TU

Technische Universität Wien

DIPLOMARBEIT

Extracting Process Information from Clinical Practice Guidelines

ausgeführt am Institut für

Softwaretechnik und Interaktive Systeme

der Technischen Universität Wien

unter der Anleitung von

a. o. Univ. Prof. Dr. Silvia Miksch und Mag. Katharina Kaiser

durch

Cem Akkaya, 0026749

in der Studienrichtung

Diplomstudium Informatik, 881

Wien, Mai 2005

Abstract

Each day, more and more text data is made available in electronic form generating huge repositories of knowledge. To efficiently process such a huge amount of information needs special information management techniques and tools. Especially, the medical domain offers many application areas for these techniques. One of these is the creation of computerized medical guidelines which offers many advantages in patient management by allowing computer-supported execution of clinical guidelines. Unfortunately, available tools for this task are restricted in the way that they only allow a manual computerization process. This feature makes generation of computerized medical guidelines a cumbersome task even though by the presence of sophisticated modelling frameworks. Therefore, there is a need for automation of this process. Information Extraction techniques can meet this need.

Along with this thesis, an Information Extraction framework (CPGPro) based on knowledge engineering approach is implemented. CPGPro extracts relevant clinical actions and relations among them from otolaryngology guidelines automatically for subsequent processing by other tools. Processing is done in subsequent stages based on handcrafted lexical resources and extraction rules, which are implemented as heuristics derived from linguistic patterns encountered in otolaryngology guidelines. The documents of interest are XHTML-conform. Therefore CPGPro makes use of delimiters by defining extraction patterns not only on syntactic/semantic constraints as it is common in conventional Information Extraction systems but also on delimiters that bound the text. The fact that XHTML-conform guidelines consist of both structured- and free text made CPGPro a hybrid between conventional Information Extraction systems and wrapper tools.

The results show that it is possible to extract relevant actions by very simple natural language analysis methods like heuristics used in CPGPro. This phenomenon can be explained by the fact that actions in these documents are usually expressed in small number of forms with common attributes, but identifying relations among actions needs more sophisticated natural language analysis like syntactic- and coreference analysis.

Kurzfassung

Die Menge an Textdaten in elektronischer Form vergrößert sich von Tag zu Tag. Diese Daten erzeugen sehr große Mengen an Wissen. Die wirksame Verarbeitung solch einer großen Menge von Information setzt spezielle Verwaltungstechniken und Werkzeuge voraus. Besonders der Bereich der Medizin bietet viele Verwendungsgebiete für diese Techniken an. Eine davon ist die Erstellung von rechnergestützten medizinischen Leitlinien, die viele Vorteile zur Verwaltung von Patienten anbietet, indem es die computerunterstützte Ausführung der medizinischen Leitlinien erlaubt. Zu diesem Zweck entwickelte Werkzeuge erlauben allerdings nur einen manuellen Modellierungsvorgang. Trotz hoch entwickelter Modellierungsframeworks erschwert diese Einschränkung die Erzeugung rechnergestützter medizinischer Leitlinien. Methoden der Informationsextraktion können diesen Mangel beseitigen.

Im Rahmen dieser Arbeit wird ein Informationsextraktion Framework (CPGPro) eingeführt, das auf Heuristiken basiert. CPGPro extrahiert automatisch relevante klinische Prozesse und Relationen aus Leitlinien der Hals-Nasen-Ohren-Heilkunde für die nachfolgende Verarbeitung durch andere Werkzeuge. Die Verarbeitung wird in den autonomen und nachfolgenden Stadien durchgeführt, die auf ein handgefertigtes Lexicon und handgefertigte Extraktionsregeln basieren, die als Heuristiken eingeführt werden. Die Heuristiken werden von den linguistischen Mustern abgeleitet, die in den HNO-Leitlinien gefunden wurden. Die Dokumente, die verarbeitet werden, sind XHTML-konform. Deshalb verwendet CPGPro auch Auszeichnungsmarkierungen, indem es Extraktionregeln nicht nur mittels syntaktischen und semantischen Einschränkungen definiert, wie es in den herkömmlichen Informationsextraktionsystemen üblich ist. Die Tatsache, dass XHTML-konforme Leitlinien sowohl strukturierten als auch freien Text aufweisen, macht den maßgeschneiderten CPGPro zu einem Hybriden zwischen herkömmlichen Informationsextraktionsystemen und Wrapper-werkzeugen.

Die Resultate zeigen, dass relevante medizinische Tätigkeiten durch sehr einfache Analysenmethoden der natürlichen Sprache, wie es beim CPGPro der Fall ist, extrahiert werden können. Dieses Phänomen kann durch die Tatsache erklärt werden, dass Prozesse in diesen Dokumenten normalerweise durch eine geringe Anzahl von Formen mit allgemeinen Attributen ausgedrückt werden. Das Extrahieren von Relationen zwischen Prozessen benötigt aber fortgeschrittene Analysemethoden der natürlichen Sprache, wie syntaktische Analyse oder Coreference Analyse.

Acknowledgement

First of all, I would like to express my gratitude to my direct supervisor Mag. Katharina Kaiser whose help, tips and encouragement helped me in all the time of writing of this thesis. Especially, her effort in reading, correcting and providing me with valuable comments on earlier versions of this thesis is of great importance.

I am also deeply indebted to Univ. Prof. Dr. Silvia Miksch for her positive attitude and understanding. I thank her a lot.

Finally, I would like to thank to my family, especially to my parents Emine and Ahmet Akkaya who supported me and made it possible for me to reach these days.

List of Tables

List of Figures

1 Introduction/Motivation

Clinical Practice Guidelines (CPGs) are referred by [Field, 1990] *as*

> *"systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific circumstances."*

Recently, they have gained much interest because they offer many advantages in patient management like defining appropriate care based on the best available scientific evidence, reducing inappropriate variation in practice (standardization) and avoiding additional costs caused by incorrect clinical decisions. CPGs have been applied to many tasks like clinical decision support, workflow management, quality assurance, and resource-requirement estimates [Warren, 1998]. The benefits expected from the appliance of CPGs provoked many medical guidelines to be created by diverse institutes.

Like in many other domains, computerization is seen as a means to facilitate the effective use of CPGs. Therefore, transforming CPGs in a machine-readable and executable format has been the gist of research. Consequently, many guideline representation languages are developed (for a comprehensive overview see [Peleg et. al., 2003]) and systems are designed which convert CPGs in their corresponding models defined in these guideline representation languages [Kaiser, 2005]. A common drawback of these tools is that the process of converting is done manually. Because of the complexity of the underlying representation language, the task can be very cumbersome and time-consuming.

Facing problems motivated us to find a way to extract relevant information automatically. The objective of this thesis is building an Information Extraction (IE) framework (CPGPro), which is based on heuristics to extract relevant information of clinical processes and their relations from semi-structured CPGs to implement it in "Java". The framework works on totally hand-crafted lexical resources and extraction rules, which are based both on syntactical and semantical evidence – for the most part on syntactical – and follow an atomic approach. The ultimate goal of the designed system is filling designed templates which use "XML" as the low-level syntax with high precision and recall values in the test phase.

The output of the designed system can serve as input to other systems, which process the acquired information further for different purposes like helping for formalization of CPGs or classification of documents for Information Retrieval (IR). There are many possibilities to utilize the output of this system with appropriate post-processing.

This thesis consists of two main parts, a theoretical and a practical part. In the theoretical part of the thesis, the technologies used for generating the system and their interaction are described. In the practical part, we will concentrate on the design decisions and heuristics based on the information investigated on the theoretical part.

2 Information Extraction

2.1 Introduction

Today's post-industrial society is marked by an increase in the amount of information technology which led to the "Information Age" where movement of information is faster than physical movement and characterized by the shift from property or political criteria to knowledge as the base of power. In the presence of these facts it is not surprising that there is more text data in electronic form than ever before. It is impossible for a human to process so much information and to use it systematically. Therefore, many information management techniques are explored by researchers. Information Extraction is one of these techniques. In [Riloff, 1999] IE is described as followed.

> *"Information extraction (IE) is a form of natural language processing in which certain types of information must be recognized and extracted from text."*

Especially information end-user industries like finance companies, banks, publishers and governments are interested in IE, because the vast amount of information which they need to process is sometimes infeasible to be processed manually and even if it is feasible, IE may offer great economic benefits when compared to purely manual extraction.

Finding management changes of diverse companies in magazines or determining victims of terrorist attacks reported in news articles are good examples for a typical IE task. For example, a task in the third Message Understanding Conference (MUC) involving terrorist events was to determine the incident type, date, location, perpetrator, physical or human target, effect on targets, and instrument from corpus.

Information Extraction Systems (IE systems) extract specified types of information from natural language text. The task of IE systems can vary from relatively simple problems like Named Entity Recognition to more complex tasks like identifying relationships among entities and events. Ultimately, the system records the extracted information in data structures called templates. It reduces the whole text to a predefined structure which holds only relevant information.

The task of an IE system, which carries out the IE, can be defined in two different forms [Appelt, 1999]:

- A short description of the kind of information being sought
- A database schema or template

Whereby, a template is a tabular output format of extracted information. It constitutes of attribute-slots, which are filled in the process of IE, so that instantiated templates consist of attribute-value pairs. Each template field may be filled by a string extracted from the text or by appropriate elements from a controlled vocabulary.

IE is not Natural Language Understanding (NLU). Its distinct characteristics separate it from NLU and other Natural Language Processing technologies [Appelt, 1999].

- A fixed and limited domain
- A fixed and limited representational format
- Precise metrics of success

IE tasks are defined for a limited domain. Therefore, the objects of interest (i.e., entities) are circumscribed, too. Moreover, an IE task is usually interested in a predefined particular scenario (e.g., joint ventures) in a general predefined domain (e.g., financial news). Thus, the events and relations of interest are restricted, too. IE has precise metrics of success which will be described later. Limited and simple to evaluate character of IE makes it simpler than NLU.

Furthermore, IE tasks are defined to process lots of text, which can show informal, ungrammatical structures. Lots of text means much processing time and ungrammatical structures mean more mistakes by processing. To overcome this obstacle, IE uses simple finite-state methods which can process large amount of texts relatively fast and robust methods which can analyze these texts in a reliable way even in the face of spelling and grammar errors. Because of such design issues, IE is defined as "compromise natural language processing" in [Appelt, 1999]. Another fact, which supports this definition, is its domain-specific character. Indeed, IE systems need a lot of world-knowledge to accomplish their tasks. This world knowledge can be gained manually (e.g., by hand-crafted rules) or by training on a corpus of domain-relevant texts.

Before discussing IE system deeply, some technical terms will be explained and introduced by an example from National Institute of Standards (NIST)[1].

- **Entity** is an object of interest such as a person or organization
- **Attribute** is a property of an entity such as its name, alias, descriptor, or type
- **Fact** is a relationship held between two or more entities
- **Event** is an activity or occurrence of interest such as a terrorist act or an airline crash
- **Named entity** is a named object of interest such as a person, organization, or location
- **Annotation** is a mark-up of a text span in a specific format that indicates a feature or features of the text within the span
- **Evaluation** is the assessment of performance according to agreed upon measures
- **Training** is a process by which a system learns about a dataset

The following text passage and tables [Table 2, Table 3, Table 4, and Table 5] show what an IE system should extract along two example extraction patterns defined by terms of syntactic constituents and basic entities [Table 1]

> *"Fletcher Maddox, former Dean of the UCSD Business School, announced the formation of La Jolla Genomatics together with his two sons. La Jolla Genomatics will release its product Geninfo in June 1999. Geninfo is a turnkey system to assist biotechnology researchers in keeping up with the voluminous literature in all aspects of their field."*
>
> *"Dr. Maddox will be the firm's CEO. His son, Oliver, is the Chief Scientist and holds patents on many of the algorithms used in Geninfo. Oliver's brother, Ambrose, follows more in his father's footsteps and will be the CFO of L.J.G. headquartered in the Maddox family's hometown of La Jolla, CA."*

[1] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/

Sample Extraction Patterns
<subject>(person) <noun,dean> <prep_phrase,of>(organization)
<subject>(organization) <verb, release> <object>(artifact) <adv_phrase>(date)

Table 1 Some Extraction Patterns for the Sample Text Segment

Entities				
Persons	**Organizations**	**Locations**	**Artifacts**	**Dates**
"Fletcher Maddox"	"UCSD Business School"	"La Jolla"	"Geninfo"	"June 1999"
"Dr. Maddox"	"La Jolla Genomatics"	"CA"	"Geninfo"	
"Oliver"	"La Jolla Genomatics"			
"Oliver"	"L.J.G."			
"Ambrose"				
"Maddox"				

Table 2 Entities in the Sample Text Segment

Attributes		
Name	**Descriptor**	**Category**
"Fletcher Maddox" "Maddox"	"former Dean of the UCSD Business School" "his father" "the firm's CEO"	PERSON
"Oliver"	"His son" "Chief Scientist"	PERSON
"Ambrose"	"Oliver's brother" "the CFO of L.J.G."	PERSON
"UCSD Business School"		ORGANIZATION
"La Jolla Genomatics" "L.J.G."		ORGANIZATION
"Geninfo"	"its product"	ARTIFACT
"La Jolla"	"the Maddox family's hometown"	LOCATION
"CA"		LOCATION

Table 3 Attributes for Extracted Entities

Facts	**PERSON**	**Employee_of**	**ORGANIZATION**
	"Fletcher Maddox"	Employee_of	"UCSD Business School"
	"Fletcher Maddox"	Employee_of	"La Jolla Genomatics"
	"Oliver"	Employee_of	"La Jolla Genomatics"
	"Ambrose"	Employee_of	"La Jolla Genomatics"
	ARTIFACT	**Product_of**	**ORGANIZATION**
	"Geninfo"	Product_of	"La Jolla Genomatics"
	LOCATION	**Location_of**	**ORGANIZATION**
	"La Jolla"	Location_of	"La Jolla Genomatics"
	"CA"	Location_of	"La Jolla Genomatics"

Table 4 Facts about Entities

Events			
COMPANY-FORMATION_EVENT		**RELEASE-EVENT**	
COMPANY	"La Jolla Genomatics"	COMPANY	"La Jolla Genomatics"
PRINCIPALS	"Fletcher Maddox" "Oliver" "Ambrose"	PRODUCT	"Geninfo"
DATE		DATE	"June 1999"
CAPITAL		COST	

Table 5 Events of Interest

The most appropriate kind of text for an IE task is the one with factual like mentioned news about terrorist or management domain or technical information like scientific journals or hospital reports, so that the text can be reduced in a structured form with individual facts. Especially, medical domain is interested in such a technology because of the need of analysing reports in natural language. On the other hand, IE systems may not be well suited for texts in which nearly every sentence is relevant.

2.2 History

IE is a new technology not a new idea: as long as 1964 can be found papers with titles like "Text searching with templates" [Wilks, 1987], but these were ideas not backed by any computational power capable of carrying them out."[Wilks, 1997] The earliest effective IE project was "the Linguistic String Project" of Naomi Sager at New York University. The project belonged to the medical domain and aimed to convert patient discharge summaries to a form for subsequent use. He basically concentrated on a computerized representation of English grammar [Sager, 1981].

In the early 1980s, many projects were established. The project of DaSilva and Dwiggins was one of them [DaSilva, 1980]. They built a system to extract satellite-flight information from

reports all around the world. The system used a prolog text grammar. In 1981 Cowie developed a system that extracted canonical structures from field-guide descriptions of plants and animals [Cowie, 1983]. Jong's FRUMP system is another project from early 1980s [DeJong, 1982]. It aimed to extract terrorist events from AP newswires. It was used both for routing and extraction. Another system from this period was built by Zarri [Zarri, 1983]. The system intended to extract relationships and meetings of French historical figures.

Beginning from late 1980s IE research was fostered and shaped by the competitive and objective environment created by Message Understanding Conferences (MUCs) and TIPSTER IE project.

