Information Extraction: Techniques and Challenges

Ralph Grishman

Computer Science Department
New York University
New York, NY 10003, U.S.A.

1 What is Information Extraction?

This volume takes a broad view of information extraction as any method for filtering information from large volumes of text. This includes the retrieval of documents from collections and the tagging of particular terms in text. In this paper we shall use a narrower definition: the identification of instances of a particular class of events or relationships in a natural language text, and the extraction of the relevant arguments of the event or relationship. Information extraction therefore involves the creation of a structured representation (such as a data base) of selected information drawn from the text.

The idea of reducing the information in a document to a tabular structure is not new. Its feasibility for sublanguage texts was suggested by Zellig Harris in the 1950's, and an early implementation for medical texts was done at New York University by Naomi Sager[20]. However, the specific notion of information extraction described here has received wide currency over the last decade through the series of Message Understanding Conferences [1, 2, 3, 4, 14]. We shall discuss these Conferences in more detail a bit later, and shall use simplified versions of extraction tasks from these Conferences as examples throughout this paper.

Figure 1 shows a simplified example from one of the earlier MUC's, involving terrorist events (MUC-3) [1]. For each terrorist event, the system had to determine the type of attack (bombing, arson, etc.), the date, location, perpetrator (if stated), targets, and effects on targets. Other examples of extraction tasks are international joint ventures (where the arguments included the partners, the new venture, its product or service, etc.) and executive succession (indicating who was hired or fired by which company for which position).

Information extraction is a more limited task than "full text understanding". In full text understanding, we aspire to represent in a explicit fashion *all* the information in a text. In contrast, in information extraction we delimit in advance, as part of the specification of the task, the semantic range of the output: the relations we will represent, and the allowable fillers in each slot of a relation.

2 Why the Interest in Information Extraction?

There has been a growing interest in developing systems for information extraction, of which this volume is just one indication. This interest represents a

19 March — A bomb went off this morning near a power tower in San Salvador leaving a large part of the city without energy, but no casualties have been reported. According to unofficial sources, the bomb — allegedly detonated by urban guerrilla commandos — blew up a power tower in the northwestern part of San Salvador at 0650 (1250 GMT).

INCIDENT TYPE	bombing
DATE	March 19
LOCATION	El Salvador: San Salvador (city)
PERPETRATOR	urban guerrilla commandos
PHYSICAL TARGET	power tower
HUMAN TARGET	-
EFFECT ON PHYSICAL TARGET	destroyed
EFFECT ON HUMAN TARGET	no injury or death
INSTRUMENT	bomb

Fig. 1. A terrorist report and a template of extracted information.

confluence of need and ability — observing what is possible with current natural language processing technology, and how the possible may indeed be useful.

An enormous amount of information exists only in natural language form. If this information is to be automatically manipulated and analyzed, it must first be distilled into a more structured form in which the individual "facts" are accessible. Most of the world's news, for example, is reported in newspapers, radio, and TV broadcasts. The best that current commercial technology has to offer is to (try to) retrieve relevant passages from a text archive. If we want to know who has signed contracts over the past year to deliver airplanes or natural gas, or which jurisdictions have enacted new restrictions on smoking, we must pour over reams of retrieved documents by hand. Information extraction offers the potential of extracting such data with much greater precision, producing lists of companies or cities rather than lists of documents. Equal benefits would accrue in processing more technical texts: extracting information from scientific journals, from legal decisions, or from hospital reports. Hospitals and medical researchers, in particular, are faced with a need to perform a wide range of retrospective analyses on reports collected primarily in natural language form.

Research groups in the 1950's and 1960's already recognized the potential value of automatically structuring natural language data, and projects were created for tasks such as the transformation of entire encyclopedia entries to structured form. Such projects, however, faced a broad range of problems in natural language processing and knowledge representation, many of which still lack effective solutions. More recently, it has been recognized that by setting a goal of *selective* information structuring — i.e., information extraction — we can define a range of tasks that appears within reach of current technology.

A mature information extraction technology would allow us to rapidly create extraction systems for new tasks whose performance was on a par with human performance. We are not there yet, for reasons to be discussed later in this paper.

However, systems with more modest performance (which miss some events and include some errors) can be of value. They can be used to extract information for later manual verification.[1] They can also be useful in circumstances where there would not be time to review all the source documents, and incomplete extracted information is better than no information.

