other. One reason why various techniques are used in field IE is that IE is actually a real hard task. A single technique is only suitable for a few defined specialised problems in IE. In general, there is no common solution for the total problem fields of IE. However, It is worth to make a rough classification to identify which advantages and disadvantages each technique does have, as a prelude to proposing a hybrid solution to the general IE problems.

In order to understand the discussion, which follows about the progress and problems of information extraction, we need to briefly describe the MUC evaluation process as benchmark. In a MUC evaluation, participants are initially given a detailed description of the scenario (the information to be extracted), along with a set of documents and the templates to be extracted from these documents (the "training corpus"). Systems developers then get some time (1 to 6 months) to adapt their system to the new scenario. After this time, each participant gets a new set of documents (the "test corpus"), uses their system to extract information from these documents, and returns the extracted templates to the conference organiser. Meanwhile, the organiser has manually filled a set of templates (the "answer key") from the test corpus. Each system is assigned a variety of scores by comparing the system response to the answer key. The primary scores are ***precision*** and ***recall***. Let N_{key} be the total number of filled slots in the answer key, $N_{response}$ be the total number of filled slots in the system response, and $N_{correct}$ be the number of correctly filled slots in the system response (i.e., the number which match the answer key). Then

$$\textbf{precision} = N_{correct} / N_{response}$$

$$\textbf{recall} = N_{correct} / N_{key}$$

Sometimes an "***F score***" (*F-Measure*) is also used as a combined recall-precision score (Grishman 1999):

$$\textbf{F-Measure} = (2 * \text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

It must be mentioned that precision and recall are orthogonal measures and F-Measure can not represent the real property of an IE system.

Some other scores are also used for different evaluation goals in IE. In order to make an easy classification; it should be useful to use some criteria to scale various techniques. In following we will introduce some useful criteria used in this section.

- IE tasks
  As we know there is no technique as universal solution for all IE problems. Each technique focuses on one or some defined IE tasks. Using this criterion we can classify which technique is suitable for which situation.

- Document types
  Different document type (structured, semi-structured and free text) has different structure properties. Some techniques are specially developed for one documents type (such as spatial model for template filling, which is designed for structured and semi-structured documents, see Section 9), while other techniques (such as NLP, syntactic rules etc.) can be used in different document types as basic techniques. Likewise, difference between normal documents and online documents have to be respected. Techniques can be developed only for HTML (such as wrapper), or for both document types (such as syntactic rules).

- Result Performance
  As mentioned above, the most important performance scores are precision and recall. However, precision and recall are not the absolute benchmarks. We can not conclude that algorithm with higher precision and recall is definitively better than the algorithm with lower scores. There are many boundary conditions that have to be taken in account. In one full automatic IE system, the performance with slight error is already critical, while in a system allowing manual post-processing the tolerance of error could be loose. Moreover, for various IE tasks the difficulties to reach high performance are totally different. For instance, to extract a distance value (such as *100km*) is easy, but to extract a distance value between beach and hotel becomes much more difficult. Hence, it is necessary to mention the boundary conditions if concrete value of performance of a technique or algorithm is introduced. Other performance scores are e.g. accuracy, coverage and error rate. The definitions of such scores are as similar as those of precision and recall.

- Trainability
  Hand coding rules and patterns are time consuming and error prone. Learning algorithm can help the user to generate patterns and rule automatically. Moreover, some sophisticated machine learning algorithms can process a deep analyse only from the given example and construct model or rules (such as HMM or C4.5), which can not be created manually. Another advantage of trainability is flexible. Using given examples techniques can be easily adjusted to a new document domain.

- Processing Speed
  IE system can be either runtime tools or back office tools. That is, if IE system is used as run time tool, real time processing speed becomes a important requirement of practical IE. Some Machine Learning algorithm uses deep analyse and global optimisation, which is computational intensive and also time consuming. The result of such algorithms could be satisfied. But they become practical not usable if the user has to wait for result more than several hours, even days. On the other hand, some techniques perform quick processing but bad result performance. A balance point between result performance (Precision and Recall) and processing time performance must be investigated.

- Limitation
  Each technique has his own advantages but also limitations. Defined application domain, types of documents, size of input documents, required manual pre-processing etc. are typical limitations.

3.2 Basic Technologies

This section describes basic techniques used in information extraction. Basic technique means, that they are used widely and set as condition for other advanced methods. Generally, the basic technologies used in IE are syntactic rules and Natural Language Processing (NLP). In this section, these two main basic techniques are introduced. Some typical examples using these techniques are also given briefly.