2.2.1 MUC

MUCs were organized by NOSC - The Naval Ocean Systems Center – with the assistance of DARPA – The Defense Advanced Research Projects Agency – which is an agency of the United States Department of Defense and responsible for the development of new technology for use by the military. Because of this, the subject domain of these conferences was defense-oriented like analysing military messages and searching newspapers for terrorist activities to replace human analysts.

Each MUC, except for MUC-1, provided a prepared training-corpus (documents and templates) and a task definition. Each Participant should then adapt its system to the new scenario by using the training corpus. Shortly before the conference, participants got a test-corpus and used their systems to fill provided templates. The results then sent to the MUC organizer, which had created templates with right information (answer-key) manually and evaluated against answer key. MUC evaluations were subsequently represented on the conference, in order to share findings and approaches. Below, each MUC is described in detail [Grishman & Sundheim, 1996].

MUC-1 was organized in 1987. There was neither a predefined output format nor a formal evaluation. Each participant had its own format to represent its results. Therefore, it was not possible to compare single systems. The domain of interest was naval operations. Template formats were developed during MUC-1.

MUC-2 was organized in 1989. Output format was defined as template with 10 slots for attributes – concrete information about events in the text. Using shared output format allowed comparing individual systems. In MUC-2, the domain of interest was also messages about naval operations.

MUC-3 [MUC3] was organized in 1991. The analysed texts were news articles about terrorist activities in Central and South America. This time, template consisted of 18 slots.

MUC-4 [MUC4] was organized in 1992. The analysed texts were again news articles about terrorist activities, but this time with a more complex template, namely 24 slots.

MUC-5 [MUC5] was organized in 1993. Two kinds of texts were processed in MUC-5. These were news articles about

- Microelectronics
- Joint venture

in English and Japanese. In MUC-5, the analysed events were more complicated, so the template too. For the first time, nested templates had been used and had a total of 47 slots. In the previous MUCs, only one template had been used. But this form was not enough flexible to represent events with many participants with their own attributes to be recorded. So one

main template to record information about events which refers to other templates in which information about participants are recorded was more appropriate. Just to show the difficulty of the task, it should be stated that even the task definition was more than 40 pages long.

MUC-6 [MUC6] was organized in 1995. The analysed texts were news articles about management changes. Before MUC-6, new goals and new tasks had defined. These goals were set as a reaction to the trends in previous MUCs. These goals were

- Building task-independent and reusable components for IE systems
- Building more portable IE systems
- Deeper understanding of text

For these purposes, new tasks had been defined. In the previous MUCs, there had been only one task, the "Scenario Template Task" (ST), which consisted of extracting pre-specified event information with involved entities. The participants could subscribe themselves for any subset of these tasks and this time they got less time to adapt their systems. The mentioned new tasks are [Marsh, 1998].

- **Named Entity Task (NE).** Identifying each constituent in the text, which represents a person, an organization, a location name, a date, a currency or percentage figure.
- **Template Element Task. (TE)** Identifying descriptions of entities. For example, Bill Gates-the most richest man in the world.
- **Coreference Task (CO).** Identifying coreferring constituents, thus all mentions of a given entity. Coreference task used identified constituents from NE and TE tasks.

Templates for this conference were called "mini MUCs". They were simple due to portability reasons more like MUC-2 than MUC-5, but they kept the nested design of MUC-5 templates.

MUC-7 [MUC7] was organized in 1997. Analysed texts were news articles about space vehicle and missile launches. MUC-7 included two new tasks, which were

- **Multi-lingual Entity Task (MEN)** NE task for Chinese and Japanese
- **Template Relation Task (TR)** Identifying relational information between entities. For example, employee_of, manufacture_of, and location_of relations

Table 6 presents an overview of MUCs and their defined tasks

	Scenario Template Task	Named Entity Task	Template Element Task	Coreference Task	Template Relation Task	Multilingual Entity Task
MUC-2	X					
MUC-3	X					
MUC-4	X					
MUC-5	X					
MUC-6	X	X	X	X		
MUC-7	X	X	X	X	X	X

Table 6 MUCs and Defined Tasks

MUCs fostered and shaped IE technology. They defined goals and subtasks for IE. Thus, it was possible through evaluations to see which subtasks needed improving and which tasks were satisfied. They helped to determine weak points of existing systems and to highlight differences between different NLP methods and set new trends in IE technology [Cowie 2000].

Data Sets (i.e., training and test corpus, task definitions, answer keys) and automated scoring software, which were prepared for MUCs, have been helping for progress in this science by allowing developers evaluating their systems on the basis of these examples and tasks. Actually, the effort to create these data sets makes the main difference between the systems build in 1980s and 1990s.

At the beginning, MUCs evaluated systems, which processed text corpora from military domain, but in the course of time the conference changed in such a way that increasingly civilian texts were used, because of the huge potential of IE systems in scientific (e.g., medicine) and economic domains.

2.2.2 MUC Metrics

MUC evaluations developed metrics to evaluate participating systems, which acquired a broad acceptance. These metrics are precision and recall [Grishman, 1997]. The formulas for these metrics look as follows

$$\text{Precision (P)} = \frac{Ncorrect}{Nresponse}$$

$$\text{Recall (R)} = \frac{Ncorrect}{Nkey}$$

Thereby,

- **Nkey** is the total number of filled slots in the answer key
- **Nresponse** is the total number of filled slots in the system response
- **Ncorrect** is the total number of correctly filled slots in the system response

Precision provides information about the percentage of correct slots in the response of the system. It can be enhanced by avoiding "false positives", which are filled slots that are incorrect. Whereas, Recall shows how much percent of the right answers (i.e., slots from answer key) found by the system. It can be enhanced by avoiding "false negatives" which are not extracted relevant information.

F-score is another metric used to measure the performance of IE systems. It is a weighted combination of recall and precision. If precision is more important, F-score will be calculated with a parameter value which weights precision, otherwise with a parameter value which weights recall. The formula for F-score looks as follows

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \quad ; \text{P:Precision, R:Recall, } \beta\text{:Weighting}$$

Table 7 presents maxima of achieved scores from MUCs [Sundheim, 1995; Appelt, 1999; Marsch, 1998; Grishman, 2000].

	Scenario Template Task	Named Entity Task	Template Element Task	Coreference Task	Template Relation Task
MUC-3	R<52% P<58% F<46%				
MUC-4	R<59% P<59% F<56%				
MUC-5	R<59% P<60% F<52%				
MUC-6	R<59% P<72% F<57%	R<96% P<97% F<97%	R<77% P<88% F<80%	R<63% P<72% F<65%	
MUC-7	R<50% P<69% F>51%	R<92% P<95% F<94%	R<87% P<87% F<87%	R<79 P<59 F<62	R<67 P<87 F<76
HUMAN F-Score (MUC-7)	85.15%-96.64%	96.95%-97.60%			

Table 7 MUC Evaluations

2.2.3 TIPSTER

"The TIPSTER Text Program[2] was a Defense Advanced Research Projects Agency (DARPA) led government effort to advance the state of the art in text processing technologies through the cooperation of researchers and developers in government, industry, and academia."[TIPSTER]

The actual contribution of TIPSTER TEXT Program to IE was its attention to the creation of a common software architecture for NLP, in order to standardize the technology components and thus provide reusability in multi-component IE systems. CRL's Temple machine translation system [Zajac & Vanni, 1996], Oleada language training system [Ogden & Bernick, 1996] and the Sheffield GATE system[3] are some of the systems, which followed this software architecture [Cowie & Wilks, 2000].

[2] http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/overv.htm

[3] http://gate.ac.uk/

2.3 Approaches

With the lead of MUCs numerous IE system have been designed and implemented. Although they use diverse methods, these approaches can be arranged into two basic groups, one using a Knowledge Engineering approach the other using learning approach or automatic training approach [Appelt, 1999].

In the Knowledge Engineering (KE) approach a professional, who is familiar with the application domain and the function of the designed IE system, is fundamental. She/he is concerned with the definition of rules used to extract the sought-after information. A corpus of domain-relevant texts will be available to the knowledge engineer for this task. Because the general knowledge and intuitions of the knowledge engineer also flow in the process of writing rules, the skills of the knowledge engineer are crucial in this type of systems.

The KE approach is an iterative process. Within each iteration rules are modified as a result of the output of the system on a training corpus. Therefore, the KE approach demands a lot of effort.

On the other hand, in the automatic training approach an annotated corpus of domain-relevant texts is fundamental. Because of this, there is no need for system expertise. There should be only someone who has enough knowledge about the domain and the tasks of the system to annotate the (underlying) corpus of texts appropriately. After the generation of the annotated corpus, a training algorithm is run on it. Then the system can use the knowledge gained from the annotated corpus on new texts from the same domain to extract the desired information.

For some types of tasks like Named Entity Recognition the process of annotation is simple. But in case of complex tasks, annotation of texts could be cumbersome. The difficulty of the annotation process increases with the complexity of the task which should be accomplished by the IE system. At the extreme it can excess the difficulty of manually creating rules in the KE approach.

It is not possible to say that one approach is superior to the other. Both approaches have their strengths and limitations. Before choosing one of them the application domain and resources available should be taken in consideration.

Handcrafted systems tend to achieve higher performances than automatically trained systems. But the iterative character (test-debug cycle) is most of the time laborious. Furthermore, the knowledge engineer needs access to resources like lexicons for the application domain.

Automatically trained systems depend on training data. It is an advantage that annotators usually can be found easily. Moreover, domain portability is less complicated, because usually domain-specific constituents of the IE system are customized faster and without system expertise. However, dependency on training data could cause some problems, if training data is expensive or difficult to obtain.

The mentioned advantages and disadvantages lead to a list of points, which should be considered before designing an IE system [Appelt & Israel, 1999]:

- **Availability of rule writers** is the most important prerequisite for the handcrafted systems. If there is not any skilled knowledge engineer, the automatic training approach should be chosen.
- **Availability of resources** is another important prerequisite for the handcrafted systems. Lexicons and name lists are examples for these required resources. In case of their lack an automatic training approach should be taken.

- **Availability of training data** is the most important prerequisite for automatically trained systems. For simple tasks, like Name Recognition, it is easy to obtain training data. The annotation process will be rapid and the quantity of the annotated text will be satisfactory. For some domains it may be difficult to find enough text and for some tasks it may be slow, difficult, and expensive to annotate texts for the sought-after information. In these cases, where training data is scarce and expensive to obtain, it is not a good idea to use an automatic training approach.
- **The instability of extraction specification** affects both approaches in a different manner. It is common that specifications of a task change with the progress. Some changes may force the re-annotation of the training data and retraining, though they can be handled in the handcrafted systems with a few new extraction rules or omission of them. To be concrete, a task in the knowledge engineering approach with the aim of extracting mountain names can be modified easily to extract river names additionally. In this case, the modification of an automatically trained system will be a cumbersome process. The whole training data must be re-annotated and the system must be retrained from the updated training corpus. However, some changes can be handled in an automatically trained system more easily. For example, for a Name Recognition task the system is designed to process mixed lower and upper case text, but after some time it is decided that the system should process uppercase text only. In this case, automatically trained system needs only the training data to be mapped to uppercase and to be retrained. But extraction rules of a handcrafted system need to be rewritten.
- **Importance of highest possible performance** is another determining factor. MUCs show, that performances of automatically trained systems converge the performances of handcrafted systems. However, the highest possible performance is always achieved by the handcrafted systems.

2.4 The Architecture of Information Extraction Systems

IE system have a modular design. It resembles the "pipe-and-filter style" in the software architecture terminology. Each module in the IE system works as a transducer. It filters and restructures the input text by applying pattern-based rules, which can be created manually or automatically, and adds new features to its input. The output of each module becomes input to the next module and the following module makes use of the new structure and features, which are added by the preceding module. Besides, each module can be constructed independently from other modules after either of the mentioned approaches. The various modules in an IE system work together – in a sequential fashion – to extract the sought-after facts in the analysed text and integrate them in new or larger facts. Ultimately, the templates are created from these facts. Some systems may be interested only in extracting entities and their attributes, but usually users are interested in the facts and events connected with these entities. To extract entities, attributes, facts, and events, patterns are created. A pattern is described as a set of rules, which can be used to generate the linguistic realizations of the facts (or events or attributes or entities). This process is called Pattern Recognition. These patterns cannot be defined as natural word sequences because of the complexity of natural language. Instead, linguistic constituents are abstracted to components and relations. Then patterns are defined in terms of these components and relations.

Figure 1 shows the general structure of an IE system as described in [Appelt & Israel, 1999].

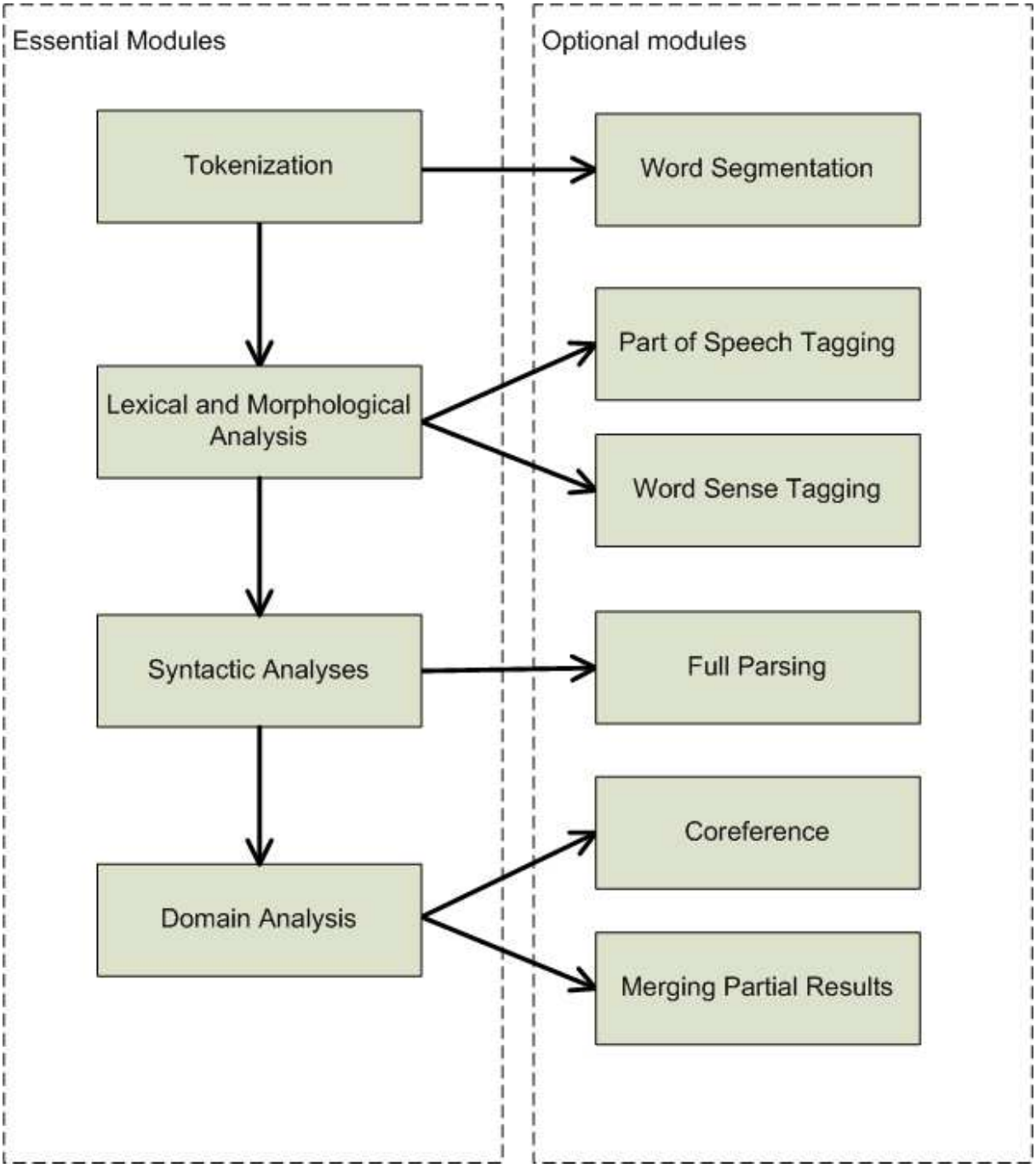

Figure 1 The General Structure of an IE System [Appelt & Israel, 1999]

There are four essential modules, which are implemented by most IE systems. These are modules for tokenization, lexical and morphological analysis, syntactic analysis, and domain analysis. Some application domains may need additional modules. Adding optional modules means extra processing. Because of this, if additional modules do not bring notable performance improvements, they shall not be used.