Recent research (stimulated, in part, by the MUC conferences) has shown that such modest extraction systems can – in some cases – be implemented using relatively simple natural language analysis methods. Current methods will be successful if the information to be extracted is expressed directly (so that no complex inference is required), is predominantly expressed in a relatively small number of forms, and is expressed relatively locally within the text.[2]

The following section describes the structure and components of an information extraction system. Then, after a brief discussion of system evaluation, we examine some of the current issues in trying to advance the state of the art of information extraction.

3 The Basic Techniques

3.1 The Overall Flow

The process of information extraction has two major parts. First, the system extracts individual "facts" from the text of a document through local text analysis. Second, it integrates these facts, producing larger facts or new facts (through inference). As a final step after the facts are integrated, the pertinent facts are translated into the required output format.

The individual facts are extracted by creating a set of patterns to match the possible linguistic realizations of the facts. Because of the complexity of natural language (except in the most restricted of sublanguages), it is not practical to describe these patterns directly as word sequences. So, as in most natural language processing systems, we begin by structuring the input, identifying various levels of constituents and relations, and then state our patterns in terms of these constituents and relations. This process typically begins with lexical analysis (assigning parts-of-speech and features to words and idiomatic phrases through morphological analysis and dictionary lookup) and name recognition (identifying names and other special lexical structures such as dates, currency expressions, etc.). This is followed by a full syntactic analysis (parse), or – in most current systems – by some form of partial parsing to identify noun groups, verb groups, and possibly head-complement structures. After all this is done, we use task-specific patterns to identify the facts of interest.

[1] A process which may offer economic benefits when compared to purely manual extraction, in the same way that machine translation followed by post-editing may be more efficient than manual translation.

[2] Bagga and Biermann [7] discuss the relation between the success rate of extraction systems and the locality of the information in the text, measured in terms of the number of syntactic relations involved.

The integration phase examines and combines facts from the entire document or discourse. It resolves relations of coreference, which can range from the use of pronouns to multiple descriptions of the same event. It may also need to draw inferences from the explicitly stated facts in the document. The overall flow is shown in Figure 2.

Following the terminology established by the Message Understanding Conferences, we shall call the specification of the particular events or relations to be extracted a *scenario*. Thus, we distinguish between a general *domain*, such as financial news, and a particular scenario, such as international joint ventures or aircraft sales. We shall refer to the final, tabular output format of information extraction as a *template*.

3.2 Pattern Matching and Structure Building

In the remainder of this section, we shall look at each of the stages of processing. As we go through the stages, we shall focus on a simplified version of the MUC-6 scenario, involving executive succession, and shall follow the progress of the following brief document

> Sam Schwartz retired as executive vice president of the famous hot dog manufacturer, Hupplewhite Inc. He will be succeeded by Harry Himmelfarb.

as it is gradually transformed into the two templates shown in Figure 3. The details we present will be those of our own (New York University Proteus Project) extraction system[3] [11], but we will note some of the places where they differ significantly from those of other systems.

In our system, and many other current extraction systems, most of the text analysis is performed by matching the text against a set of regular expressions. If the expression matches a segment of text, the text segment (constituent) is assigned a label, and possibly one or more associated features. The patterns are organized in sets, and constituent labels assigned in one pattern set may be referenced in patterns in subsequent sets. In effect, we perform a limited form of deterministic, bottom-up parsing.[4]

[3] Although with many simplifications from our actual implementation.

[4] The overall control of the matching process differs substantially among pattern-matching extraction systems. In the NYU system, each pattern has an associated set of actions; the main action generally is the tagging of a text segment with a new label, but other actions may be performed. The pattern sets are applied one at a time. All the patterns in a set are matched starting at the first word of the sentence. If more than one pattern matches, the one matching the longest segment is selected; if more than one pattern matches the longest segment, the first is taken. The actions associated with that pattern are executed. If no pattern matched, the patterns are reapplied starting at the next word of the sentence; if a pattern matched and an action labeled a text segment, the patterns are reapplied past the end of that segment. This process continues until the end of the sentence is reached.