Syntactic rules are the processes of using common regular expressions or other similar rules to form the lowest level of extraction, effectively constructing a bottom-up parsing of the text. Syntactic rules are also used on a larger scale when processing tokens that have been assigned a syntactic value.

Many systems go into building up a lexical based approach (NLP) to fact extraction. With NLP, at the lowest level, the text is broken into tokens (as in programming language parsing). From there sentences can be identified. Within sentences we can look to determine likely context of words and phrases, using various dictionaries and domain specific lexicons. By this stage we should have recognised tokens that are proper names or other useful information.

3.2.1 Syntactic Rules

Syntactic rules describe string properties in the lowest syntactic level. The most popular syntactic rule is regular expression. Information, which conforms to the rules, is extracted by pattern matching. No grammatical or semantic analyse are used for syntactic rules. Such rules are suitable for documents that consist of a mixture of grammatical, telegraphic, and/or ungrammatical text. Performing IE tasks on e.g. corpora of job postings, bus schedules, or apartment rentals have immediate practical applications. In general, syntactic rules can be interpreted in Finite State Automata (FSA) and can be generated automatically from given examples. In order to use syntactic rules to extract information exactly, some types of extraction rules presented in this section combine syntactic constraints with delimiters that “bound” the text to be extracted.

- **Regular Expressions**

A regular expression is a pattern that describes a set of strings. Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions. For example, such rule “[a,p]m [0-9]+:[0-9]+” extract time description such as “AM 12:45” from documents. Regular expressions are widely used in Unix world as searching and replacing instruction. Using regular expressions for Information Extracting is similar as using regular expression for searching in Unix. Regular expression is suitable for such contents with significant syntactic properties (such as number, date etc.). In general regular expressions can be used for all document types as basic technique. Compared with other sophisticated methods, the processing of regular expressions is very quickly, because the only input of extracting rule is the defined regular expressions. No background knowledge or lexicon is required. Because regular expressions are based on Finite State Automata (FSA), learning and automatically generating regular expressions are theoretical possible.

Although regular expressions locate fine information exactly, the contexts around the underlying fine information, which can help to locate information exactly, are not respected by using regular expressions alone. For instance, a regular expression “[0-9]+” defines all digit number sequence, but this rule is not able find out currency *100$* (which with suffix “$”) exactly (assumed only the number has to be extracted but not the suffix). This shortcoming of regular expression causes sometimes very bad precision and recall when only standard regular expressions are used. Consequent of this shortcoming of regular expressions is that regular expressions have to be combined with other constraints, in order to ensure high performance IE. Hence, regular expressions are normally considered as basic technique and must be used with respect of context, in order to get a high performance extracting.

- **WHISK Rule**

WHISK (Soderland 1998) is a learning system that generates extraction rules for a wide variety of documents ranging from rigidly formatted to free text. The WHISK extraction patterns are a special type of regular expressions that have two components: one that describes the context that makes a phrase relevant, and one that specifies the exact delimiters of the phrase to be extracted. Depending of the structure of the text, WHISK generates patterns that rely on either of the components (i.e., context-based patterns for free text, and delimiter-based patterns for structured text) or on both of them (i.e., for documents that lay in between structured and free text). In Figure 1 we show a sample WHISK extraction task from online texts. The sample document is taken from an apartment rental domain that consists of ungrammatical constructs, which, without being rigidly formatted, obey some structuring rules that make them human understandable. The sample pattern in Figure 1 has the following meaning: ignore all the characters in the text until you find a digit followed by the “br” string; extract that digit and fill the first extraction slot with it (i.e., “Bedrooms”). Then ignore again all the remaining characters until you rich a dollar sign immediately followed by a number. Extract the number and fill the “Price” slot with it.

DOCUMENT:	EXTRACTEDDATA:
Capitol Hill- 1 br twnhme.	<Bedrooms: 1
D/W W/D. Pkg incl $675.	Price: 675>
3BR upper flr no gar. $995.	<Bedrooms: 3
(206) 999-9999 	Price: 995>

Extraction rule:	* (<Digit>) 'BR' * '$' (<Nmb>)
Output:	Rental {Bedrooms @1} {Price @2}

Figure 1: A WHISK extraction task.

A more sophisticated version of the pattern could replace the “br” string by the semantic class “Bedroom”, which is defined as

Bedroom ::= (br; brs; bdrm; bedrooms; bedroom)

That is, the semantic class “Bedroom” is a placeholder for any of the abbreviations above. Actually, semantic class is a type of background knowledge. Such background knowledge is domain dependence and used as standard input. Normally, background knowledge is generated manually or can be provided from domain specified thesaurus or lexicon.