2.4.1 Tokenization

The Tokenization module is responsible for splitting the input text into sentences and tokens, so that they can be looked up in the lexicon in the subsequent stage of processing – the Lexical Analysis. Tokenization is a trivial task for English and some languages with clear

word borders, which is accomplished with the help of orthographic clues. Some languages like Japanese, which do not fulfil the mentioned requirement, need an amendatory Word Segmentation module [Appelt & Israel, 1999].

2.4.2 Lexical and Morphological Analysis

Morphology is a sub discipline of linguistics and is interested in the structure of word forms. For the morphological point of view words are created from morphemes, which are combined by word formation rules. In English, morphological concepts like inflection, derivation, and compounding do not generate a complexity. Inflectional variants and compound nouns can be listed in the lexicon. Therefore, most IE systems processing English texts do not use a Morphological Analysis module. Unlike weakly inflected languages like English, highly inflected language families and agglutinative languages like German, which tend to create very long words (i.e., compound nouns) with derivational morphemes, are challenging to process. Because of this, processing of such languages require morphological analysis.

In the lexical analysis the tokens, which are determined by the Tokenization module, are looked up in the dictionary to determine their possible parts-of-speech and other lexical features, which are required by subsequent stages of processing. The most important job of lexical analysis is recognizing named (i.e., proper names) and numeric entities. Entities are objects of interest such as persons or organizations. Actually, the sought-after information is events, facts, or properties linked with these entities [Appelt & Israel, 1999].

Named Entities:

- People names
- Company names
- Organization names
- Acronyms
- Product names
- Location names
- etc.

Numeric Entities:

- Dates
- Times
- Phone numbers
- etc.

Proper names and numeric entities can be recognized by a set of rules. As mentioned before, there are two approaches: (1) hand-crafted rules and (2) automatically trained rules derived from an annotated corpus. The rules make use of parts-of-speech, internal structure, orthographic features, and name lists. In Table **8** there are some elementary examples for such handcrafted rules [Callan 2004].

Company names	A sequence of words beginning with uppercase letters and ending with the word "Inc."	<Word>+ <"Inc.">
People names	A sequence of words beginning with uppercase letters which are listed in the available name list for common person names and possibly beginning with a academic title "Prof."	<"Prof.">? <Name>+
Dates		<Day>"/"<Month>"/"<Year> <Day>"."<Month>"."<Year>

Table 8 Examples of Hand-crafted Rules

In most instances, the existence of words beginning with an upper case letter is an evidence for proper names. In one-case text the lack of the orthographic evidence can cause ambiguities, because of the overlap in proper names and normal nouns.

An exhaustive approach to recognize proper names is not adequate, because it is not possible to enumerate all person or company names in a list. Furthermore, new companies are being established. This means that there will be always new company names and in the same way new product names which do not appear in an exhaustive name list. Another point, which can cause trouble, is the existence of foreign names.

It was mentioned that lexical features of tokens are looked up in a lexicon. Another possibility is using automatic taggers, like a parts-of-speech tagger [Appelt, 1999]. A parts-of-speech tagger can avoid incorrect analysis based on ambiguities, which are caused by rare-word senses in an elaborate lexicon. The cost of using parts-of-speech tagger is the processing time for it and the need of training corpus.

Another possibility for word sense disambiguation is the application of word sense taggers. There are three major considerations, which are used by word sense taggers [Wilks & Stevenson, 1996]:

1. Syntactic context, which is usually determined by the window of words in which a token occurs.
2. Relevance to subject matter.
3. Overlap of word occurrences within the definitions of the senses to be distinguished.

2.4.3 Syntactic Analysis

Syntactic Analysis has the aim to identify syntactic structure of the analysed document. Syntactic relations (syntactic roles) in a text correspond to semantic relations (conceptual roles) between entities. Therefore, better syntactic analysis means simpler and more accurate pattern matching in the phase of domain analysis.

The detail of syntactic analysis has varied in different systems. Some systems tried to build a complete parse tree for each sentence in a text (full-parsing) [Grishman et al., 1991; Montgomery et al., 1991]. Others chose to use partial-parsing techniques. There have been even IE systems which totally skip the phase of syntactic analysis [Dolan et al., 1991]. Since MUC-3 [MUC3] there has been a trend towards partial-parsing techniques.

Today, the systems tend to prefer partial-parsing (shallow parsing) techniques, because systems, which use full-parsing techniques, do not seem to enjoy a significant advantage over systems, which do not, and moreover, full parsing is very expensive of computer time [Appelt, 1999].

The triumph of partial parsing is due to the nature of the IE task, which is only interested in specific types of information in a text, and the difficulties in full parsing. IE systems ignore portions of text, which are not relevant for their task. Therefore, parsing these portions – sentences or parts of sentences – and finding irrelevant grammatical relationships will be a waste of time. The sought-after information can often be identified by searching a single clause or phrase. I will demonstrate an example from [Riloff, 1999]. The sentence below originates from a MUC-3 text.

> *"In an action that is unprecedented in Colombia's history of violence, unidentified persons kidnapped 31 people in the strife-torn bananagrowing region of Uraba, the Antioquia Governor's Office reported today.'*

The relevant information in this sentence is the perpetrators, victims, and locations, so the system shall distinguish that unidentified persons kidnapped 31 people in the region of Uraba. The following scenario pattern from domain analysis

> *X kidnapped Y in Z*

will distinguish the relevant information. The other constituents of the sentence can be ignored. Therefore, full parsing will cause unnecessary overhead.

Moreover, creating complete parse trees is a complicated task. Defining conjunction scopes and modifiers is very difficult. Full-sentence parsers can make things worse, if they prevent by making an incorrect parsing simple predicate-argument structure, which holds the desired semantic relationships from being observed. This phenomenon is accounted in [Grishman, 1997]:

> *"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 decisions about structures in their quest to create a parse spanning the entire sentence. In effect, global constraints may make things worse."*

Partial-parsing systems build partial syntactic structures, which can be built with high confidence and using local information [Grishman, 1997]. Generally, they try to define the predicate-core argument structure. Simple noun groups and verb groups are reliably distinguished by local information. For such constituents, it is possible to write unambiguous finite state grammars. Modifiers and ad positional phrases, which make parsing a cumbersome task, are ignored for all except a set of domain relevant words.

An example from [Appelt, 1999] describes how a MUC-5 joint venture text looks after being analyzed by such a finite state grammar.

> *"[Bridgestone Sports CO.]NG [said]VG [Friday]NG [it]NG [has set up]VG [a joint venture]NG [in]P [Taiwan]NG [with]P [a local concern]NG [and]P [a Japanese trading house]NG [to produce]VG [golf clubs]NG [to be shipped]VG [to]P [Japan]NG"*

Subsequent analysis can help to build larger constituents – attaching right modifiers, conjunctions – with rules based on the properties of the head of the constituents. Usually, these rules contain semantic constraints and because of this they are domain-specific unlike

rules building noun and verb groups. It is common that IE systems combine the noun groups separated with a conjunction in the subsequent analysis. The preceding sentence will look as follows

> *"[Bridgestone Sports CO.]NG [said]VG [Friday]NG [it]NG [has set up]VG [a joint venture]NG [in]P [Taiwan]NG [with]P [a local concern and a Japanese trading house]NG [to produce]VG [golf clubs]NG [to be shipped]VG [to]P [Japan]NG"*

Under these considerations we can say that partial parsing is more robust and faster than full parsing. But it is not so domain-independent like full parsing, because of the use of domain heuristics to get attachment decisions.

2.4.4 Coreference Analysis

Usually, relevant entities are referred throughout the analyzed text in different ways. For the success of the system it is important to know which noun phrases refer to the same entity. Therefore, before starting scenario pattern matching, anaphoric references shall be resolved. For this purpose, an IE system needs a Coreference Analysis module. One system can omit it, but a system with an adequate coreference analysis phase will produce better results due to more accurate pattern matching.

Anaphora resolution is one such problem, which includes resolving [Riloff, 1999].

- Proper names
- Pronouns
- Definite noun phrases

Proper names and their variants (i.e., alias, acronym, abbreviation, etc.) should be distinguished as coreferring. To be concrete, in the sentence

> *"After the police arrested Christopher Unger, Mr. Unger claimed to be innocent."*

"Christopher Unger" and *"Mr. Unger"* refer to the same entity. Although identifying coreferring names is very important, it can be handled by the Name Recognition module, too. The discussed module mainly interests with pronoun-antecedent coreference and definite description coreference.

Pronouns should be associated with their antecedents (i.e., named entities) by this module correctly. After analyzing the same sentence with a pronoun

> *"After police arrested Christopher Unger, he claimed to be innocent."*

"Christopher Unger" and "*he"* should be recognized as coreferring.

For resolving definite noun phrases the system needs world knowledge. For example, in this passage

> *"Last Friday we went to Mont Blanc. The mountain was beautiful."*

The system should know that "*Mont Blanc"* is a mountain to resolve the noun phrase "the mountain". The world knowledge can be obtained from domain-dependent is-a hierarchies or more general ontological resources like WordNet[4] and CYC[5,6] . One cannot expect the system

[4] http://wordnet.princeton.edu/

[5] http://www.cyc.com/

to know every definite noun phrase's sort. Because of this dealing with domain relevant words is more feasible.

Like every other module in an IE system, the module responsible for coreference analysis can be designed by a KE or Machine Learning (ML) approach. A general KE approach, which is implemented in FASTUS [Hobbs et al., 1996] is described in [Appelt, 1999] as follows (1) marking each noun phrase with available sortal information (company, lake), number (singular, plural), gender, and syntactic features (name, pronoun, definite/indefinite), other grammatical features, and relations (2) for each of them determine accessible antecedents, filter with semantic/sortal consistency check, and order with dynamic syntactic preferences. The scope chosen determines the accessible antecedents. Names have usually a large scope. It can consist the whole text. In contrast, definite noun phrases have a narrower scope and for pronouns the scope is even smaller. By determining antecedents for pronouns it is feasible to make use of paragraph restrictions. Usually pronoun references are valid in the same paragraph as the pronoun itself, but for definite pronouns seem to be independent from the paragraph structure.

Filtering takes number, gender, and sortal consistency into account. In the sentence

"Christopher eats their apple."

"Christopher" is first a candidate antecedent for pronoun *"their"*, but after number filtering it is eliminated. In the preceding example, *"Mont Blanc"* and *"the mountain"* will pass the filtering, because of the sortal consistency.

After filtering the candidate antecedents, the module must find out which one is most likely the antecedent. The best method to choose the right candidate to be the antecedent is using relative locations in the text. First, the candidates in the same sentence as the referring phrase are inspected from left to right, because it is more common that the subject of a sentence is referred in the same sentence and in English the subject is often at the most leftward position. If there is no candidate in the same sentence the immediately preceding sentence is searched again in the left-to-right order. If no candidate is found again, the preceding sentences are searched but this time in the right-to-left order.

In the automatic learning approach a corpus is annotated with coreference pairs and the system is trained using this annotated corpus. Both probabilistic (e.g., Hidden Markov Models) and non-probabilistic (e.g., decision trees) methods [McCarthy & Lehnert, 1995] are applicable as learning technique.

2.4.5 Domain Analysis

Some IE systems are only interested in simple tasks like Named Entity Recognition, but most of the time users want to obtain facts and events concerned with entities. Therefore, domain analysis is the core of most IE systems. The preceding analyses prepare the text for the domain analysis by adding semantic and syntactic features to it.

Domain analysis aims filling templates, which are in general constructed as attribute-value pairs accurately. As a result, design of the templates is very important for the success of this phase of processing. They can be filled either by elements of a controlled vocabulary or by extracted text from the processed text.

---

[6] http://www.opencyc.org/

For extraction of facts and events, the system needs domain specific extraction patterns (i.e., extraction rules or case frames). These patterns can be generated manually (by means of KE) or automatically (by means of –ML). The portion of text, which matches a defined linguistic pattern is memorized and the information is extracted by the guidance of the extraction rule from this portion of text to fill the template.

There are two approaches to define extraction rules manually [Appelt, 1999].

- Molecular Approach
- Atomic Approach

The Molecular approach is more common in the IE world. It "involves matching all or most of the arguments to an event in a single pattern"[Appelt & Israel, 1999]. The following pattern

<Person> [kidnap-verb, passive] (Adverbial) by <Organization>

will match the text assumed that named entities *"Carlos Ramon"* and *"FMLN"* are recognized by the preceding modules as a person and as an organization.

"Carlos Ramon, mayor of the small coastal village of Santo Domingo, was kidnapped last Tuesday by suspected guerrillas of the FMLN"

The creation of a template structure follows matching. The following template in Figure 2 is created from the sentence above [Appelt, 1999].

```
INCIDENT-0001
        TYPE: KIDNAPPING
        STATUS: SUSPECTED
        DATE: 12-NOV-86
        PERPETRATOR: <ORG-0001>
        TARGET: <PERSON-0001>

PERSON-0001
        NAME: ”Carlos Ramon”
        TITLE: ”Mayor”

ORG-0001
        NAME:”FMLN”
```

Figure 2 The Corresponding Template [Appelt, 1999]

The Molecular approach is a sort of KE. Therefore, it is an iterative process. It starts with common patterns to aim high precision. But recall value is at first iterations relatively low. Recall will be improved with adding less common but relevant patterns in the following iterations. Whereas, overgenerating due to the new rare patterns will cause an expense in precision value. Consequently rule engineers should make a compromise between high precision and low recall.

In contrast, atomic approach aims first high recall and low precision. Subsequent filtering techniques should help to enhance the low precision value. Improving filters will improve the precision value too. As described in [Appelt, 1999] the atomic approach

"suggests going all the way, and building a domain module that recognizes the arguments to an event (the atoms) and combines them into template structures strictly on the basis of intelligent guesses rather than syntactic relationships."

The tasks, which are appropriate for the use of atomic approach, are characterized in [Appelt, 1999] by the two features

- The types of entities can be determined easily
- The structure of the templates should assure that few possible slots exist for an entity of a given type and only certain types of entities can fill these slots.

The MUC-5 [MUC5] microelectronics domain is an example for such a domain.

2.4.6 Inferencing and Merging

Usually the sought-after information is spread among different sentences. In these cases information should be combined before creating the ultimate templates. For this purpose some IE systems use a Merging module. This module uses an algorithm to decide which templates can be merged.

Some information exists implicitly in the text. If the system wants to make this information explicit, it needs some production system rules like the ones, which are used in expert systems [Grishman, 1997].

2.5 Elements Influencing Design Issues

After discussing all important modules for IE systems, the factors influencing design issues summed up as follows

- Type of the text
- Type of the task
- Language of the text

The type of the text can make things easier or more complicated. A text with good orthographic features is easier to analyze. To be concrete, mixed-case characters simplifies the task of named entity recognition. Furthermore a formal text with correct grammatical constructs is processed easier by Syntactic Analysis module.

The type of task can make some modules unnecessary. For instance, an IE system for Named Entity Recognition does not need a module for syntactic and domain analysis.