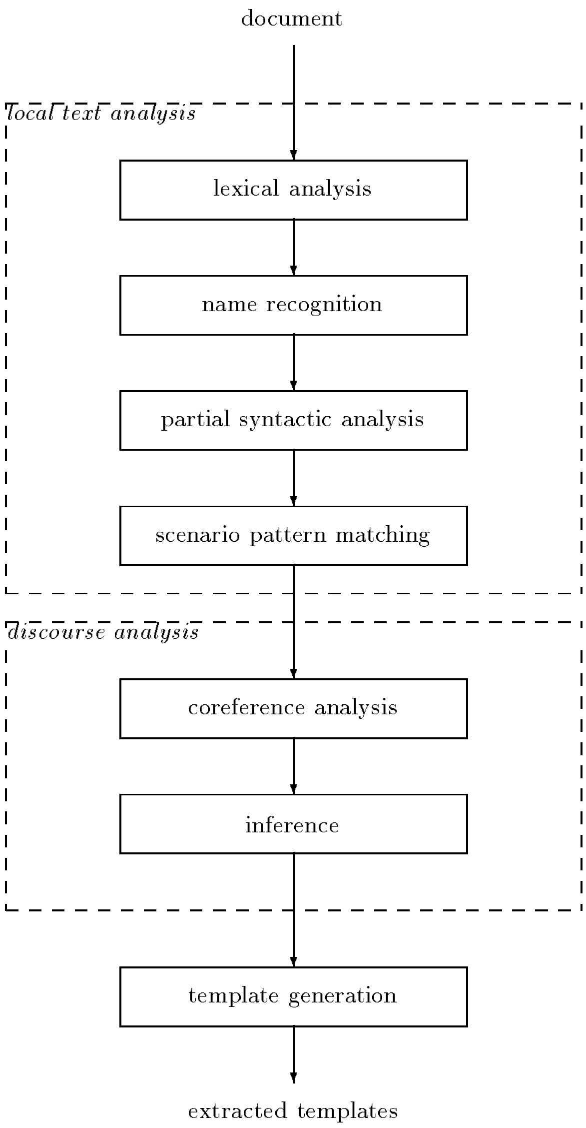

Fig. 2. Structure of an information extraction system.

Associated with some of the constituents in our system are semantic structures called *entities* and *events*. These structures will ultimately be used to construct the templates.

3.3 Lexical Analysis

The text is first divided into sentences and into tokens. Each token is looked up in the dictionary to determine its possible parts-of-speech and features. The Pro-

EVENT	leave job
PERSON	Sam Schwartz
POSITION	executive vice president
COMPANY	Hupplewhite Inc.

EVENT	start job
PERSON	Harry Himmelfarb
POSITION	executive vice president
COMPANY	Hupplewhite Inc.

Fig. 3. Events extracted from Hupplewhite text.

teus dictionary includes the Comlex Syntax dictionary (a large, general-purpose English dictionary [5]) [12] plus various special dictionaries, such as dictionaries of major place names, major companies, common (American) first names, and company suffixes (such as "Inc."). In our example "Sam" and "Harry" would be tagged as first names; Inc. will be tagged as a company suffix.[6]

3.4 Name Recognition

The next phase of processing identifies various types of proper names and other special forms, such as dates and currency amounts. Names appear frequently in many types of texts, and identifying and classifying them simplifies further processing; names, furthermore, are important as argument values for many extraction tasks.

Names are identified by a set of patterns (regular expressions) which are stated in terms of parts-of-speech, syntactic features, and orthographic features (e.g., capitalization). Personal names, for example, might be identified by a preceding title

Mr. Herrington Smith

by a common first name

Fred Smith

by a suffix

Snippety Smith Jr.

or by a middle initial

Humble T. Hopp

[5] Available through the Linguistic Data Consortium.

[6] In addition, our system uses a probabilistic part-of-speech tagger from BBN to exclude unlikely part-of-speech assignments; the system can operate without the tagger, but with slightly degraded performance.

Company names can usually be identified by their final token(s), such as

Hepplewhite Inc.
Hepplewhite Corporation
Hepplewhite Associates
First Hepplewhite Bank

However, some major company names (e.g., "General Motors") may be mentioned without any such overt clues, so it is important to also have a dictionary of major companies.

In our example passage, three names will be identified:

$[_{\textit{name type: person}}$Sam Schwartz] retired as executive vice president of the famous hot dog manufacturer, $[_{\textit{name type: company}}$Hupplewhite Inc.] He will be succeeded by $[_{\textit{name type: person}}$Harry Himmelfarb].

Name identification typically also includes the processing required to identify *aliases* of a name — in effect, name coreference. For example, a system would identify "Larry Liggett" with a subsequent mention of "Mr. Liggett", and "the Hewlett-Packard Corp." with "HP". Alias identification may also help name classification. Thus, if we read "Humble Hopp reported ..." we may not know if "Humble Hopp" is a person or company, but if there is a subsequent reference to "Mr. Hopp", the ambiguity is resolved.