WHISK can also be used on free text domains, where it identifies the exact phrase of interest and allows semantic class constraints on all sentence elements. For grammatical text, the WHISK regular expressions provide a set of special constructs that are helpful when dealing with information provided by syntactic analysers (e.g., the clause structure of a sentence). For instance, the pattern * (PObj) * @Passive *F 'bomb' * {'by' *F (Person)} extracts the target and the name of the terrorist involved in a bombing action. The "*F" construct is similar to "*" but it skips characters only within the current syntactic field, while the "@Passive" requires the verb "bomb" to be in its passive form.

- **Remarks**

Because the syntactic rules define strict sequences of string, there is sometimes conflict between precision and recall. If rules are detailed, we get then good precision but poor recall. Conversely, if rules are crude, the recall will be improved, but the precision goes down. A compensation of this shortcoming is to use context information of the underlying contents.

Furthermore, WHISKY's rules describe not only the extracting rules for underlying contents, but also the construction of the template. That is, the name extracting and template filling is processed at one time. Such rules can only describe simple templates containing a few contents. However, in the practical domain template is normally much more sophisticated than the simple rental template. A template can have more than fifty contents. Each can have different cardinality again. In this case simple rules such as WHISK are not able to describe the template. A more reliable approach is to describe extracting rules for underlying contents (NE-rules) and the construction of template (TE-rules) separately (see also Section 5.1).

3.2.2 Natural Language Processing

Natural language processing (NLP) techniques rely on syntactic and semantic knowledge that is often manually encoded for a particular domain. Initially NLP is used for machine translation, speech recognition and also knowledge representation. IE researches use NLP techniques to pre-process documents and to extract underlying information. Basically, the methods presented in this section rely on the identification of terms in the documents, and uses NLP techniques such as automated Part-of-Speech Tagging to pre-process the textual data and Term Extraction to get the useful information. NLP techniques can be considered as an automated generalised indexing procedure that extracts from the full textual content of the document linguistically significant structures.

In this section we introduce at first basic NLP techniques used in IE. The most important NLP techniques (POS tagging and term extraction) are described in detail. Then we will review some NLP-based IE systems.

3.2.2.1 Part-Of-Speech (POS) tagging

The objective of the Part-Of-Speech tagging (POS-Tagging) is to automatically assign Part-of-Speech tags (i.e. morpho-syntactic categories such as noun, verb, adjective...) to words in context. For instance, a sentence as “a computational process executes programs” should be tagged as “a/DET computational/ADJ process/N executes/V programs/N”. The main difficulty of such a task is the lexical ambiguities that exist in all natural languages. For instance, in the previous sentence, both words ‘process’ and ‘programs’ could be either nouns(N) or verbs(V).

POS-tagging is initially used for machine speech translation and speech recognition. With POS-tagging technique, IE system tags at first sentences in different morpho-syntactic categories. An extracting rule using POS-tagging is normally a definition of combination of some morpho-syntactic categories with defined order. The following extracting is actually pattern matching processing to such POS-tagging rules (example see Section 3.2.2.2).

Several techniques have been designed for POS-tagging, two most important techniques are:

- Hidden Markov Model based approaches (Cutting et al. 1992);
- Rule-based approaches (Brill 1992);

Figure 2 attempts to depict the various approaches to automatic POS tagging. In reality, the picture is much more complicated, since many tagging systems use aspects of some or all of these approaches.

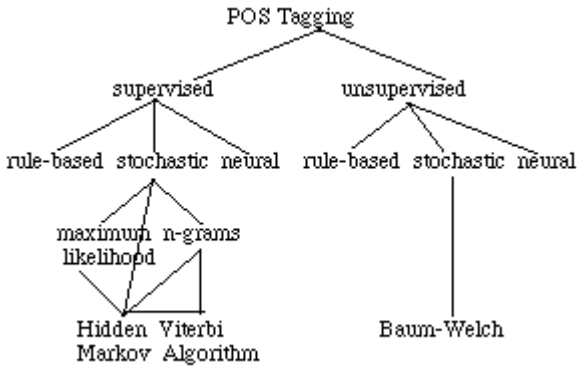

Figure 2: The various approaches to automatic POS tagging (Guilder 1995)

- **Supervised vs. Unsupervised**

One of the first distinctions, which can be made among POS taggers, is in terms of the degree of automation of the training and tagging process. The terms commonly applied to this distinction are *supervised* vs. *Unsupervised*. Supervised taggers typically rely on pre-tagged corpora. Such pre-tagged corpora are actually training examples and serve as the basis for creating any tools to be used throughout the tagging process, for example: the tagger dictionary, the word/tag frequencies, the tag sequence probabilities and/or the rule set. Unsupervised models, on the other hand, are those which do not require a pre-tagged corpus but instead use sophisticated computational methods to automatically induce word groupings (i.e. tag sets) and based on those automatic groupings, to either calculate the probabilistic information needed by stochastic taggers or to induce the context rules needed by rule-based systems. Each of these approaches has pros and cons.