As mentioned before, some languages need word segmentation and more complex morphological analysis. Therefore, the processed language is an important factor in design decisions.

In Figure 3, the activities of an IE system with the processing level on which they are carried out are shown [Cowie & Lehnert, 1996].

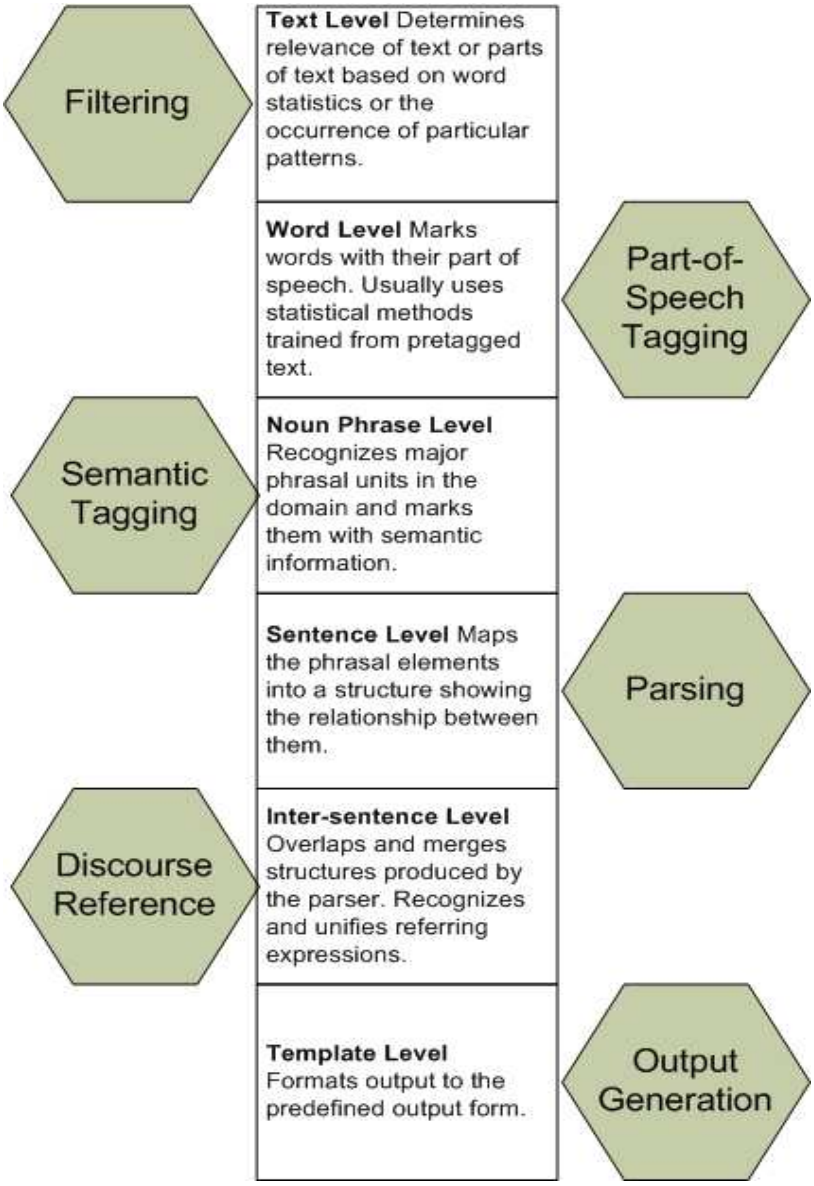

Figure 3 Activities and Processing Levels [Cowie & Lehnert, 1996]

2.6 Template Design

Template design is a major issue in IE research. It influences both the success of the IE system and further processing. Even a perfect IE system without an appropriate output format is not very useful.

Jerry Hobbs and David Israel made great contributions in the area of template design as part of the Data Access for Situation Handling (DASH) research project. They described in [Hobbs & Israel, 1994] template design issue as following:

> *"The problem of template design is a special case of the general problem of knowledge representation. In particular, it is the problem of representing, within a constrained formalism, essential facts about situations in a way that can mediate between texts that describe those situations and a variety of applications that involve reasoning about them."*

Designing a template (i.e., output format) for an IE system requires three different, but interacting considerations [Hobbs & Israel, 1994]:

- The template as an representational device
- The template as generated from input
- The template as input to further processing, by humans or by programs or both

The first consideration is connected with an adequate representation of the analysed domain within a constrained formalism in connection with the task of the IE system. For this purpose, the essential facts should be determined. They can be derived considering requirements of the given task from the semantic model of the domain, which is not necessarily dependent on a concrete syntax of the template.

The second point goes hand in hand with the first one. It states that the output representation of the extracted information should match the typical mode of expression of that information in the text.

The last point covers thoughts about concrete syntax of the template for assuring readability and appropriateness for the given further processing.

The representation of the output of an IE system usually consists of the following kinds of domain elements [Hobbs & Israel, 1994]:

- Basic entities of interest and their significant attributes
- Relations of interest between these entities
- Momentous changes in attributes and relations (Events of interest)

2.6.1 Basic Entities

> *"Basic Entities should be things that endure throughout the temporal focus of the task."* [Hobbs & Israel, 1994]

Thereby, temporal focus is determined by an analysis of the kinds of changes, which are of interest. Properties and relations, which may change within this temporal scope, are thought to be transient. Whereas, properties and relations, which stay unchanged throughout this period, are thought to be permanent. Temporal focus depends on the underlying task. As we will see later, the concept of temporal scope is very important for template design. Figure 4 describes the "temporal scope" [Hobbs & Israel, 1994].

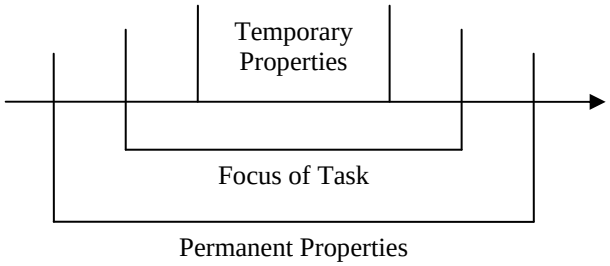

Figure 4 Temporal Scope [Hobbs & Israel, 1994]

Basic entities can be represented as atomic elements or structured objects. For instance a person entity can be represented as an atomic entity if it is not important to record its properties other than its name, but if we need additional information like its age or sex, we

should use a structured object with its own slots for attributes to represent that person. By assigning an attribute to an entity, it is important to be sure, that the attribute really belongs to that entity rather than to another related entity like a related relationship object. The representation as a structured object implicates necessity of using pointers to refer to the structured object. Therefore, readability suffers, because we should follow pointers to access the information about the entity. Because of this it is better to use an atomic representation for an entity, if there is no need to define more than one attribute for it.

As structured object	**As atomic element**
<TEMPLATE>:= ……. PERSON: "Christopher Unger" …...	<TEMPLATE>:= PERSON: <PERSON-1> <PERSON-1>:= NAME:"Christopher Unger" AGE:22 GENDER:MALE

Figure 5 Entity Representations

2.6.2 Relations

Relations can be represented as an attribute of one of the entities in this relationship or as a separate relationship-object.

As an attribute	**As a separate object**
<PERSON-1>:= NAME:"Joseph Schumacher" AGE:45 Father_Of: <PERSON-2> <PERSON-2>:= NAME:"Michael Schumacher" AGE:22 FATHER_OF:	<Father_Of>:= FATHER:<PERSON-2> CHILD:<PERSON-1> <PERSON-1>:= NAME:"Joseph Schumacher" AGE:45 <PERSON-2>:= NAME:"Michael Schumacher" AGE:22

Figure 6 Relation Representations

There are some considerations, which can help to decide which variant to use [Hobbs & Israel, 1994]:

- If the relation is of primary interest, it is better to model it as a separate object.
- If the relation has many attributes for recording some features of itself, it is better to represent it as a separate object, but if it needs only two arguments for its participants it is probably better to model it as an attribute
- If the relation is permanent in the mentioned sense, it is better to model it as an attribute.
- If a relation depends on another relation for its existence, it is better to model the dependant relation as an attribute of the other one.

2.6.3 Events

An event is an activity or occurrence of interest. Several entities can participate in an event [Hobbs & Israel, 1994] classifies events in three groups.

1. Basic events
2. Purposive events
3. Communication events

Basic events provide information only about participant entities and the kind of event. For instance, "Larry Hughes retired as executive vice president of Dona Inc." is a basic event.

Purposive events are like basic events, but with an additional purpose or aim definition. For example, "Larry Hughes retired as executive vice president of Dona Inc., in order to establish his own company." is a purposive event.

We speak from communication events if there is a communicative content in the activity of interest, which is itself an event of one of the three kinds. Figure 7 shows a typical event structure [Hobbs & Israel, 1994].

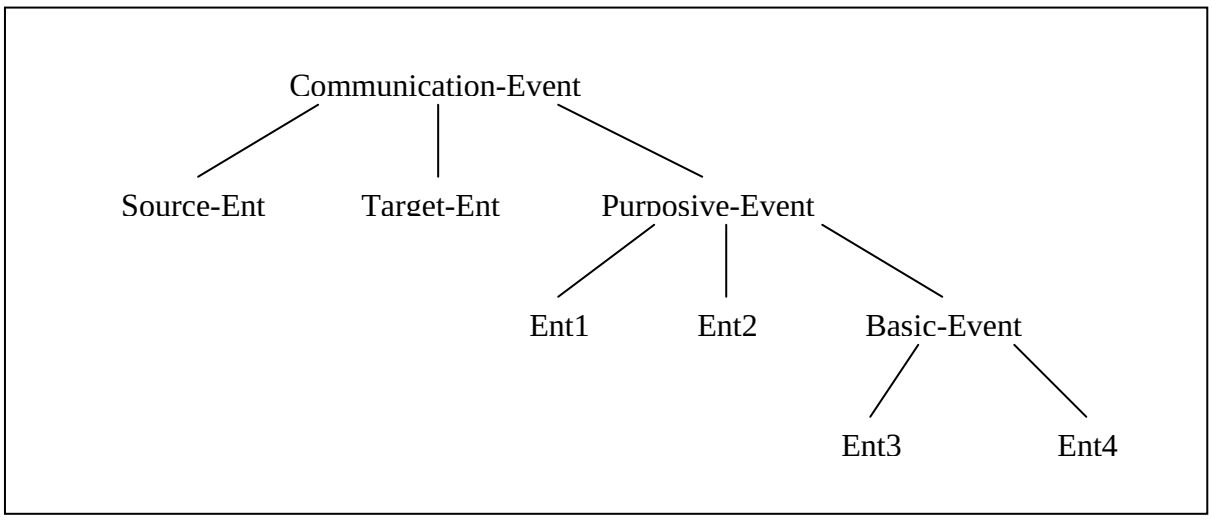

Figure 7 Typical Event Structure [Hobbs & Israel, 1994]

Relations do not only exist between entities but between events. Subevents and causality relations are just two examples.

2.6.4 Slot filling

[Hobbs & Israel, 1994] recommend to keep the slot values as simple as possible, so there are four basic alternative ways to fill these slots

- Simple data types, like strings, number
- Pointer to other objects
- Tuples of elements which can be both a simple data type or a pointer
- Sets of elements which can take any of the three mentioned types

Figure 8 shows a short part of a real MUC template for microelectronics task, which is described by a formal grammar BNF (Backus-Naur Form) [Cowie & Wilks, 2000].

```
<MICROELECTRONICS_CAPABILITY> :=
PROCESS:  (<LAYERING> | <LITHOGRAPHY> | <ETCHING> | <PACKAGING>) +
DEVELOPER:  <ENTITY> *
MANUFACTURER:  <ENTITY> *
DISTRIBUTOR:  <ENTITY> *
PURCHASER_OR_USER:  <ENTITY> *
COMMENT:  ' '

<ENTITY> :=
NAME: [ENTITY NAME]
LOCATION: [LOCATION] *
NATIONALITY: [LOCATION_COUNTRY_ONLY] *
TYPE: {COMPANY, PERSON, GOVERNMENT,OTHER}
COMMENT: ' '

<PACKAGING> :=
TYPE: {{PACK_TYPE}} ^
PITCH: [NUMBER]
PITCH UNITS: {MIL, IN, MM}
PACKAGE_MATERIAL:{CERAMIC, PLASTIC, EPOXY,  GLASS,
CERAMIC_GLASS,  OTHER} *
P_L_COUNT: [NUMBER] *
UNITS_PER_PACKAGE:[NUMBER] *
```

BONDING: {{BOND_TYPES}} *

DEVICE: <DEVICE> *

EQUIPMENT: <EQUIPMENT> *

COMMENT: ' '

Figure 8 MUC Template for Microelectronics Task [Cowie & Wilks, 2000]

3 Applications

In this chapter we will exhibit some IE Systems, which achieved good results in MUCs and contributed to the development of this field with different approaches.

One of these systems, PROTEUS[7], will be explained in detail to simplify the understanding of the mentioned steps in IE. PROTEUS resembles closely the discussed general architecture of IE systems and serves as an introduction to my system, CPGPro, which uses knowledge engineering approach like PROTEUS and has a similar architecture and features.

3.1 FASTUS

FASTUS is an acronym for Finite State Automata-based Text Understanding System. It works essentially as a cascaded, nondeterministic finite-state automaton. There are five levels of processing. Each level serves as an input to the next level, so that larger segments of text are analyzed and structured. The mentioned levels are [Hobbs et al., 1996]:

- Level of complex words: recognition of multiwords and proper names.
- Level of basic phrases: recognition of noun groups, verb groups, and some other particles.
- Level of complex phrases: recognition of complex noun groups and complex verb groups.
- Level of domain events: recognition of patterns for events of interest to build event structures.
- Level of merging structures: merging of event structures arising from different parts of the text if they refer to the same event.

Decomposition of language processing avoids unnecessary domain-independent syntax processing, so that domain-dependent semantic and pragmatic processing in the higher levels can be applied to the right scale structures. MUCs have shown that FASTUS is an effective system and very fast due to the finite-state approach.

3.2 WHISK

WHISK is a learning system that generates extraction rules for a wide variety of documents ranging from formatted to free text. In contrast to CRYSTAL, WHISK applies a supervised algorithm along with a top bottom approach [Soderland 1999].

The WHISK extraction patterns 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).

[7] http://nlp.cs.nyu.edu/index.shtml

3.3 UMass (University of Massachusetts) System

UMass System is based on portable, trainable language processing components to eliminate the knowledge engineering bottleneck. The most interesting components in UMass system are [Fisher et al., 1995]:

- MARMOT is the text bracketing module. It is responsible for part-of-speech tagging and splitting the text into annotated noun phrases, prepositional phrases, and verb phrases.
- BADGER is the extraction module. It instantiates case frames based on a concept node dictionary.
- CRYSTAL is the induction module. It generates concept node dictionary from annotated training texts for the use of BADGER.
- WRAP-UP is the discourse analyzer module. It establishes relational links between entities based on decision tree algorithms.
- RESOLVE is the coreference analyzer module. It handles merging decisions by using nonprobabilistic decision trees, which are trained with annotated training texts.

BADGER is domain and task independent. Importing of BADGER to a new domain or task is needs no adjustment provided that there is an appropriate concept node dictionary for the target domain.

Concept nodes are simply case frames, which are activated by certain linguistic expressions to extract sought-after information from surrounding text. Therefore, obtaining a domain specific concept dictionary is very important, which is handled by CRYSTAL fully automated.

3.4 NYU PROTEUS SYSTEM

Proteus was designed throughout New York Proteus Project, which focuses on the application areas of IE and Machine Translation. Proteus used knowledge engineering techniques and did well in the course of MUC evaluations.