Name identification has been worked on quite intensively for the past few years, and has been incorporated into several products which extract classified name lists from documents. The highest performing systems at present use large numbers of hand-coded patterns, but the performance of systems which learn rules from annotated corpora has been steadily improving, and is now only a few percent below that of the hand-coded systems [8].

3.5 Syntactic Structure

Identifying some aspects of syntactic structure simplifies the subsequent phase of fact extraction. After all, the arguments to be extracted often correspond to noun phrases in the text, and the relationships to be extracted often correspond to grammatical functional relations. On the other hand, the identification of the complete syntactic structure of a sentence is a difficult task. As a result, there is a great variation in the amount of syntactic structure which is explicitly identified. The trade-offs will be discussed further below (section 5.1).

Some systems don't have any separate phase of syntactic analysis. Others attempt to build a complete parse of a sentence. Most systems fall in between, and build a series of parse fragments. In general, they only build structures about which they can be quite certain, either from syntactic or semantic evidence. The current NYU Proteus system, following the lead of the SRI FASTUS system [6, 5], builds structures for noun groups (a noun plus its left modifiers) and for verb groups (a verb with its auxilliaries); both of these can be built in most cases using just local syntactic information. In addition, it builds certain larger noun phrase structures (conjoined noun groups, noun groups with appositional modifiers) if it has semantic information to confirm the correctness of the structure. All of this is done using the same regular expression pattern matcher; unlike some other systems, no special procedures are used for parsing.

The first set of patterns labels all the basic noun groups as noun phrases (*np*); in our example, this includes the three names, the pronoun, and two larger noun groups. It is followed by a set to label the verb groups (*vg*). After these patterns have been applied, the text is labeled as follows:

> [$_{np\ entity:\ e1}$ Sam Schwartz] [$_{vg}$ retired] as [$_{np\ entity:\ e2}$ executive vice president] of [$_{np\ entity:\ e3}$ the famous hot dog manufacturer], [$_{np\ entity:\ e4}$ Hupplewhite Inc.] [$_{np\ entity:\ e5}$ He] [$_{vg}$ will be succeeded] by [$_{np\ entity:\ e6}$ Harry Himmelfarb].

Associated with each constituent are certain features which can be tested by patterns in subsequent stages. For vg's these include information on tense (past / present / future), voice (active / passive), and the root form of the verb; np's have information on the root form of the head (including whether or not it is a name) and syntactic number. In addition, for each np the system creates a "semantic" *entity*; for our example, following entities will be created:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president"
entity e3	type: manufacturer
entity e4	type: company name: "Hupplewhite Inc."
entity e5	type: person
entity e6	type: person name: "Harry Himmelfarb"

The sets of patterns which follow build up larger noun phrase structures by attaching right modifiers. Because of the syntactic ambiguity of right modifiers,[7] these patterns incorporate some semantic constraints and therefore, unlike the noun and verb group patterns, are domain specific. These patterns typically coalesce two noun phrases, and possible intervening words, into a larger noun phrase, and modify the entity associated with the head noun to incorporate information from the modifier.

For our example text, the two relevant patterns will recognize the appositive construction

> *company-description*, *company-name*,

and the prepositional phrase construction

> *position* of *company*

[7] For example, sequences which look like apposition might instead be part of conjoined structures; prepositional phrases to the right of a noun might attach to a preceding verb.

In the second pattern, *position* represents a pattern element which matches any np whose entity is of type "position" and *company* matches any np whose entity is of type "company". The system includes a small semantic type hierarchy (an *isa* hierarchy), and the pattern matching uses the *isa* relation, so any subtype of *company* (such as, in our example, *manufacturer*) will be matched. In the first pattern, *company-name* specifies an np of type *company* whose head is a name; *company-description* specifies an np of type *company* whose head is a common noun. These patterns produce the following labeling[8]

> [$_{np\ entity:\ e1}$ Sam Schwartz] [$_{vg}$ retired] as [$_{np\ entity:\ e2}$ executive vice president of the famous hot dog manufacturer, Hupplewhite Inc.] [$_{np\ entity:\ e5}$ He] [$_{vg}$ will be succeeded] by [$_{np\ entity:\ e6}$ Harry Himmelfarb].

and the entities are updated as follows:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e5	type: person
entity e6	type: person name: "Harry Himmelfarb"

3.6 Scenario Pattern Matching

All of the processing until now has been in a sense preparatory for the scenario pattern matching. The role of these patterns is to extract the events or relationships relevant to the scenario. For the example of executive succession we have been following, there will be two such patterns,