The primary argument for using a fully automated approach to POS tagging is that it is extremely portable. It is known that automatic POS taggers tend to perform best when both trained and tested on the same genre of text. Supervised tagging is especially suitable for such domain dependence situation. Given some examples, supervised tagging has the best result of POS. However, creating example set is a manual job and error prone. Unsupervised tagging, on the other hand, is domain independence and can be used without any pre-tagged examples. The result of unsupervised method is normally not as good as supervised methods. For a more thorough discussion of the supervised vs. unsupervised training see (Brill & Marcus 1993, Schuetze 1993).

The following table outlines the differences between these two approaches.

SUPERVISED	UNSUPERVISED
selection of tagset/tagged corpus	induction of tagset using untagged training data
creation of dictionaries using tagged corpus	induction of dictionary using training data
calculation of disambiguation tools. May include:	induction of disambiguation tools. May include:
word frequencies	word frequencies
affix frequencies	affix frequencies
tag sequence probabilities	tag sequence probabilities
“formulaic” expressions	
tagging of test data using dictionary information	tagging of test data using induced dictionaries
disambiguation using statistical, hybrid or rule based approaches	disambiguation using statistical, hybrid or rule based approaches
calculation of tagger accuracy	calculation of tagger accuracy

Table 2 Two approaches of Part-Of-Speech (POS) Tagging (Guilder 1995)

- **Rule Based Tagging**

Typical rule based approaches use contextual information to assign tags to unknown or ambiguous words. These rules are often known as *context frame rules*. As an example, a context frame rule might say something like: If an ambiguous/unknown word X is preceded by a determiner and followed by a noun, tag it as an adjective.

In addition to contextual information, many taggers use morphological information to aid in the disambiguation process. One such rule might be: *if an ambiguous/unknown word ends in an -ing and is preceded by a verb, label it a verb* (depending on your theory of grammar, of course).

Some languages have significant properties in grammatical. Some systems use then contextual and morphological information by including rules pertaining to such factors as capitalisation and punctuation. In German for example, information about capitalisation proves extremely useful in the tagging of unknown nouns.

Rule based taggers most commonly require supervised training; but, very recently there has been a great deal of interest in automatic induction of rules. One approach to automatic rule induction is to run an untagged text through a tagger and see how it performs. A human then goes through the output of this first phase and corrects any erroneously tagged words The properly tagged text is then submitted to the tagger, which learns correction rules by comparing the two sets of data. Several iterations of this process are sometimes necessary (Guilder 1995).

For further information on rule-based taggers see (Brill 1995, Greene & Rubun 1971, Tapanainen 1994).

- **Stochastic Tagging**

Stochastic tagging means any model, which somehow incorporates frequency or probability, i.e. statistics, may be properly labelled stochastic. The term “stochastic tagger” can refer to any number of different approaches to the problem of POS tagging.

The simplest stochastic taggers disambiguate words based solely on the probability that a word occurs with a particular tag. In other words, the tag encountered most frequently in the training set is the one assigned to an ambiguous instance of that word. The problem with this approach is that while it may yield a valid tag for a given word, and can also yield inadmissible sequences of tags.

An alternative to the word frequency approach is to calculate the probability of a given sequence of tags occurring. This is sometimes referred to as the *n-gram* approach, referring to the fact that the best tag for a given word is determined by the probability that it occurs with the n previous tags. The most common algorithm for implementing an n-gram approach is known as the *Viterbi Algorithm*, a search algorithm which avoids the polynomial expansion of a breadth first search by "trimming" the search tree at each level using the best N *Maximum Likelihood Estimates* (where n represents the number of tags of the following word) (Rabiner 1989).

Combination of the previous two approaches, using both tag sequence probabilities and word frequency measurements, is known as a Hidden Markov Model. The assumptions underlying this model are the following:

Each hidden tag state produces a word in the sentence. Each word is:

- Uncorrelated with all the other words and their tags
- Probabilistic depending on the N previous tags only

(Dermatas & Kokkinakis 1995)

Hidden Markov Model taggers and visible Markov Model taggers may be implemented using the *Viterbi algorithm*, and are among the most efficient of the tagging methods discussed here. HMM's cannot, however, be used in an automated tagging schema, since they rely critically upon the calculation of statistics on output sequences (tag-states). The solution to the problem of being unable to automatically train HMMs is to employ the *Baum-Welch Algorithm*, also known as the *Forward-Backward Algorithm*. This algorithm uses word rather than tag information to iteratively construct a sequences to improve the probability of the training data (Rabiner 1989).