Figure 9 shows the overall architecture of Proteus. The single units will be explained along a simplified example from MUC-6 scenario involving executive succession. The information represented here is originated from publications of Ralph Grishman [Grishman, 1995; Grishman, 1999; Yangarber & Grishman, 1997; Yangarber & Grishman, 1998]. The following tables consist of output of diverse Proteus processing stages for the following sentence.

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

In Proteus, most of the text analysis is performed by matching the text against a set of regular expressions, which trigger associated actions. The matched parts are labelled and possibly get some features. Furthermore, two semantic structures which are called entity and event are associated with some of those matched text fragments. These structures are used to create instantiated templates. describes the Proteus architecture

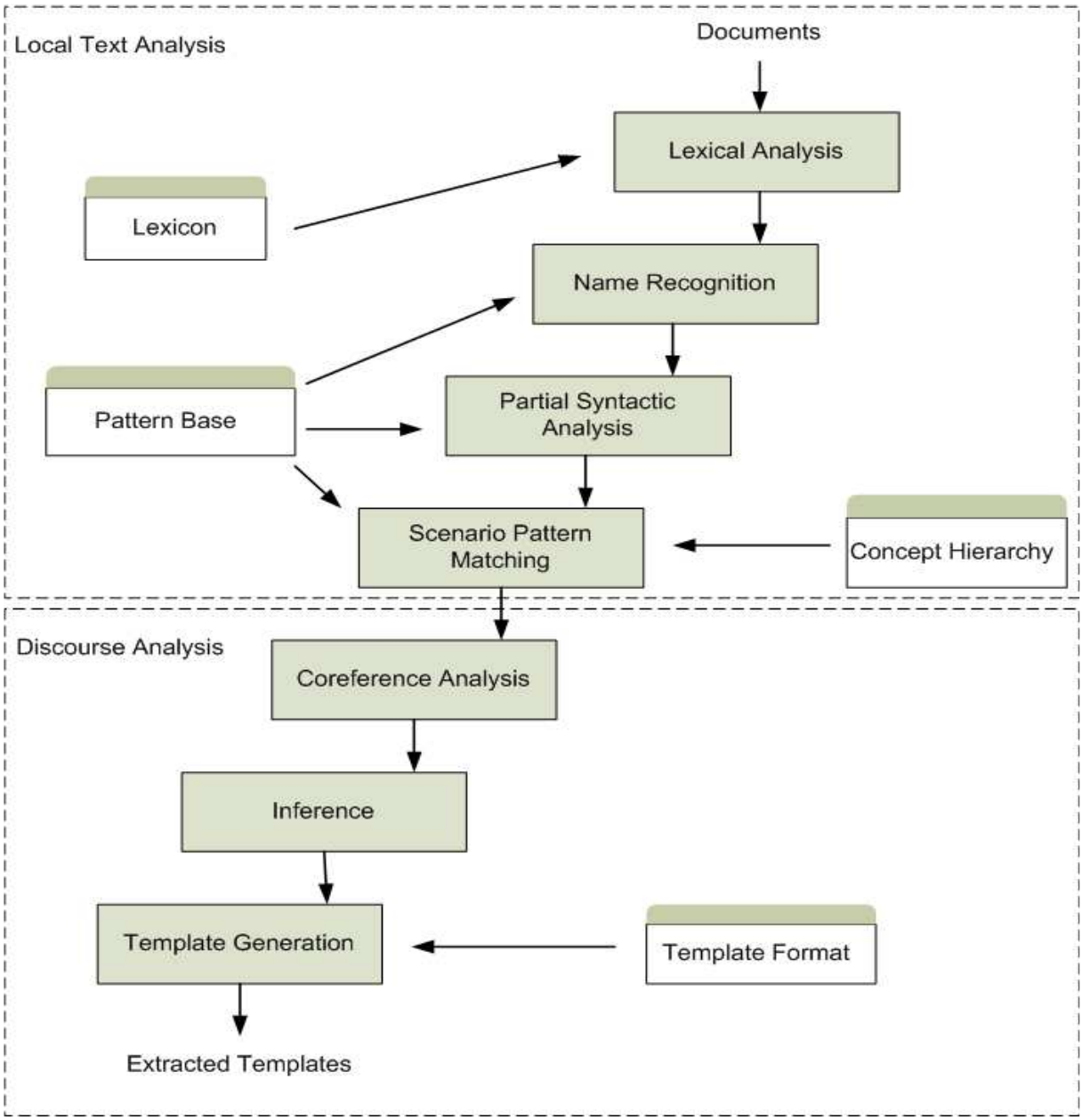

Figure 9 Architecture of NYU Proteus [Grishman, 1999]

3.4.1 Lexical Analysis

This module is responsible for dividing the input text into single sentences and tokens. Each token is looked up in the dictionary to decide its parts-of-speech and other features. The used dictionary – the Comlex Syntax –, which is also developed by New York University, is a broad-coverage dictionary of English. It provides syntactic features, but it does not define proper names. Therefore, other specialized dictionary resources were utilized.

- A small gazetteer, which contains names of all countries and most major cities
- A company dictionary
- A government agency dictionary
- A dictionary of common first names
- A small dictionary of scenario specific words

3.4.2 Name Recognition

This module identifies names (proper names) and numerical entities (dates, currency amount, etc.). It uses a set of patterns (i.e., regular expressions), which are defined in terms of parts-of-speech, syntactic and orthographic features. This stage also records for each name its type (person, company, etc.) and subsequent mentions as aliases for it (some sort of coreference analysis), if possible. For example, in the short part of text

"Larry Hughes goes into the shop…. Mr Hughes…"

"Larry Hughes" should be identified as a proper name with type person and *"Mr Hughes"* as its alias.

3.4.3 Partial Syntactic Analysis

This module recognizes noun (a noun plus its left modifier) and verb groups. Both noun groups and verb groups can be usually identified using just local information. In some cases, it is required to know global dependencies to decide for a noun's left modifier. In these cases, modifiers left unattached.

For each of these matched phrases, features are recorded which are used by patterns in subsequent stages. For example, a verb phrase has information about its tense, voice and root form. A noun phrases has information about its head. Moreover, for each noun phrase, a semantic structure (i.e., an entity) is created.

"[$_{np\ entity:e1}$ Sam Schwartz] [$_{vg}$ retired] as [$_{np\ entity:e2}$ executive 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.]"

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"

Table 9 Entities after the First Stage of Syntactic Analysis

After identifying basic noun and verb groups, Proteus can use additional analysis to attach right modifiers, in order to build larger noun phrase structures. Because of the ambiguity of right modifiers, system needs semantic constraints to decide if it should attach or not. Therefore, rules for attaching right modifiers are domain-specific. For the example above, we can define two such patterns

- <Company description>, <company name>,
- <position> of <company>

In the first pattern, <company description> represents a noun phrase of type company whose head is a common noun. Further, <company name> represents also a noun phrase of type

company, but with a head of type name. In the second pattern, <position> represents any noun phrase of type position and <company> represents any noun phrase of type company. Besides Proteus has a concept hierarchy which regarded by the process of pattern matching. In our example manufacturer will be matched as a company because of this concept hierarchy.

"[np entity:e1 Sam Schwartz] [vg retired] as [np entity:e2 executive president of the famous hot dog manufacturer Hupplewhite Inc.]. [np entity:e5 He] [vg will be succeeded] by [np entity:e6 Harry Himmelfarb.]"

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"

Table 10 Entities after Complete Syntactic Analysis

3.4.4 Scenario Pattern Matching

It is the last stage of pattern matching. It works on constituents identified in the preceding stages of pattern matching (name recognition, syntactical analysis).

Each recognized clausal pattern in this stage turned into a semantic structure called "event". Such extraction rules are based on syntactic and semantic constraints that help to identify the relevant information within a document. For example, two such patterns for our example will be

- <person> retires as <position> ; leave-job(person,position)
- <person1> is succeeded by <person2> ; succeed(person1,person2)

After matching these patterns to our example text, we get following results

"[clause event:e7 Sam Schwartz retired as executive president of the famous hot dog manufacturer Hupplewhite Inc.]. [clause event:e8 He will be succeeded by Harry Himmelfarb.]"

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

Table 11 Entities and Events after Scenario Pattern Matching

3.4.5 Coreference Analysis

This module is responsible for resolving anaphoric references by pronouns and definite noun phrases. Indefinite noun phrases considered as new information. Conversely, when a definite noun phrase or a pronoun is discovered, the preceding text is searched for antecedents. First, the current sentence is searched from right to left, and then preceding sentences sequentially from left to right. There are some constraints for accepting an antecedent

- The class of the anaphor in the mentioned concept hierarchy should be the same or more general than that of the antecedent
- The anaphor and the antecedent should match in number and gender

In our example, the pronoun *"he"* will be resolved in *"Sam Schwartz"*

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

Table 12 Entities and Events after Coreference Analysis

3.4.6 Inferencing and Event Merging

Sometimes sought-after information is distributed over different sentences, so it should be merged before instantiating templates. Moreover, some information is contained implicitly in the text, so it should be made explicit by production rules. Consider, we need in our example to extract start-job events. For this purpose we need production rules

Leave_job(Xperson,Yjob) & succeed(Zperson,Xperson) → start_job(Zperson,Yjob)

start_job(Xperson,Yjob) & succeed(Xperson,Zperson) → leave_job(Zperson,Yjob)

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

Table 13 Entities and Events after Inferencing and Event Merging

3.5 Portability of IE Systems

The term portability is used in the IE discipline as a feature of an IE system which is easy to adapt to a new scenario or domain. The most significant bottleneck preventing more adaptable systems to be built is domain-dependence in diverse sentence and discourse analyzing stages [Riloff, 1999].

Implementing this domain-specific knowledge (extraction and inference rules, lexicon, etc.) in a handcrafted fashion costs a lot of time and expertise. Therefore, such systems are not portable and the market for them will be limited [Grishman, 2002].

The key to portability is automatic acquisition of domain-specific knowledge. This approach is additionally in some degree a solution to the knowledge-engineering bottleneck by obviating the significant-cost assumption [Cowie & Lehnert, 1996]. There are many automated knowledge acquisition techniques which are applied to diverse stages of a IE pipeline. For example, there are systems, which extract domain-specific patterns automatically or semi-automatically [Huffman, 1995; Riloff, 1996a]. Some others try to apply Machine Learning techniques for discourse analysis [Soderland & Lehnert, 1994; Aone & Bennett, 1995] or lexical acquisition [Cardie, 1993; Hastings & Lytinen, 1994] or even for parts-of speech tagging.

Trainable IE systems vary in their concrete implementation a lot. We can classify them as [Appelt & Israel, 1999]

- Supervised or unsupervised
- Rule-based or stochastic

systems.

Supervised systems rely on a pre-tagged corpus. Conversely, unsupervised systems do not need a pre-tagged corpus. They use computational methods to calculate probabilistic information or context rules.Rule-based systems rely on extraction rules, which we have seen most of the time. On the other hand, stochastic systems use frequency or probability considerations. *AutoSlog* [Riloff, 1996b; Riloff 1999] is a good example to show benefits of a trainable system. It is a supervised and rule-based system that generates conceptual dictionaries for IE automatically. *AutoSlog* needed for the UMass/MUC-3 dictionary, which took approximately 1500 person-hours to be built by hand only 5 person-hours. This shows the flexibility of trainable systems, but as mentioned before this kind of systems needs a large annotated corpus. To solve this problem unsupervised systems like *AutoSlog-TS* [Riloff, 1996b; Riloff 1999] are designed.

4 XML

4.1 Introduction

The Extensible Markup Language (XML) is a text-based markup language for structuring of documents. An XML document usually consists of elements, which are called tags, and their attributes. The concept of tags should be familiar to persons interested in information technologies, especially from HTML documents. Indeed HTML, which serves for the description of hypertext documents, is a markup language and defines tags, which a web-browser interprets, in order to represent the information in the document in its intended layout. But one should confound XML in no case with HTML. Contrary to HTML, XML defines not the layout but the structure and high-level semantics of a document.

XML is not a new concept, but forms a subset of Standard Generalized Markup Language (SGML)[8], which is a standard since 1986. XML is much simpler than SGML, but has 90 % of the functionality of SGML. XML with its complementary specifications, like XSLT, XPath, and Xlink, has been developed by the World Wide Web Consortium[9] (W3C) since 1996. XML is characterized by extensibility, structuring, self-description, layout independency, and feasibility of validation [BIG, 2004a].

4.1.1 Extensibility

XML does restrict tags and their attributes, in contrast to HTML. Tags and attributes can be redefined and designated arbitrarily. That makes XML a meta-language, with whose assistance new markup languages (i.e., XML applications) can be developed. Each XML application is formally described by a schema language such as DTD or XML pattern, which we will mention afterwards. Table 14 lists some XML applications and their application domains [BIG, 2004a].

Domain	**Application**
Health Care	'HL7'
Literature	'Gutenberg'
Travel	'openTravel'
News	'NewsML'
Weather	'OMF'
Mathematics	'MathML'
Vector Graphics	'SVG'
Geo Applications	'ANZMETA'
Mobile Applications	'WML'

[8] http://www.w3.org/MarkUp/SGML/

[9] http://www.w3c.org

EGovernment	'eGovML'
Electronic Commerce	'ebXML'
Bank	'MBA'
Advertisement	'adXML'

Table 14 Application Domains and Examples of XML Applications

4.1.2 Structuring

Tags can be nested arbitrarily to complex structures. At the same time tags can have non-structured content.

4.1.3 Self-description

Tags in the XML document describe structure and semantic of its content. Tags are human-legible. In addition, they are simple for the machine to generate and parse. XML documents are generated and read by both humans and machines more easily than flat files, like tab or line-delimited text. The complexity of described data can be adequately handled with XML.

> *"The more complex your data is, the more important it is to use a hierarchical format like XML rather than a flat format like tab or line-delimited text."* [Harold, 2002]

4.1.4 Layout Independency

XML separates the structure and semantics of content from its layout. The expert should not worry during the creation of documents about its formatting.

4.1.5 Validation

XML documents may define a schema optionally, i.e., a formal description of their vocabulary and their grammar rules (Document type definition (DTD), XML Schema (XSL), etc.) that can be validated against it. Its hierarchical structure, human-legible character, and feasibility to be validated make XML very robust.

4.2 Range of Use

4.2.1 Data Transfer

Data can be exchanged by means of XML as pure notation or additionally by means of common schemata. XML is simply plain text and supports Unicode, so it is a portable format and unencumbered by licenses or restrictions. Moreover it is an international standard and there are many tools for diverse platforms to generate and process it. It is like mentioned self-describing and extensible. All of these are reasons why XML is well suited for data transfer.

4.2.2 Data Storage

When it comes to store data, XML is a good candidate to be the format chosen. The characteristics of XML, which are crucial for data transfer, are also important for data storage.

In addition, its robustness and hierarchical structure, which is suitable for many types of documents, are significant for data storage.

4.2.3 Multi Delivery

The same content can be differently presented on different terminals, because XML does not define the layout of the document.

4.3 Schema languages

Schema languages define permitted elements, appendant attributes, and rules for nesting hierarchy of XML applications. Briefly said, they define the structure of XML documents. The most well-known schema languages are DTD, XML Schema, RELAX NG[10], Schematron[11].

DTD is the oldest schema language. It was standardized along with XML. However, the DTD cannot describe very strictly, how an XML file looks like due to a lack of expressivity and a small set of data types. A further disadvantage is the fact that it uses custom non-XML syntax, inherited from SGML, to describe the schema.(see Figure 10)

XML Schema is a novel technology. It uses XML syntax and has many pre-defined data types. Moreover, it gives users the possibility to define own both simple and complex data types as well as constraints for elements and attributes. Unfortunately, the specification is complex and XML Schema instances (XSDs) are relatively hard to understand.