> *person* retires as *position*

and

> *person* is succeeded by *person*

where *person* and *position* are pattern elements which match np's with the associated type. "retires" and "is succeeded" are pattern elements which match active and passive verb groups, respectively. The result is a text labeled with two clauses, each pointing to an event structure; these event structures point in turn to the previously created entities:

> [$_{clause\ event:\ e7}$ Sam Schwartz retired as executive vice president of the famous hot dog manufacturer Hupplewhite Inc.] [$_{clause\ event:\ e8}$ He will be succeeded by Harry Himmelfarb].

and the entities / events are updated as follows:

[8] In our current system, if a new label subsumes earlier labels, the earlier labels are no longer visible for subsequent patterns, so we have removed these labels from the text, and removed entity e4.

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e5	type: person
entity e6	type: person name: "Harry Himmelfarb"
event e7	type: leave-job person: e1 position: e2
event e8	type: succeed person1: e6 person2: e5

3.7 Coreference Analysis

Coreference analysis has the task of resolving anaphoric references by pronouns and definite noun phrases. In our little text, the only example is the pronoun "he" (entity e5). Coreference analysis will look for the most recent previously mentioned entity of type *person*, and will find entity e1. It will then in effect change references to e5 to refer to e1 instead, leaving us with the following events and entities:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e6	type: person name: "Harry Himmelfarb"
event e7	type: leave-job person: e1 position: e2
event e8	type: succeed person1: e6 person2: e1

The coreference module also makes use of the *isa* hierarchy, so that if a reference to "the company" appeared in the text, it would be properly resolved to entity e3, the manufacturer.

3.8 Inferencing and Event Merging

In many situations, partial information about an event may be spread over several sentences; this information needs to be combined before a template can be generated. In other cases, some of the information is only implicit, and needs to be made explicit through an inference process.

In the executive succession domain, we need to determine what the "succeed" predicate implies, if we wish to ultimately produce templates specifying that particular individuals got or lost particular positions. For example, if we have

Sam was president. He was succeeded by Harry.

we infer that Harry will become president; conversely, if

Sam will be president; he succeeds Harry.

we infer that Harry was president. Such inferences can be implemented by production system rules,[9] such as

[9] In fact, these inferences were hard coded in our system for MUC-6, but production systems with similar rules were used in most of our prior MUC systems [13].

leave-job(X-person,Y-job) & succeed(Z-person,X-person)
⇒ start-job(Z-person,Y-job)

start-job(X-person,Y-job) & succeed(X-person,Z-person)
⇒ leave-job(Z-person,Y-job)

This will leave us with the following events:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e6	type: person name: "Harry Himmelfarb"
event e7	type: leave-job person: e1 position: e2
event e8	type: succeed person1: e6 person2: e1
event e9	type: start-job person: e6 position: e2

which can be translated into the templates shown earlier as Figure 3.

The simple scenario shown here did not require us to take account of the time of each event. For many scenarios, however, time is important: either explcit times must be reported, or the sequence of events is significant. In such cases, time information may be derived from many sources, including absolute dates and times ("on April 6, 1995"), relative dates and times ("last week"), verb tenses, and knowledge about the inherent sequence of events. Since time analysis may interact with other inferences about the discourse, it will normally be performed as part of this stage of processing.

4 Evaluation

As we noted at the beginning, one of the unusual aspects of information extraction is the degree to which its development over the past decade has been fostered and shaped by a series of evaluations, the Message Understanding Conferences (MUC). So, 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.

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 organizer. Meanwhile, the organizer 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

$$\text{precision} = \frac{N_{correct}}{N_{response}}$$
$$\text{recall} = \frac{N_{correct}}{N_{key}}$$

Sometimes an "F score" is also used as a combined recall-precision score; $F = (2 \times \text{precision} \times \text{recall})/(\text{precision} + \text{recall})$.

5 Design Issues

5.1 To Parse or not to Parse

As we noted above, one of the most evident differences among extraction systems involves the amount of syntactic analysis which is performed. The benefits of syntax analysis were noted above. In particular, the arguments we wish to collect during scenario pattern matching are usually connected by grammatical relations; for example, we want to extract the subject and object of verbs such as "hire", "fire", and "succeed". Thus, if syntactic relations were already correctly marked, we would expect the scenario patterns to be simpler and more accurate. These considerations motivated many early extraction systems to perform full syntactic analysis before looking for scenario patterns.