> *If a large lexicon (providing good coverage of the application domain) and some manually hand-tagged text are available, such methods perform automated POS-tagging in a computationally very efficient way (linear complexity) and with a very satisfying performance (on the average, 95-98% accuracy) (Guilder 1995).*

One of the important advantages of POS-tagging is to allow automated filtering of non-significant words on the basis of their morpho-syntactic category. Compared with Information Retrieval indexing, POS-tagging in more computational intensive and models the documents in grammatical level. As mentioned above, one difficulty of POS-tagging is so called Word Sense Disambiguation (WSD), whose aim is to identify the correct sense of a word in a particular context. Although there are various techniques (such as rule based tagging or HMM) to work around this barrier, WSD is yet a hard task of

NLP itself. Errors generated by POS-tagging can be propagated to following processing in IE system. Hence, compensation of such errors in IE is necessary, in order to perform high performance IE results.

3.2.2.2 NLP in IE systems

In this section we will review extraction patterns that are used only to process documents that contain grammatical, plain text. Such extraction rules are based on syntactic and semantic constraints that help identify the relevant information within a document. Consequently, in order to apply the extraction patterns below, one has to pre-process the original text with a syntactic analyser and a semantic tagger.

CIRCUS

The backbone of CIRCUS (Lehnert 1991) is a domain-specific dictionary of concept nodes. Concept nodes are structures that extract relevant information from a sentence (Lehnert 1991).

A concept node definition specifies a set of slots that extract information from text. Each slot extracts a particular type of information and contains a syntactic expectation that predicts where the information will be found in a clause. For example, $murder-passive$ contains two slots: a victim slot and a perpetrator slot. Since $murder-passive$ is activated only in a passive construction, the concept node predicts that the victim is the subject of the verb “murdered” and the perpetrator is the object of the preposition “by”. Figure 3 shows the concept node definitions for $murder-active$ and $murder-passive$.

Name:	$MURDER-ACTIVE$	$MURDER-PASSIVE$
Trigger Word:	murdered	murdered
Variable Slots:	((perpetrator (*SUBJECT* 1))	((victim (*SUBJECT* 1))
(perpetrator (*PP* (is-prep? '(by)))))		
Slot Constraints:	((class perpetrator *SUBJECT*) ((class victim *SUBJECT*) (class victim *DOBJ*)) (class perpetrator *PP*))	
Constant Slots:	(type murder)	(type murder)
Enabling Conditions:	((active))	((passive))

Figure 3: Two concept node definitions in CIRCUS

CIRCUS is one of initial IE systems of MUC. CIRCUS does not support learning and need such background knowledge as manually created domain

dependence dictionaries. The UMass/MUC-4 system (Lehnert et al. 1992) used 2 dictionaries: a part-of-speech dictionary containing 5436 lexical definitions, including semantic features for domain-specific words, and a dictionary of 389 concept node definitions for the terrorism domain. The concept node dictionary was manually constructed for MUC-4. To specify semantic preferences CIRCUS uses a set of hard and soft constraints. The hard constraints must be satisfied in order for the slot to be filled but the soft constraints act only as preferences for fillers. Therefore a slot may be filled even if a soft constraint is violated. Figure 4 shows a sample sentence and the resulting instantiated concept node produced by CIRCUS.

Sentence: Three peasants were murdered by guerrillas.

$MURDER-PASSIVE$
victim = “three peasants”
perpetrator = “guerrillas”

Figure 4: An instance concept node in CIRCUS

RAPIER

The RAPIER system (Califf & Mooney 1997) learns single slot extraction patterns that use limited syntactic information (e.g., the output of a part-of-speech tagger) and semantic class information (e.g., hypernim links from WordNet (Miller 1995)). In Figure 5, we show typical RAPIER extraction task: a sample document, the information to be extracted, and the extraction rule for the area slot. The RAPIER extraction pattern consist of three distinct slots: the Pre- and Post- “filler patterns” play the role of left and right delimiters, while the “Filler pattern” describes the structure of the information to be extracted.