4.4 Architecture of XML documents

In principle, an XML document consists of elements and attributes. The elements may be nested arbitrarily – provided that they are not overlapped. An element consists of one start-tag < tag name > and one end-tag </tag name >. Empty elements may be noted more briefly like < tag name/>. Each XML document must have only one root element, which contains all other elements. Some elements may exhibit attributes. The attribute of an element is integrated in its starting tag and consists of a keyword-value pair, whereby attribute values are between quotation marks (attribut name = "attribute value"). One speaks of the well-formedness of an XML document, if it conforms to all of XML's native syntax rules – at least one element per document, only one element as root, no overlapping of elements, each tag is closed. In addition to elements and attributes, other constructs also exist in an XML document. (see Figure 11)These are:

Entities are referable and named parts (text, markup, or files of arbitrary formats) of a XML document or DTD. They serve for character replacement and modularity of documents.

A Prolog precedes the XML data and specifies the version of XML being used. It has optional encoding and standalone attributes to define used character set and processibility without a schema.

The Document Type Declaration is an optional part of the prolog. It is used to define constraints on the logical structure and to support the use of predefined storage units. It serves for binding of external DTDs (i.e., the DTD is described in another file) or internal DTDs (i.e., the DTD is defined in the document) or both together.

---

[10] http://relaxng.org/

[11] http://xml.ascc.net/resource/schematron/

Processing instructions are intended to be interpreted by specific applications. They can appear at any position in the document outside other markup. They are used as follows:

(<target-*name parameter* ?>)

Comments in XML have the same function as usual comments in various programming languages. They can appear at any position in the document outside other markup. They are used as follows:

(<!-- comment-text -->)

Namespaces are represented by Uniform Resource Identifiers (URIs). Elements and attributes are bound to namespaces. Thus, a global identification for elements and attributes is ensured. Namespaces make it possible to join XML files without "collisions" occurring when markup intended for both XML applications use the same element type or attribute name.

CDATA sections are used to escape blocks of text containing characters. They are used as follows:

(<![CDATA[arbitrary text]]>)

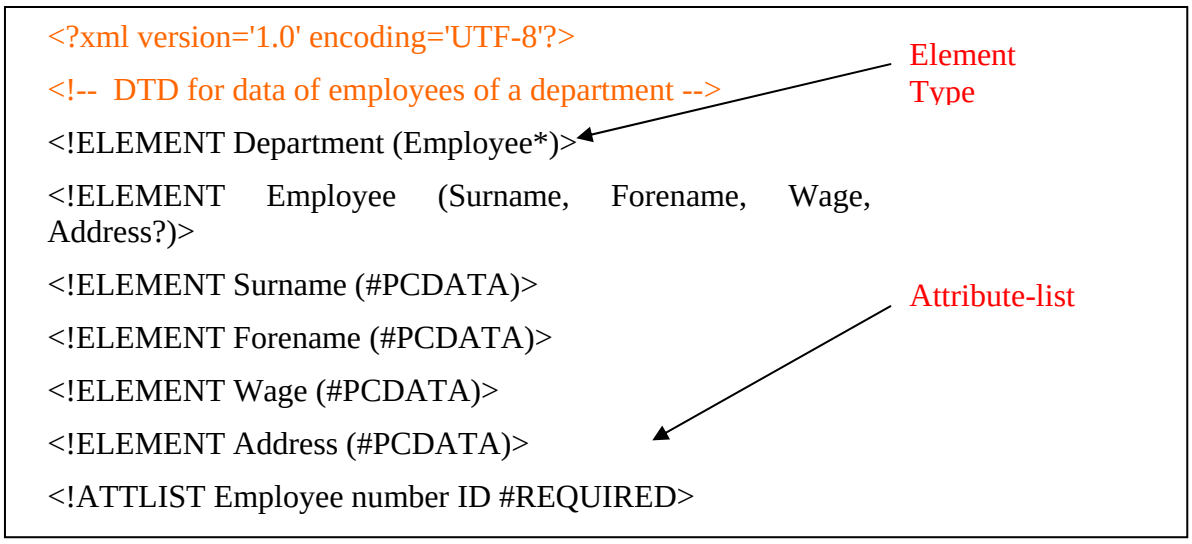

Figure 10 A DTD Document

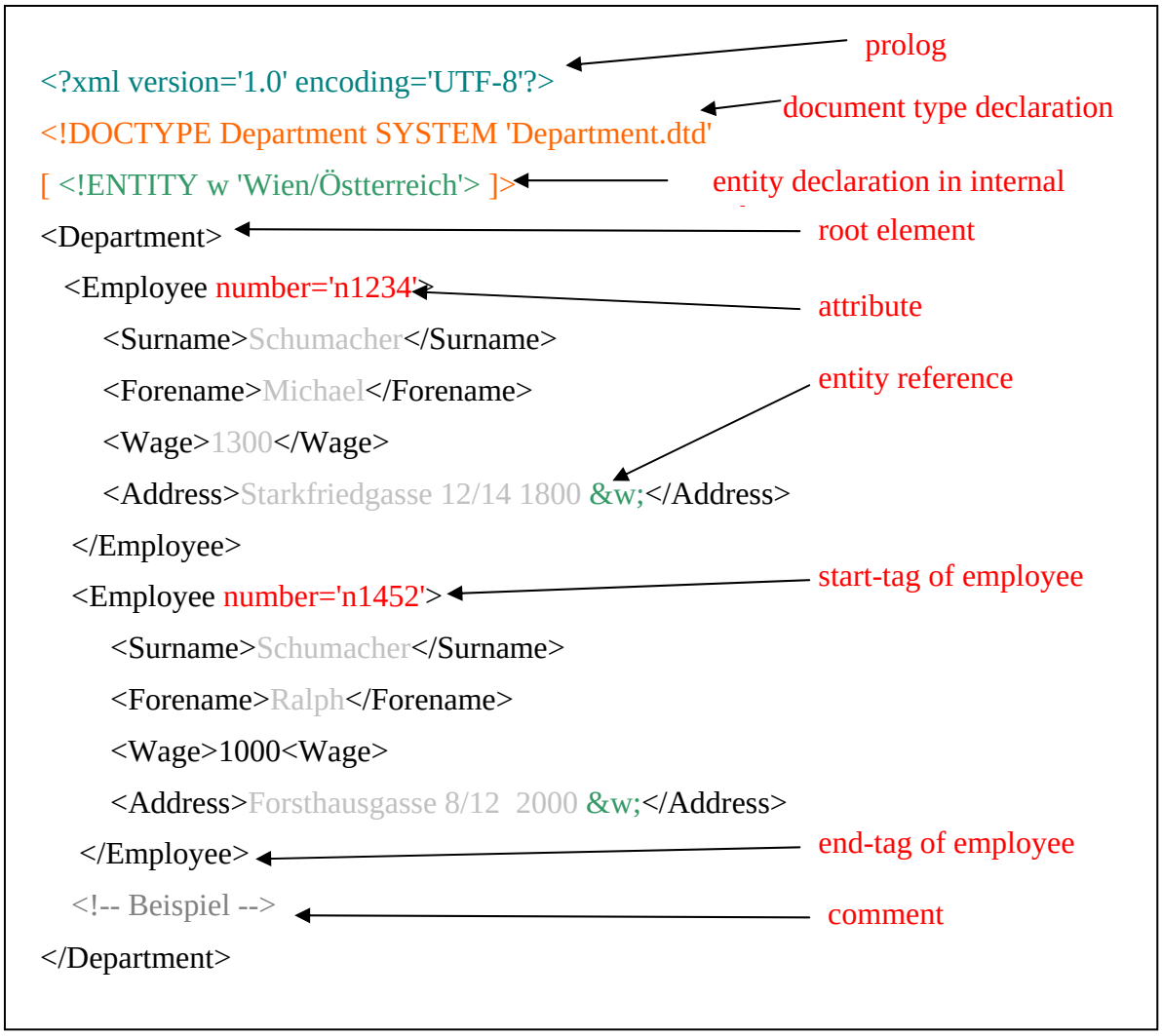

Figure 11 An XML Documet

4.4.1 Designing an XML Data Structure

For designing a schema for an application domain, knowledge and experience about this domain should be collected, in order to decide in the main which elements and attributes are necessary for the schema or an already existing schema should be used. Using an existing schema is much more time sparing. Moreover using standard schemata will make interchange possible [Phillips, 2001].

Most of the time, the existing schema will be more complex than required or not enough strong to satisfy our needs. Even in these cases modifying the existing one is more straightforward than designing a new one from scratch.

Many organizations provide industry-standard schemata. Moreover, there are online repositories for standard schemata. One of them is to find at www.XML.org, which is created by the Organization for the Advancement of Structured Information Standards (OASIS). Another useful online repository is CommerceOne's XML Exchange[12].

[12] www.xmlx.com

XML documents are ultimately processed by software at some point. Therefore, the document structure effects how difficult the processing software will be to develop. The importance of good structure resulted in effort for creating XML design patterns like design patterns in software engineering. XMLPatterns.com[13] is a good online resource to find XML-categorized design patterns and external links to other design patterns.

A complexity factor of designing schemata is to decide, if a concept should be modelled as an element or an attribute. Attributes should be simple and short and they cannot contain substructures. On the other hand elements have exactly the opposite features. Table 15 exhibits the design choices in respect of the mentioned nature of both elements and attributes [Armstrong et al., 2004].

The data contains substructures	The data must be modelled as an element
The data contains multiple lines	It makes sense to model the data as an element
Multiple occurrences are possible	The data must be modelled as an element
The data is a small, simple string that rarely if ever changes	The data can be modelled as an attribute
The data changes frequently	It makes sense to model the data as an element

Table 15 Design Issues

These considerations can help us usually to find the adequate representation for our data, but sometimes it is not clear how to model the data. In these cases a feeling of "XML style" will be helpful. There are a few ways to approach it [Armstrong et al., 2004].

4.4.2 Visibility

The first design heuristic is based on the concept of visibility. If the data should be shown to the end user, it should be modelled as an element. If the end user does not have any use for the data it should be modelled as an attribute. For instance, a document for ordering shoes should have shoe size as an element, but its manufacturer code as an attribute.

4.4.3 Container vs. Contents

Another design heuristic is thinking of an element as a container. The contents of the container are modelled as elements and the characteristics of the container are be modelled as attributes.

> *"Good XML style separates each container's contents from its characteristics in a consistent way."* [Armstrong et al., 2004]

4.5 Java & XML

XML documents are text files. Therefore, one needs a program, in order to manipulate and to do something useful with it. Java has been the language of choice for such programs since the emergence of the XML technology.

[13] www.xmlpatterns.com

The emergence of Java is very similar to it of XML. XML is a result of efforts to reduce complexity of SGML and at the same time to maintain functionality as much as possible. The developers of Java followed the same way. They limited unnecessary complexity con C++. Therefore, both technologies are refinements of accepted concepts.

There has been a close relationship between Java & XML since early days of XML. The major reasons for this phenomenon are concealed in the listed features of Java

- Unicode support
- Portability
- Powerful APIs

As mentioned before XML uses Unicode. Therefore, the realization of completely XML-compliant applications requires that the used language and libraries support Unicode too. At the first stages of XML effort, Java was the only popular language, which used Unicode from the bottom up [Fuchs, 1999]. This helped Java to overcome other languages like Perl and Python, which were traditionally used for text manipulation and did not support Unicode.

Another important factor for closeness of Java & XML is portability. Sun's formulation describes this complementary relationship very well [Ruby, 2002].

> *"Java brings portability to application behaviour, while XML brings portability to data. Together they form a platform for standards-based, distributed computing on the Web."*

Java has many powerful APIs and tools for processing and creating XML documents. Some of them will be introduced below.

4.5.1 XML Parser

An XML Parser (i.e., an XML processor) functions between the XML document and an XML-based application like a broker. It parses an XML document to determine its content and makes its content available for the application over an API. The processing of the document contains different activities, for instance [BIG, 2004b]:

- Checking the document for well-formedness and optionally for validity
- Resolving references on entities
- Assigning attributes types
- Normalizing attribute values

Thus an XML Parser releases the programmer from low-level processing, so the programmer can concentrate on the substantial task. Figure 12 exhibits the context in which a parser works [BIG, 2004b].

There are various XML Parsers for Java, for instance [McLaughlin, 2001],

- Apache Xerces[14]
- IBM XML4J[15]

[14] http://xml.apache.org

[15] http://alphaworks.ibm.com/tech/xml4j

- James Clark's XP[16]
- Oracle XML Parser[17]
- Sun Microsystems Crimson[18]

Each of them has different design and features. Thus, there are some criteria to be considered before choosing a parser [Harold, 2002].

- Validating or non-validating
- Supported APIs
- License
- Correctness
- Efficiency in regard to memory and processor time consumption

Xerces and Crimson are the most popular parsers in the Java world. Both support SAX2, DOM2 and JAXP APIs, which are described below. Crimson is a component of JDK after the version 1.4. With respect to efficiency, there is no difference between the two parsers [Harold, 2002].

XML APIs make it for applications possible to access the parsed document content. There are various standardized XML APIs such as SAX, DOM, JDOM, dom4j, ElectricXML. They offer different access methods on XML documents. The standardization grants the user additional flexibility, because thus a possibility exists to change the used parser without changing the source code. SAX and DOM are mostly used parsers.

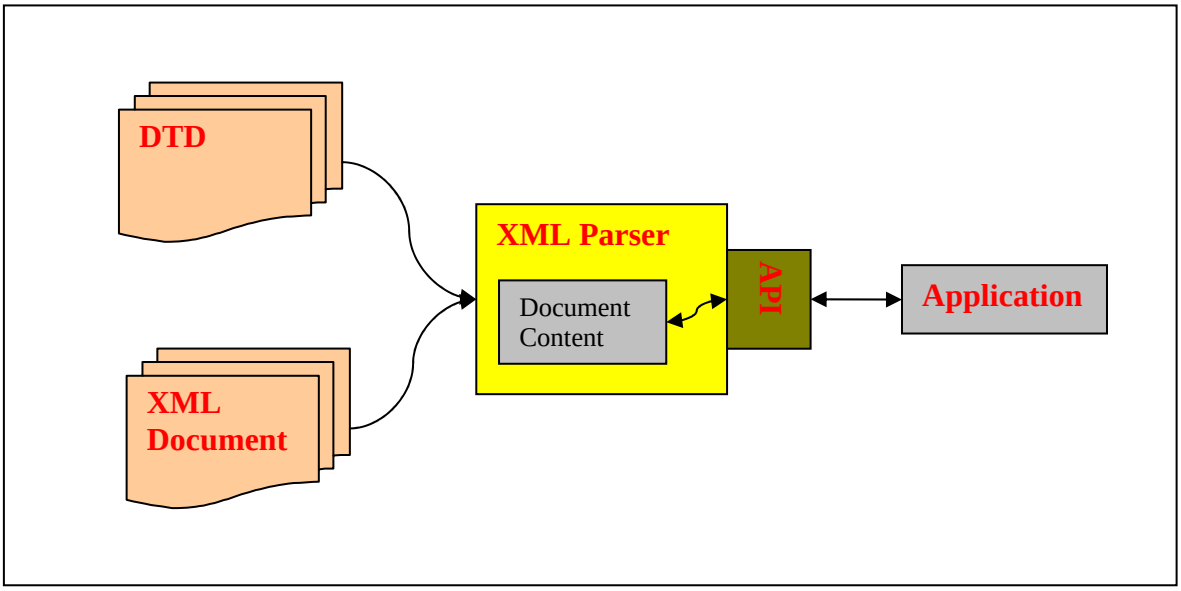

Figure 12: XML Parser [BIG, 2004b]

[16] http://www.jclark.com/xml/xp

[17] http://technet.oracle.com/tech/xml

[18] http://xml.apache.org/crimson

4.5.2 Simple API for XML (SAX)

SAX is an event-based and java native XML API. SAX converts a file stream into an event stream. For example, occurrence of a start-tag or end-tag is such an event. Programs can register themselves for individual events by callback interfaces. The processing style is sequential. An advantage of SAX is that the entire XML file does not have to be in the memory during its processing. That is, however, a disadvantage if there is a dependency in processing between different parts of document, which are scattered over the whole document instead of being sequentially ordered.