However, building a complete syntactic structure is not easy. Some decisions, such as conjunction scope and the attachment of modifiers, are particularly difficult. In principle, full sentence analyzers should be able to use global constraints to resolve local ambiguities. In fact, however, because of the inevitable gaps in grammatical and lexical coverage, full sentence parsers may end up making poor local decisions about structures in their quest to create a parse spanning the entire sentence. In effect, global constraints may make things worse. Furthermore, full sentence parsers are relatively expensive of computer time — not surprising, since they have a large space to search. Finally, we should note that for information extraction we are only interested in the grammatical relations relevant to the scenario; correctly determining the other relations may be a waste of time.

As a result, most current high performance extraction systems create only partial syntactic structures; syntactic structures which can be created with high confidence and using local information. Some systems, such as the BBN system [22], use separate parsing procedures. In our own work, the considerations just listed led us to shift from a full parsing approach (employed for earlier MUCs) to the partial parsing described here. Specifically, we followed the approach adopted by FASTUS [6, 5], which brackets noun and verb groups; both of these structures are quite reliably identified by local information.[10] In addition, as illustrated in

[10] The most common ambiguity regarding noun groups involves leading present participles, as in the classic example "They are flying planes." In such cases, we bracket the

our earlier example, we identify some larger noun phrase constituents if there is semantic evidence to confirm the correctness of our attachment.

In general, we can say that most current extraction systems parse *conservatively* — only if there is strong evidence of the correctness of a reduction. This gains most (though not all) of the benefits of a full parse, while not introducing errors which would block applications of scenario patterns.

Traditional syntactic analysis has the role not only of *identifying* syntactic structure but also of *regularizing* syntactic structure. In particular, different clausal forms, such as active and passive forms, relative clauses, reduced relatives, etc. are mapped into essentially the same structure. This regularization simplifies the scenario pattern matching — it need deal with fewer forms of each scenario pattern.

If we employ a partial parsing approach which does not perform such regularization, we must have separate scenario patterns for each syntactic form, in effect multiplying the number of patterns to be written by a factor of 5 to 10. For example, we would need separate patterns for

IBM hired Harry
Harry was hired by IBM
IBM, which hired Harry, ...
Harry, who was hired by IBM, ...
Harry, hired by IBM, ...
etc.

To lessen this burden, the SRI FASTUS system [5] and the NYU Proteus system [11] have recently introduced *metarules* or *rule schemata*: methods for writing a single basic pattern and having it transformed into the patterns needed for the various syntactic forms of a clause. Thus we might write something like

subject=*company* verb=hired object=*person*

and the system would generate the patterns

company hired *person*
person was hired by *company*
company, which hired *person*
person, who was hired by *company*
person, hired by *company*
etc.

This approach bears some similarity to the metarules of GPSG [Generalized Phrase Structure Grammar], which expand a small set of productions into a larger set involving the different clause-level structures.

Clausal patterns need also to account for modifiers which can intervene between the sentence elements of interest. For example, we may find between the subject and verb either sentence modifiers

minimal noun group, excluding the present participle, and make allowances in subsequent patterns for the possibility that there may be an unattached present participle preceding the noun group.

IBM *yesterday* promoted Mr. Smith to executive vice president.

or noun modifiers which are not accounted for by the noun phrase patterns

GE, *which was founded in 1880*, promoted Mr. Smith to president.

With our partial parsing approach, we would need to include optional pattern elements separately in each clausal pattern to match such modifiers. With the metarule mechanism, we can automatically insert such pattern elements into the appropriate positions in each clausal pattern. In this way we can regain some of the benefits of syntactically-based clausal analysis.

The widespread use of partial parsing is a reflection of the current state of full-sentence parsing technology: experience shows that such parsers are likely to introduce more errors than they fix for information extraction tasks. However, this parsing technology is improving rapidly, thanks largely to corpus-based methods. A few years ago, good hand-coded grammars, operating on newspaper texts, got recall and precision scores in the range of 60 to 70%.[11] Recently, several parsers trained on large hand-bracketed corpora have gotten recall and precision scores above 80% [18, 9], and the performance is steadily improving. If the performance gets good enough, it may be worth revisiting the decision about using full-sentence parsing.

5.2 Portability

One of the barriers to making information extraction a practical technology is the cost of adapting an extraction system to a new scenario. In general, each application of extraction will involve a different scenario. If implementing this scenario requires a few months of effort and the skills of the extraction system designers, the market for extraction systems will remain limited indeed. We need to have tools which will allow potential users to adapt such a system, and create an initial system in days or weeks, not months.