Each “Filler pattern” consists of a (possibly empty) list of pattern items or pattern lists. The former matches exactly one word/symbol from the document, while the latter specifies a maximum length N and matches 0 to N words/symbols from the document. The constraints imposed by the pattern items/lists consists of exact match words, parts of speech, and semantic classes.

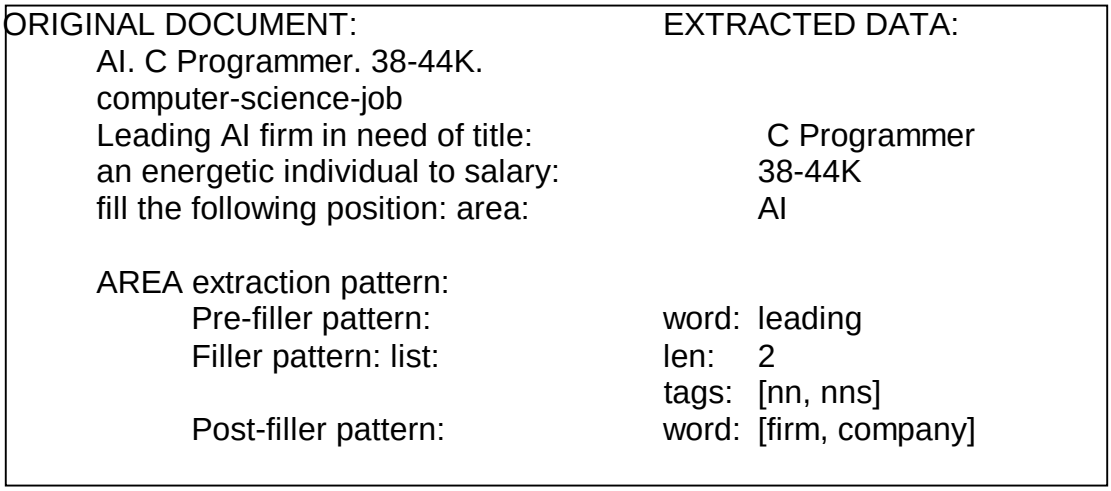

Figure 5: A RAPIER extraction task.

SRV

SRV (Freitag 1998) generates first-order logic extraction patterns that are based on attribute-value tests and the relational structure of the documents. For instance, the pattern from
Figure 6 extracts the names of the companies that were the target of an acquisition process. The extraction rule has the following meaning: the company name consists of a single, capitalised word (first two predicates) and is followed by a lower-case word (third predicate).

The most interesting constraints in Figure 6 are imposed by the last predicate, which uses features derived by the link grammar parser (Sleator & Temperley 1993) and WordNet (Miller 1995). The “right-AN” construct refers to the “right AN link” in a link grammar, which connects a noun modifier with the noun it modifies. In this example, the information to be extracted (i.e., “?A”) is connected by “[right-AN]” to the word that follows it, which, in turn, is one of the WordNet synset associated with stock.

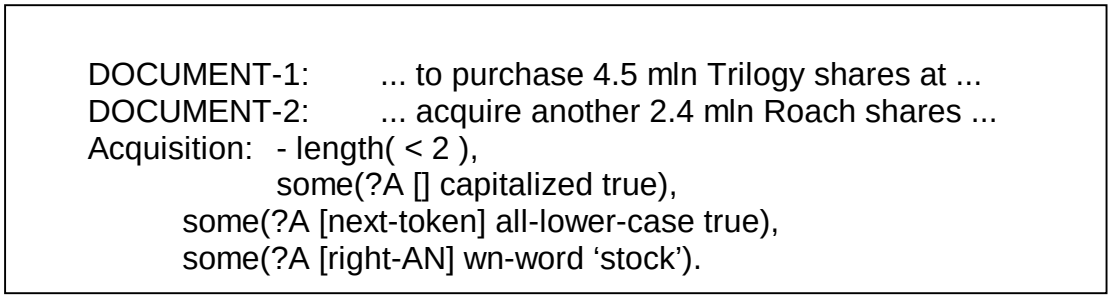

Figure 6: An SRV extraction task.

Other systems using NLP techniques

AutoSlog (Riloff 1993) builds a dictionary of extraction patterns that are called concepts or concept nodes. Each AutoSlog concept has a conceptual anchor that activates it and a linguistic pattern, which, together with the 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. 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.

LIEP (Huffman 1995) is a learning system that generates multi-slot extraction rules. That is, rather than learning one extraction pattern for each item of interest in a sentence (e.g., target and perpetrator), LIEP generates a single rule for all items of interest. The LIEP extraction rule extracts both the target and the perpetrator. The pattern consists of two sets of predicate-rules: the syntactic constraints and the semantic constraints.