When using SAX, it is not possible to look back. Therefore one does not receive any information about the context of an event. If context information of an event is needed, a programmer should provide his own data structures to record the context information. These data structures are filled gradually, while the document is analyzed. The principle is [Harold, 2002]

> *"The complexity of any SAX program is largely a function of the complexity of the data structures you need to build."*

Advantages

- SAX allows an application to process files before they are completely transferred. This characteristic makes SAX suitable for streaming applications.
- SAX needs relatively little memory and processor time. Therefore SAX is suitable for very large documents too.

Disadvantages

- SAX allows a programmer only serial access to the XML document. Therefore data structures should be constructed, in order to access the context of an event.

4.5.3 Document Object Model (DOM)

The Document Object Model (DOM) was standardized by the World Wide Web Consortium (W3C). It is defined in the Interface Definition Language (IDL) of the Object Management Group (OMG). For this reason, DOM is independent of a programming language. For the practical use one needs an implementation of the DOM interfaces. Such implementations are contained with common XML parsers for Java such as Xerces or Crimson.

DOM represents the logical structure of a document in form of a parse tree. It allows reading, as well as dynamic editing of the structure and content of an XML document. Therefore, DOM is called a document-oriented API. The whole document is loaded into main storage for processing. Afterwards the user can traverse the tree using DOM interfaces, access and manipulate nodes, which represent XML constructs such as elements or comments in the XML document.

Advantages

- With DOM one can manipulate XML documents electively. That is, one does not have to process the document necessarily sequentially in contrast to SAX API.
- Programmers are used to the Pull model of DOM, where they traverse the document tree and ask for the content of nodes by themselves. In contrast, the Push model – through call-back interfaces – of SAX needs getting used to it.

Disadvantages

- Because the XML document is loaded completely in the memory as a tree, it is inefficient for its storage use. Consequently there is no possibility for streaming.
- The DOM API is relatively slow.

5 CPGPro

CPGPro is a Java Framework with the intention to extract relevant medical actions and their relations associated with the therapy plan in a clinical guideline for post-processing. "CPG" stands for clinical practice guidelines and "Pro" indicates purpose so that "CPGPro" is resolved to "for clinical practice guidelines".

CPGPro is a knowledge engineering approach based on heuristic methods with a multi-step transformation process. The framework works on totally handcrafted lexical resources, extraction rules and template merging rules, which are based both on syntactical and semantical evidence – for the most part on syntactical. The ultimate goal of the designed system is filling designed templates which use "XML" as the low-level syntax with high precision and recall values in the test phase.

5.1 Application Area

Clinical Guidelines offer many advantages in patient management like defining appropriate care based on the best available scientific evidence, reducing inappropriate variation in practice (standardization) and avoiding additional costs caused by incorrect clinical decisions. Therefore, they have been applied to many tasks like clinical decision support, workflow management, quality assurance, and resource-requirement estimates [Warren, 1998].

As a means of effective use of CPGs, there have been efforts to formalize CPGs for computer-supported authoring and execution. As a result of these efforts, many guideline representation languages have been developed for modelling CPGs in a formal representation (for a comprehensive overview see [Peleg et al., 2005]). Consequently systems are designed which convert CPGs in their corresponding models defined in these guideline representation languages [Kaiser, 2005]. A common drawback of these frameworks is the difficulty of the manual conversion process due to the complexity of the underlying representation language,. In this point, CPGPro can come into operation to support the process of formalization.

The medical actions and their relations, which are extracted by CPGPro, can be utilized by subsequent processes like formalization tools. The application area is not restricted to pre-processing for formalization tools. It can be used with appropriate processing for diverse tasks. For example, the output may be of value for text classification or summarization tools.

5.2 The Task

CPGPro works on XHTML conforming clinical guidelines. The domain chosen is otolaryngology specialty. The task includes determining relevant medical actions with their properties and defining relations between detected actions. Actions are divided in two groups, those which are recommended and those which should be avoided. A relation is defined between two or more of following medical actions [Kaiser et al., 2005]:

- Sequential processes
- Processes without temporal dependencies
- Processes which exclude each other
- Processes containing subprocesses
- Recurring processes

Figure 13 shows the template for this task. It also demonstrates the coverage of the task.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT treatment (actions, relations,dependencies)>
<!ELEMENT actions (action*)>
<!ELEMENT relations (group | selection)*>
<!ELEMENT dependencies (temporalD*)>
<!ELEMENT action (description, duration?, instrument*, dosage?, recurrence?,
condition?, annotation?,context?,cause?,medContext?)>
<!ELEMENT group (reference+,description?,condition?)>
<!ELEMENT selection (reference+,description?,condition?)>
<!ELEMENT condition (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT duration (#PCDATA)>
<!ELEMENT insrument (#PCDATA)>
<!ELEMENT dosage (#PCDATA)>
<!ELEMENT recurrence (#PCDATA)>
<!ELEMENT annotation EMPTY>
<!ELEMENT context (#PCDATA)>
<!ELEMENT cause (#PCDATA)>
<!ELEMENT reference EMPTY>
<!ELEMENT temporalD (offset?)>