The basic question in developing such a customization tool is the form and level of the information to be obtained from the user (assuming that the goal is to have the customization performed directly by the user rather than by a expert system developer). If we are using a “pattern matching” system, most work will probably be focused on the development of the set of patterns. However, changes will also be needed to the semantic hierarchy, to the set of inference rules, and to the rules for creating the output templates.

Unless users have considerable experience with writing patterns (regular expressions with associated actions) and some familiarity with formal syntactic structure, they may find it difficult and inconvenient to operate directly on the patterns. One possibility is to provide a graphical representation of the patterns (i.e., a finite state network), but this still exposes many of the details of the

[11] Using the Parseval metric, which compares the bracketing produced by a system to that of the University of Pennsylvania Tree Bank.

patterns. Instead, several groups are developing systems which obtain information primarily from *examples* of sentences of interest and the information to be extracted.

One of the first sites to experiment with this approach was the University of Massachusetts at Amherst. Their system, as developed for MUC-3, relied on a large number of small, lexically-triggered patterns which built individual concepts; these concepts were then consolidated to create the information needed for template filling [17]. For MUC-4, they developed a tool ("AutoSlog") to create such patterns semi-automatically from the development corpus with its templates [16, 19]. Given a template slot which is filled with words from the text, such as a name, their program would search for these words in the text and would hypothesize a pattern based on the immediate context of these words (for example, the governing verb of a noun phrase). These patterns would then be presented to a system developer, who could accept or reject the pattern. More recently, they have developed a system which uses machine learning techniques and does not rely on any human review [21, 10]. This system seeks to generalize and merge the patterns derived from individual examples, checking that the resulting patterns do not overgenerate (do not match corpus examples which are not marked as being relevant to the scenario).

Several of the earlier MUCs involved large training corpora, with over a thousand documents and their templates; such corpora encouraged such a corpus-based approach. However, the centralized preparation of large, consistent training corpora proved to be an expensive proposition; this suggested that such large corpora would not be available for most real tasks. Users may only be willing to prepare a few examples, or at best a few dozen examples, of filled templates. Experiments with smaller training collections, such as the 100 documents provided for MUC-6, suggest that fully automated learning techniques, when provided only with text examples and associated templates, and with minimal automatic syntactic generalization, may not be able to achieve sufficient coverage [10].

It is possible to compensate in part for the lack of training data by providing more information about each example. In HASTEN, the system developed by SRA for MUC-6 [15], the developer builds a structural description of each example, marking the type of constituent, the constraints on the constituent, and the semantic label of each constituent. This approach was able to achieve a good level of performance on the MUC-6 task, but requires some expertise on the part of the developer to mark up the examples.

At NYU, we are building an *interactive* tool for customizing an extraction system. We believe that this will provide the most efficient approach to acquisition in situations where the user does not have a large, pre-annotated corpus (and, in fact, may be refining the scenario in response to new examples), and does not have the expertise to create patterns unaided. The user begins by providing an example (normally drawn from the corpus) and the fact (template) to be extracted. The system responds by using the existing patterns to create a structural description of the example. It then can interact with the user to extend and generalize the example, both syntactically and semantically. Syntactic generalizations can be produced through the metarule mechanism discussed above; semantic generalizations can be produced through the *isa* hierarchy. In this way, it is possible to quickly extend a single example into a pattern with broad coverage. In addition, by allowing the user to see other examples in the corpus which match the generated pattern, it is possible to insure that the pattern is not being overgeneralized.[12]

5.3 Improving Performance

The other major barrier to the widespread use of extraction systems is the limitation on performance. Any observer of the most recent MUC, MUC-6, would be struck by the relatively similar level of performance of the top-ranked systems. Five of the nine systems got F scores in the range of 51 to 56 (reflecting recall of 43 to 50% and precision of 59 to 70%) [4].

What can account for such a clustering of performance? Any response at present must be speculative. In part, it reflects a convergence of technologies; the top systems are fairly similar in their overall design. However, it probably also reflects characteristics of the task itself. It seems reasonable from what we know of other linguistic phenomena (e.g., the distribution of vocabulary or syntactic structures) that a large fraction of the relevant facts are encoded linguistically by a small number of forms (a small number of lexical items, a small number of syntactic structures, etc.). As a result, it is relatively easy (once a reasonable framework has been established) to reach some middling level of performance. Beyond that point, one is working on the "tail" of the distribution of linguistic phenomena, and further improvement becomes increasingly expensive.