The **PALKA** system (Kim & Moldovan 1995) learns extraction patterns that are expressed as frame-phrasal pattern structures (for short, FP-structures). An FP-structure consists of a meaning frame and a phrasal pattern. Each slot in the meaning frame defines an item-to-be-extracted together with the semantic constraints associated to it (e.g., *the target of the bombing event must be a physical object*). The phrasal pattern represents an ordered sequence of lexical entries and/or semantic categories taken from a predefined concept hierarchy.

The FP-structure combines the meaning frame and the phrasal pattern by linking the slots of the former to the elements of the latter. As opposed to AutoSlog, FP-structures can be activated both by exact match and via the is_a() relationship within the predefined concept hierarchy. However, it is interesting to note that PALKA is not strictly more expressive then AutoSlog because FP-structures can express exact word constraints only on verbs, while AutoSlog can also do it on nouns.

CRYSTAL (Soderland et al. 1995) generates multi-slot concept nodes that allow both semantic and exact word constraints on any component phrase. The concept definition includes semantic constraints on both the subject and the prepositional phrase, and it also imposes exact word matching for the verb and the preposition. Furthermore, the “Terms include:” construct introduces an additional exact word matching for the subject of the sentence.

CRYSTAL's extraction patterns are more expressive than the PALKA ones because the latter allow exact word matching only on the verb's root.

Furthermore, PALKA rules do not allow semantic constraints on prepositional phrases that do not contain relevant information.

The extraction patterns generated by **HASTEN** (Krupka 1995) are called Egraphs, and they can be seen as lists of (SemanticLabel, StructuralElement) pairs. Except for the ANCHOR semantic label, which is similar to AutoSlog's conceptual anchor, all other semantic labels represent potential items of interest. The structural element associated to the semantic label represents a set of syntactic and semantic constraints. In the extraction phase, HASTEN uses a similarity metric to compare an Egraph with the input text. In a first step, the system matches the structural elements and binds the semantic labels of the successfully matched structural elements. Then it uses a set of fixed weight factors to compute the percentage of the matched Egraph, and it compares the final score with a predefined threshold value.

Summary

All the NLP methods above are used to extract relevant data from grammatical, text. All of them use syntactic and semantic constraints to identify the items of interest. We can make some categories of the systems mentioned above following some criteria.

First, CIRCUS, AutoSlog, LIEP, PALKA, CRISTAL and HASTEN are classical MUC IE systems, they are only evaluated with MUC theorist domain. The systems are designed for free documents. RAIPIER and SRV are designed for practical using: RAIPIER is evaluated with Job Ads and SRV with seminar announcement. Those two domains are semi-structured document sets.

Second, the granularity of the extraction is not the same: CIRCUS, RAIPIER, LIEP and HASTEN identify the exact phrase of interest, while SRV, AutoSlog, PALKA, and CRYSTAL determine only the syntactic field that contains the target phrase.

Third, except for CRYSTAL, all other systems allow semantic class constraints only on the slots to be extracted (for the other sentence elements they allow exact word and verb root constraints), whereas CRYSTAL allows a multi-slot template filling.

All MUC systems mentioned above used MUC Theorist domain to evaluate the performance. Although results showed satisfying performance, it must be noted that such evaluation is somewhat artificial. The systems are developed especially for such domain and are not proved to be the portability. Consequently the NLP techniques used here are also domain dependent. The rules, the lexicon or the pre-defined patterns are designed only for one defined domain. This is one reason why the performance of such research systems is sometimes considerable good. RAPIER and SRV, on the other

hand, are developed for some practical applications such as newsgroup or email distributions. However, many of features used in SRV seem quite specific to the seminar announcement domain too. In contrast RAPIER seems to be the system with the best portability. But the result of RAPIER is sometimes not good: for Job Ads there are only 85% precision and 60% recall.

As mentioned above, the advantage of NLP is to analyse complicated free text. However, there is a shortcoming of IE methods with NLP techniques that can not be ignored, namely portability. Even for a well designed complete IE system, porting to a new domain is not an easy task.

3.3 Machine Learning Approaches

Based on the basic techniques, many well known Machine Learning approaches are used to perform high quality Information Extraction. Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Such tasks involve recognition, diagnosis, planning, robot control, prediction, etc.

There are several reasons why machine learning is important in Information Extraction. Some of these are:

- Some tasks cannot be defined well except by example; that is, we might be able to specify input/output pairs but not a concise relationship between inputs and desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and thus suitably constrain their input/output function to approximate the relationship implicit in the examples.

- It is possible that hidden among large piles of data are important relationships and correlations. For instance, template elements in semi-structured documents may have spatial relationships, which can not be found out by user easily. Machine learning methods can often be used to extract these relationships.

- Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on-the-job improvement of existing machine designs. Incremental machine learning algorithm (such as Case-based reasoning) can be used to improve extracting quality in the working phase.

- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down. For instance, an XSL-Pattern used to extract information from HTML documents can be more than 1000 bytes. Manual creating such patterns causes errors and is time consuming. With help of machine learning techniques patterns can be learned and exactly created. The knowledge mentioned above is particularly shallow knowledge but not deep knowledge. That is, machines are good at learning masses of knowledge that can be described by patterns, forms or rule etc.

- New knowledge about tasks is constantly being discovered by human's vocabulary changes. There is a constant stream of new events in the world. Continuing redesign of machine learning systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

There are two major settings in which we wish to learn a function. In one, called *supervised* learning, we know (sometimes only approximately) the values of f for the m samples in the training set, *S*. We assume that if we can find a hypothesis, h, that closely agrees with f for the members of *S*, then this hypothesis will be a good guess for, especially if *S* is large.

In the other setting, termed *unsupervised* learning, we simply have a training set of vectors without function values for them. The problem in this case, typically, is to partition the training set into subsets, S_1 , : : : , S_R , in some appropriate way. (We can still regard the problem as one of learning a function; the value of the function is the name of the subset to which an input vector belongs.) Unsupervised learning methods have application in taxonomic problems in which it is desired to invent ways to classify data into meaningful categories. (Mitchell 1997)

Many well known machine learning algorithms are widely used in the area Information Extraction. Examples are such as statistical approach (Xiao 2000, Hearst 1994, Berger et al. 1996, Richmond 1997), inductive rule learning (Soderland 1998, Xiao 2001b, Muslea 1999, Doorenbos et al. 1997, Kushmeric et al. 1997), Bayes (Freitag, 1998b), decision tree (Bennett, Aone, & Lovell, 1997, Matsumoto, 1997), hidden markov model (Seymore 1999, Kupiec 1992, Yamron et al. 1998, Stolcke, Shriberg, & others 1998, Bikel et al. 1997, Freitag & McCallum 1999), Case-based Reasoning (Aamodt 1994, Riloff 1994, Ashley, 1990; Hammond, 1986), Neural Network (Merkl 1996), text data mining ((Feldman and Hirsh 1997) etc.

3.4 Short Conclusion

As introduced in detail above, the both two basic technologies have their own advantages and disadvantages. The syntactic rules provide high speed, exact extracting, but are not able to process complicated concepts. The Nature Language Processing (NLP) methods, on the other hand, can be used for some sophisticated situations (such as Scenario Template production, ST). But they are difficult to build. Moreover, as mentioned above, NLP algorithms are domain specific and can result sometimes poor result. In our system, we primarily use syntactic rules as basic technology, because the syntactic rules provide a exact and high speed extracting. Based on this technology, we have hybrid techniques to enable system extracting complicated information and concepts.

Machine learning is very useful in the information extraction. Unlike Text Mining, in Information Extraction, useful information must be defined at first. Hence, examples must be given before learning. In our system we use primarily supervised machine learning methods. That is, from given examples, rules and patterns are generated by corresponding machine learning algorithms such as wrapper induction with XSL-pattern (see Section 7.1.2.2), hierarchical concept learning (see Section 8), statistical keyword generating (see Section 7.2.3) and statistical spatial model for interrelationship under elements in template (see Section 9). The system configuration to enable different learning processing is introduced also in Section 5.3.

Systems introduced above have normally defined extracting techniques for one document types (HTML or normal). Some systems are primarily developed for MUC evaluation sample set. Actually, they can not be used for practical domains. This research has presented a complete information extraction system not only for research projects but also for practical applications. The system is designed as a hybrid architecture embodying multiple agents that can not only be applied in parallel but also in collaboration in order to provide optimal overall performance. While this approach adds complexity, in terms of runtime coordination of multiple agents and, from a methodology perspective, in terms of having to marry the appropriate technique(s) to each relevant type of information, the hybrid approach is viewed as a fundamental necessity – the diversity of types and formats of textual information precludes a “silver bullet” solution to information extraction – i.e. no one technique can address all problems. In particular, the system clearly distinguishes between NE-tasks (which find isolated occurrences of information items of interest) and TE-tasks (which compile complete content descriptions form the information items). Moreover, it is supported by a general methodology that allows construction of the IE system to be carried out in a focussed and cost-effective way.