How can we hope to move further along the tail — improve extraction performance — with a fixed investment of labor for each new scenario? Some of the shortcomings represent scenario-independent phenomena. These include more complex syntactic structures, such as different types of subordinate clauses, and more complex anaphoric phenomena, such as anaphors with split antecedents. Temporal information plays a significant role in many extraction tasks, and most temporal processing can be made independent of the specific scenario. Thus, as these aspects of the core extraction system are gradually enhanced, we can expect an improvement in performance across all scenarios.

Other shortcomings in performance reflect a lack of knowledge relative to a specific scenario, and will be more difficult to address. As tools for interacting with users to acquire and generalize patterns improve, we can expect to augment this knowledge more rapidly. For example, a tool which suggests related lexical items can broaden lexical coverage beyond what is seen in the training corpus. In addition, as more extraction scenarios are implemented, we can expect to see pattern sets which are applicable to families of related scenarios or to entire domains. For example, patterns for basic actions such as the purchase and sale of goods may be applicable to many scenarios within the domain of business.

[12] Such interactive analysis requires a fast extraction system; the current NYU system can process a typical newspaper article in about 2 seconds on a high-end PC.

References

1. *Proceedings of the Third Message Understanding Conference (MUC-3)*. Morgan Kaufmann, May 1991.
2. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. Morgan Kaufmann, June 1992.
3. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, MD, August 1993. Morgan Kaufmann.
4. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
5. Douglas Appelt, Jerry Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Meyers, and Mabry Tyson. SRI International FASTUS system: MUC-6 test results and analysis. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
6. Douglas Appelt, Jerry Hobbs, John Bear, David Israel, and Mabry Tyson. FASTUS: A finite-state processor for information extraction from real-world text. In *Proc. 13th Int'l Joint Conf. Artificial Intelligence (IJCAI-93)*, pages 1172–1178, August 1993.
7. Amit Bagga and Alan Biermann. Analyzing the performance of message understanding systems. Technical Report CS-1997-01, Dept. of Computer Science, Duke University, 1997.
8. Daniel Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proc. Fifth Applied Natural Language Processing Conf.*, Washington, DC, April 1997. Assn. for Computational Linguistics.
9. Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proc. 34th Annual Meeting Assn. Computational Linguistics*, pages 184–191, Santa Cruz, CA, June 1996.
10. David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. Description of the UMass system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
11. Ralph Grishman. The NYU system for MUC-6 or where's the syntax? In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
12. Ralph Grishman, Catherine Macleod, and Adam Meyers. Comlex Syntax: Building a computational lexicon. In *Proc. 15th Int'l Conf. Computational Linguistics (COLING 94)*, pages 268–272, Kyoto, Japan, August 1994.
13. Ralph Grishman, Catherine Macleod, and John Sterling. New York University: Description of the Proteus System as used for MUC-4. In *Proc. Fourth Message Understanding Conf. (MUC-4)*, pages 233–241, McLean, VA, June 1992.
14. Ralph Grishman and Beth Sundheim. Message Understanding Conference - 6: A brief history. In *Proc. 16th Int'l Conf. on Computational Linguistics (COLING 96)*, Copenhagen, August 1996.
15. George Krupka. SRA: Description of the SRA system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
16. W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. University of Massachusetts: MUC-4 test results and analysis. In *Proc. Fourth Message Understanding Conf.*, McLean, VA, June 1992. Morgan Kaufmann.

17. W. Lehnert, C. Cardie, D. Fisher, E. Riloff, and R. Williams. University of Massachusetts: Description of the CIRCUS system as used for MUC-3. In *Proc. Third Message Understanding Conf.*, San Diego, CA, May 1991. Morgan Kaufmann.
18. David Magerman. Statistical decision-tree models for parsing. In *Proc. 33rd Annual Meeting Assn. Computational Linguistics*, pages 276–283, Cambridge, MA, June 1995.
19. Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proc. 11th Annl. Conf. Artificial Intelligence*, pages 811–816, 1993.
20. Naomi Sager, Carol Friedman, and Margaret Lyman. *Medical Language Processing: Computer Management of Narrative Data.* Addison Wesley, 1987.
21. W. Soderland, D. Fisher, J. Aseltine, and W. Lenhert. CRYSTAL: Inducing a conceptual dictionary. In *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI-95)*, pages 1314–1319, Montreal, Canada, 1995.
22. Ralph Weischedel. BBN: Description of the PLUM system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.