<!ATTLIST action
  key ID #REQUIRED
  type (positiv | negative) #REQUIRED
>
<!ATTLIST annotation
IDREFS #REQUIRED
>
<!ATTLIST group
  key ID #REQUIRED
>
<!ATTLIST selection
ID #REQUIRED
>
<!ATTLIST temporalD
IDREFS #REQUIRED
(preceding | succeeding | concurrent) #REQUIRED
>
<!ATTLIST reference
ID #REQUIRED
>
```

Figure 13 Template for Clinical Actions

5.2.1 Design of the Template

The template is designed as an XML document. Therefore, attention is paid to both mentioned design issues for templates of IE systems and for XML-templates. Under these considerations, basic entities, relations and events are identified and modelled.

For the defined task, basic entities are the main objects, which are relevant in a medical action. It is considered that they should correspond to fairly natural and intuitive ways to characterize a medical action and obey analytical demands of the task. Basic entities are outlined as atomic elements, which are filled with linguistic expressions from the analysed text. In the ultimate template, they do not exist as standalone attributes, but always as subelements of actions. On the one hand, this approach causes redundancies by allowing more than one action to declare the same entity, but on the other hand, it increases readability and eases the post-processing.

Relations are restricted to actions. There is not any direct relation between the entities. The relations are divided in two groups. One represents temporal dependencies. The other one represents groupings among them. Both temporal dependencies and relations have unique IDs, but they are implemented in the XML syntax differently, because it is common that relatively more participants take part in a grouping relation, so that readability would suffer if pointers to actions were implemented as attributes in XML template.

Relevant medical actions are the events of interest with unique IDs. They contain associated entities and take part in relations with other actions. Each action is represented in the underlying text at least with one sentence. Figure 13 shows DTD of the template for sought-after information

5.3 Extraction Patterns

Guidelines processed by CPGPro are XHTML-conform. Therefore, they are semi-structured. They consist of a mixture of grammatical and telegraphic text and have additional formatting information (e.g., tags). Extraction patterns developed for analyzed guidelines take into account these features. They are both based on syntactic/semantic constraints and delimiters that bound the text to be extracted. These patterns are defined in three levels. These are:

- Phrase level
- Sentence level
- Discourse level

Extraction patterns defined at each level serve as concept classes in the preceding levels. Phrase level extraction patterns are used for identifying basic entities. Sentence level patterns use phrase level patterns as concept classes to identify medical actions and their linguistic realizations as attributes of these actions. Discourse level patterns are used for action merging.

Patterns described here are linguistic constructs, which frequently occur in the otolaryngology guidelines. Most of them are so general, that they can be probably used in other specialties, too. They have a modular structure and are built up from various levels of constituents by stringing them together. They are obtained in two steps:

- First, several CPGs are examined to find out frequent semantic categories (constituents).
- Then their interaction in CPGs is analysed to find out, how relevant parts of CGPs are built up from these constituents or distinguished by them.

Following three subsections describe each kind of extraction pattern in detail.

5.3.1 Phrase level patterns

Phrase level patterns are syntactic rules, which describe string properties in the lowest syntactic level. They are defined by means of regular expressions. The basic entities defined by these patterns build the attributes of actions. Table 16 shows a list of phrase level patterns with corresponding concept classes.

<context>	\b(for|in)\b[^,.:]+(?<=(patient(s)?|child(ren)?|person(s)?|those|adult(s)?))[^,.:]+
<condition>	(in case|when|if|unless)[^,.:]+
<time>	([\d-]+|(to))+ (hour(s)*|day(s)*|week(s)*|month(s)*|minute(s)*)
<duration>	(for|with)?[a-z]*<time>
<recurrence>	\b(TID|BID|QD)\b \bQ <time> ([\d-]+|(to))+[a-z]*(times|doses) (per|a) day every <time>
<dosage>	([\d-]+|(to))+ mg ((tab(s)?)? ([\d-]+|(to))+ glass(es)? double strength tab
<reason>	because[^,.:]+

Table 16 Phrase Level Patterns

5.3.2 Sentence level patterns

Sentence level patterns are described by means of basic entities. Two major constituents of sentence level patterns are medical terms and triggering words. Members of both of these concept classes are obtained from a lexicon which is also created manually.

CPGPro lexicon holds agent medical terms (e.g. agents, surgical procedures), which are medications usually used in medical actions for otolaryngology specialty. Besides, medical terms are organized in a flat semantic hierarchy, which assign them into groups (e.g. antibiotics, decongestants). Moreover it contains triggering words - mainly verbs - for these medical terms, which indicate the use of an agent in free text and negative triggering words, which indicate avoidance of an agent. A synonym list for some mentioned terms and disorder names are also found in the lexicon (for examples, see section 5.4.).

The other concept classes (e.g., <context>, <condition>) for sentence level patterns are derived from phrase level patterns. In contrast to phrase level patterns, sentence level patterns are delimiter-based, which also use syntactic constraints. Table 17 and Table 19 show sentence level patterns found in seven guidelines from NHC[19], which are used for developing CPGPro. The table of complete guideline names for acronyms (e.g., G1, G2) can be found in the next chapter (see chapter 6)

[19] http://www.guideline.gov/

Name	Pattern	Example linguistic realization
F-Action1	<agent> <trigger>	TMP/SMX can be prescribed
F-Action2	<agent><trigger><duration>	Similar to F-ActionC2
F-ActionC1	<condition> <agent> <trigger>	Similar to F-ActionC2
F-ActionC2	<agent> <trigger> <condition>	An antibiotic that covers resistant bacteria (amoxicillin-clavulanate) should be used to treat patients if amoxicillin fails on 10 to 14 days.
F-ActionCN	<condition><agent><n_trigger>	Similar to F-ActionC2
F-ActionM1	<med_Context> <agent> <trigger>	In PCN-allergic patients, erythromycin ist the drug of choice.
F-ActionM2	<agent> <trigger> <med_Context>	Ampicillin and amoxicillin are often used for treatment of GABS pharyngitis
F-ActionM4	<med_Context> <trigger> <agent> <condition>	Patients with this diagnosis should be treated with erythromycin if they are allergic to penicillin.
F-ActionMN	<med_Context> <n_trigger> <agent>	Similar to F-ActionM2
F-ActionMN2	<agent> <n_trigger> <medContext>	Similar to F-ActionM2
F-ActionN	<agent> <n_trigger>	Particularly ampicillin should be avoided.
F-Relation1	<time_Rel><action><action>	After 10 to 14 days of failure of first line antibiotic (amoxicillin or TMP/SMX), an antibiotic that covers resistant bacteria should be prescribed.
F-Relation1	<n_trigger><agent><med_Context> <reason>	Do not use aspirin with children and teenagers, because it may increase the risk of Reyes Syndrome.

Table 17 Sentence Patterns for Both Grammatical and Telegraphic Text

Name	G1	G2	G3	G4	G5	G6	G7
F-Action1	2	4	-	-	1	2	-
F-Action2	-	-	-	-	1	1	-
F-ActionC1	1	-	-	-	-	-	-
F-ActionC2	2	1	-	-	-	-	1
F-ActionCN1	-	-	-	1	-	-	-
F-ActionM1	2	2	-	-	-	1	-
F-ActionM2	4	1	2	-	1	-	1
F-ActionM3	1	-	1	-	-	-	-
F-ActionMN1	-	-	-	1	-	-	-
F-ActionMN2	-	-	-	-	-	-	1
F-ActionMN3	1	-	-	-	-	-	-
F-ActionN1	-	-	-	1	1	-	-
F-Relation1	-	1	-	1	-	-	-

Table 18 Occurrences of Free-text Patterns in Used CPGs

The patterns in the list above can be applied to free text with grammatical structures in CPGs which are usually organized as paragraphs between and delimiters, but to telegraphic text usually found as list entries or captions, too. These patterns show that, agents (also surgical procedures) combined with trigger words (e.g. use, apply) define relevant sentences for sought-after medical actions provided that they occur in the same clause of the sentence and do not appear in semantic classes <med_Context> or <condition>. This constraint ensures that agents in <med_Context> or <condition> do not interfere with for the action relevant agents and that actions in relations can be separated. Concept classes like <condition>, <med_Context>, <duration> and the others can be combined arbitrarily with <agent> <trigger> pairs to build attributes of the associated medical action. <n_trigger> indicates a negative action (an action which is not recommended). Words like "avoid", "discontinue", "forbid" belong to this category.

Name	Pattern	Example linguistic realization
L-Action1	<agent>	• Cephalexin
L-Action2	<agent><dosage><recurrence> <duration>	• TMP/SMX: one double strength tab BID x 10 to 14 days
L-Action3	<duration><agent><dosage><recurrence>	Similar to L-Action2
L-ActionM1	<med_Context><agent><recurrence><duration>	Similar to L-Action2
L-Action4	<agent><recurrence><duration>	Similar to L-Action2

L-ActionC1	<condition> <agent> <dosage><recurrence>	Similar to L-Action2
L-Action5	<surgical procedure>	Similar to L-Action1
L-ActionC2	<agent> <dosage> <condition>	Similar to L-Action2
L-Action6	<agent><dosage><recurrence>	Similar to L-Action2
L-Action7	<agent><duration>	Similar to L-Action2
L-Measure	<measure> <action>+	**Home Self Care Measures** a. Maintain adequate hydration b. ….

Table 19 Patterns Only for Telegraphic Text

Name	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**
L-Action1	4	1	-	-	1	8	7
L-Action2	-	15	-	-	-	-	-
L-Action3	-	2	-	-	-	-	-
L-ActionM1	-	1	-	-	-	-	-
L-Action4	-	-	-	-	1	-	-
L-ActionC1	-	-	-	1	-	-	-
L-Action5	-	-	-	1	1	2	1
L-ActionC2	-	-	-	-	-	2	-
L-Action6	-	-	-	-	-	1	-
L-Action7	-	-	-	-	-	1	-
L-Measure	3	4	-	-	2	2	-

Table 20 Occurrences of Telegraphic-text Patterns in Used CPGs

The patterns in the list above are only applicable to list entries and captions. In these regions of text, there is usually telegraphic and ungrammatical text. Because of this there is no need to search after trigger words. Patterns show that even only an agent name without a proper trigger word in a list entry indicates relevant clinical actions. The last pattern in the list indicates that list entries are accepted as actions even if they do not include an agent, when some linguistic expressions (e.g. remedy, measure, activity) which are grouped under the concept class <measure> are included in the context -captions under which the list exists – of these list entries.

5.3.3 Discourse level patterns

Discourse level patterns are common ways for merging and grouping clinical actions. In contrast to sentence level patterns, they have also semantic constraints derived from a flat semantic hierarchy. Discourse level patterns will be described later in the following section (see section 5.5.).

5.4 Lexicon

CPGPro lexicon is implemented as a Foreground Lexicon (FL) [Cavaglia, 1999]. A Background Lexicon (BL) is left out. CPGPro lexicon contains only those words that are necessary for the application. These are:

- Medical terms
- Triggers

Medical terms contain medical agents (e.g., "amoxicillin", "cyproheptadine"), surgical procedures (e.g., "tympanostomy", "plastic surgery") and diagnosis terms (e.g., "sore throat", "otitis media"). Obtaining relevant medical terms is completed in two subsequent steps:

- Gathering a core lexicon of medical terms from the development corpus manually.
- Using WordNet to expand the core lexicon

With the application of WordNet, it was possible to find for each manually extracted medical term its synonyms, hyponyms, hypernyms, and coordinate words (words which have the same hypernym), so the core lexicon is expanded and the flat semantic hierarchy needed by CPGPro is obtained.

Triggers are words, which activate a medical term (e.g. "use", "apply"). They are obtained from the underlying corpus with help of a text analysis tool[20] by finding *medical term/trigger* collocations. Their purpose will be explained in the following sections.

5.5 CPGPro Architecture

CPGPro extracts clinical actions from CPGs in five steps. Therefore, it has a modular architecture. Each module is responsible for one step and independent from other modules to some degree – each module works on data that it gets from the preceding module -, so that it was possible to develop and improve each module separately. Figure 14 exhibits the modular architecture of CPGPro.

[20] http://www.niederlandistik.fu-berlin.de/textstat/software-en.html

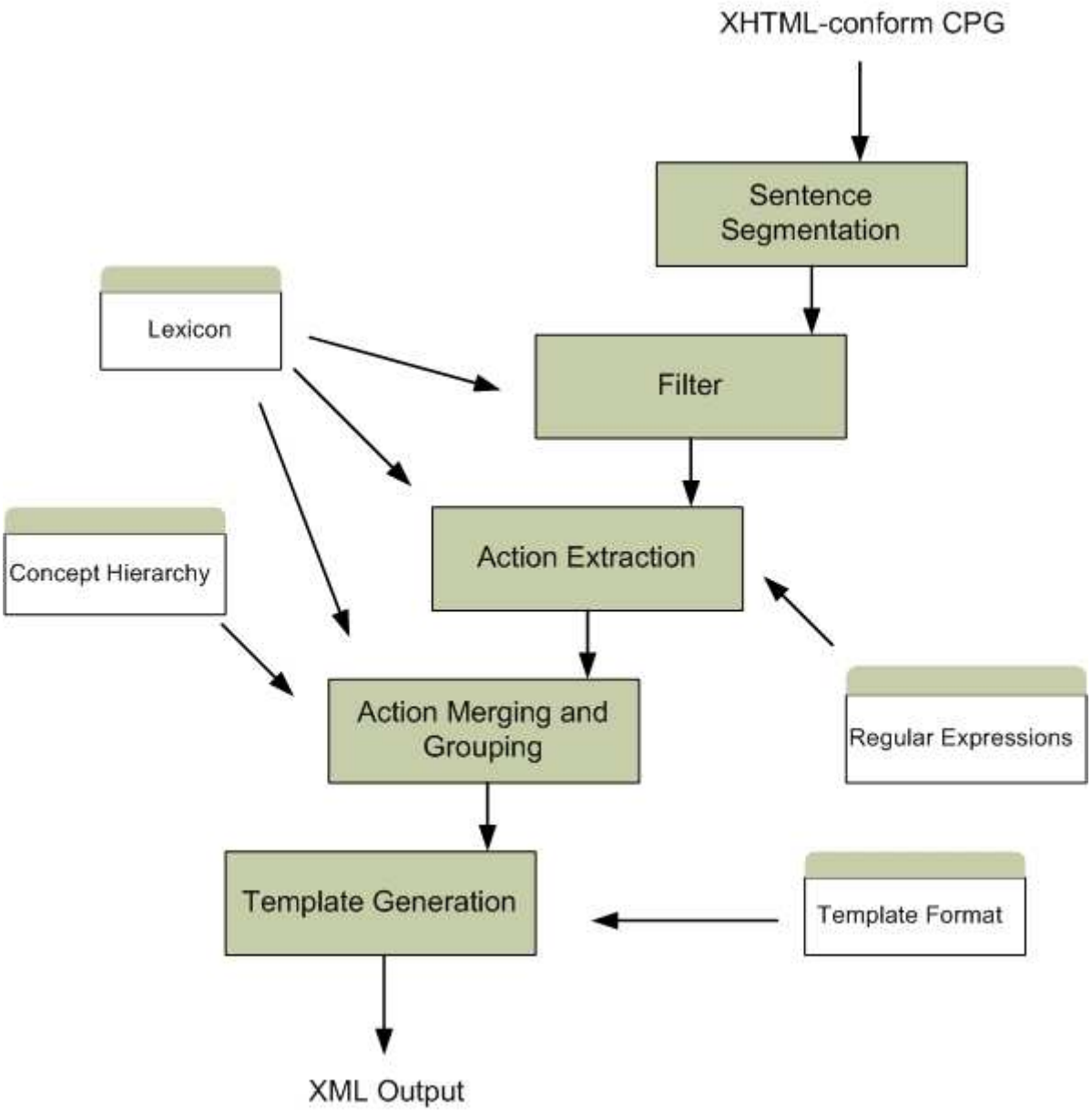

Figure 14 CPGPro Architecture

5.5.1 Sentence Segmentation

First module is responsible for splitting the CPG document in individual sentences. Because of the nature of CPGs, these segments should not always correspond to grammatical correct sentences. For example, they can consist of telegraphic text in list entries. Moreover, this module tags each sentence with additional information. The data recorded with each sentence consists of its delimiter and if it is a complete sentence (grammatical text) or just a phrase group (telegraphic). This information is important, because clinical actions are usually found in telegraphic text and in list entries, which correspond to text between <li> and tags in XHTML documents. Moreover, each sentence is stored in such a way, that it is possible to obtain its relative position in the XHTML tree structure.

5.5.2 Filter

Like in every other IE systems, only small portions of the target document are of interest. The task is limited to finding relevant medical actions and their relations. Therefore, it is needless to process sections associated with diagnosis or symptoms. Moreover, processing these sections could decrease the precision score of CPGPro because of the presence of "false positives".

Filtering occurs in the section level. Sections of the CPG document with captions, which indicate a diagnosis or symptom assertion, are left out. Key words like "history", "sign", "assessment", "factor" are thought to be evidence for such sections provided that they appear in the caption of a section, so all sentences with one of these key words in their context are eliminated.

5.5.3 Action Extraction

This module makes use of both lexicon and regular expressions. Lexicon is used to search in the text for medical terms (e.g., agents, surgical procedures) and their trigger words. Trigger words are needed, because sentences with medical terms and without a trigger word tend to be general information about these terms, in which the task is not interested.

Action Extraction module handles grammatical and telegraphic text differently. Grammatical text, which is usually found in paragraphs (between and delimiters), can only be selected as a medical action for further processing, if it has a medical term with a trigger or negative trigger word in the same clause of the sentence, because of the mentioned reason. In contrast, for a telegraphic it is enough only to have a medical term to be considered as a sentence indicating a clinical action. The reason is that telegraphic text is usually found in lists, which are used to register the recommended medical actions. Moreover, they occur in paragraphs, too. In this context, they indicate the beginning of a list or they give information about the content of the following paragraphs. Therefore, they could contain a medical action and are very important for identifying sought-after information. Negative actions in the list entries are distinguished, too. They are identified with help of negative trigger words in the context or in the proceeding paragraph.

Some sentences in CPGs do not hold these constraints, although they indicate medical actions. These are usually recommendations for self-care. Because they do not consist any medical term, they are hard to find. They have in common, that they have specific key-words (e.g. home, remedies, measures, changes, activities, modifications) in their context, in the preceding paragraph, or list entry.

After identifying sentences, which indicate a medical action, they are processed further to extract some additional attributes, besides medical terms. These attributes correspond to entities, which have internal structure but too many to explicitly enumerate in the lexicon (e.g. dosage, recurrence, duration). Because of this, appropriate regular expressions are used to extract them. After this stage, all actions are identified with all their medical terms and attributes (see Figure 12). Each extracted action is stored also in such a way that it is possible to obtain the relative position of its corresponding sentence in the xhtml tree structure, which will be helpful in the next step for extracting relations between the actions.

5.5.4 Action Merging and Grouping

Action Merging and Grouping works on topographic and semantic features of the extracted actions. There are two needs for merging actions:

- To discard a general action, if a more specific action is found later in the document
- To find annotations for an action later in the document.

General actions are actions with general medical terms (e.g. "antibiotics", "decongestant"). They are usually followed by more specific recommendations. To be concrete, consider there is a sentence which recommends the use of antibiotics with no specific antibiotic agent and it is identified as an action. If this sentence is followed by a sentence with a specific antibiotic agent, it also identified as an action (specific action). In this case, storing the general action is unnecessary, but its attributes. Its attributes are added to the specific action. To solve general-specific medical term relation, CPGPro has a flat semantic hierarchy, which lists all medical terms with their categories.

The second kind of merging is applied, if two actions with same medical terms are encountered. The first action gets the second one as annotation and records attributes of the second one in its attribute fields.

For both merging methods, a merging window is defined. It ensures that actions from different contexts ("treatment"/"further treatment") are not merged. The window is defined so that all actions can search the actions of the one higher level and all lower levels in its section to find specific or annotation actions. Thereby, levels are created with help of <li> taggers. Occurrence of <li> tag presents the start of a new lower level. In contrast, occurrence of tag presents the end of the actual level and switching to the next higher level.

Grouping actions consists of:

- Combining actions in select-one-of relation which models actions excluding each other
- Finding temporal relationships between medical actions

Clinical actions with medical terms from the same category and in the same context exclude each other. To find these actions CPGPro searches in the mentioned window all actions with medical terms from the same category and group them with select-one-of relation.

Temporal relations are really hard to extract. CPGPro detects relations between actions, which are explicitly mentioned within the text. For this purpose, key words (e.g. after, when, until) are used to separate sentences in clauses and actions in different clauses are combined with appropriate relations (i.e., preceding, succeeding, concurrent). Moreover CPGPro uses the context of the actions to derive temporal relations. For example, key word "further treatment" in the context of an action indicates that it is a preceding action to actions in the document before without this key word in their context.

5.5.5 Template Generation

Template generation is the last step in document processing. This module takes all actions and their relations identified by preceding modules and fills the template (see Figure 12) with this information.

6 Evaluation

CPGs used along the implementation of CPGPro can be found on NGC Homepage, which is a comprehensive database for evidence-based CPGs. These text resources are divided in two parts

- CPGs for defining constituents and relations in terms of which patterns are stated and consequently for developing heuristics based on these patterns (training corpus)
- CPGs for testing the accuracy of heuristics by means of CPGPro Framework (testing corpus)

There are a total of 25 CPGs for clinical specialty of Otolaryngology intended for treatment. Seven of these CPGs are used as the training corpus and 14 of them are used as the testing corpus. A CPG for training corpus is not selected arbitrarily, but taking some considerations into account which are listed below. The same considerations are also applicable for creating the testing corpus.

- Owner organization of CPG
- Intended disease of CPG
- Hierarchical structure of CPG

Training corpus is created in such a way that it contains CPGs for different diseases. Thus it was possible to create a set of extraction patterns and an extensive lexicon with a good coverage of the otolaryngology specialty. Moreover, owner organization plays a role by selecting training corpus, because CPGs offered by NGC are created by different organizations and each organization has a widely different style for representing the CPG content. Unfortunately it is not uncommon that an organization uses different formats for different CPGs. Because of this, hierarchical structure of each document is also taken into account, which is very important for the correct operation of heuristics, as it is seen in the preceding chapter. The main idea is that CPGs for both training- and testing corpus should show a lot of varieties in mentioned selection criteria. Table 21 shows a list of CPGs used as the training corpus with their aliases as used in the preceding chapter.

Acute pharyngitis	G1
Acute sinusitis in adults	G2
Reduction of the influenza burden in children	G3
Sore throat and tonsillitis	G4
Diagnosis and treatment of obstructive sleep apnea	G5
Diagnosis and treatment of otitis media in children	G6
Allergic rhinitis	G7

Table 21 CPGs from Training Corpus

Evaluation of CPGPro is carried out in two stages. First the accuracy of finding relevant sentences (sentences, which indicate clinical actions) is tested and then the accuracy of

extracting features out of actions and relations among them is tested. Table 22 and Table 23 show evaluation scores for each of these stages. Thereby, depending on the evaluation stage

- N_{key} is either the number of relevant sentences or the number of extracted attributes and relations in the answer key.
- $N_{response}$ is either the number of sentences or the number of belonging attributes and relations identified by the system as relevant.
- $N_{correct}$ is either the number of extracted sentences or the number of extracted attributes and relations, which agree with the answer key.
- P is the precision value (see section 2.2.2.).
- R is the recall value (see section 2.2.2.).

Title	$N_{correct}$	N_{key}	$N_{response}$	R	P
Evidence based clinical practice guideline for medical management of acute otitis media in children 2 months to 13 years of age"	20	26	27	0.77	0.74
Allergic rhinitis and its impact on asthma	62	77	65	0.81	0.95
Evidence based clinical practice guideline for children with acute bacterial sinusitis in children 1 to 18 years of age	8	14	12	0.57	0.66
Diagnosis and management of acute otitis media	0	3	0	0	-
Otitis media	7	7	7	1	1
Management of obstructive sleep apnoea/ hypopnoea syndrome in adults. A national clinical guideline	8	12	8	0.66	1
Diagnosis and management of childhood otitis media in primary care. A national clinical guideline	7	13	7	0.54	1
Rhinitis	48	56	48	0.85	1
Acute rhinosinusitis in adults	11	17	11	0.65	1
Otitis media with effusion	4	6	4	0.66	1
Evidence based clinical practice guideline for medical management of otitis media in children 2 months to 6 years of age	14	20	14	0.70	1
Symptomatic treatment of radiation-induced xerostomia in head and neck cancer patients	3	3	4	1	0.75
Pneumococcal vaccination for cochlear implant candidates and recipients: updated recommendations of the Advisory Committee on Immunization Practices	4	6	4	0.66	1
Management of sore throat and indications for tonsillectomy. A national clinical guideline	0	20	0	0	-
Overall	196	280	211	0.70	0.92

Table 22 Results from the First Stage

Title	$N_{correct}$	N_{key}	$N_{response}$	R	P
Evidence based clinical practice guideline for medical management of acute otitis media in children 2 months to 13 years of age.	110	132	155	0.83	0.71
Allergic rhinitis and its impact on asthma	271	329	308	0.82	0.88
Evidence based clinical practice guideline for children with acute bacterial sinusitis in children 1 to 18 years of age.	37	55	73	0.67	0.51
Diagnosis and management of acute otitis media.	0	13	0	0	-
Otitis media.	29	33	38	0.88	0.76
Management of obstructive sleep apnoea/hypopnoea syndrome in adults. A national clinical guideline	33	53	35	0.62	0.94
Diagnosis and management of childhood otitis media in primary care. A national clinical guideline.	26	44	31	0.59	0.83
Rhinitis	161	215	200	0.75	0.81
Acute rhinosinusitis in adults	47	59	51	0.79	0.87
Otitis media with effusion.	17	24	21	0.71	0.81
Evidence based clinical practice guideline for medical management of otitis media in children 2 months to 6 years of age.	68	88	87	0.77	0.78
Symptomatic treatment of radiation-induced xerostomia in head and neck cancer patients.	11	13	15	0.85	0.73
Pneumococcal vaccination for cochlear implant candidates and recipients: updated recommendations of the Advisory Committee on Immunization Practices.	7	16	16	0.44	0.44
Management of sore throat and indications for tonsillectomy. A national clinical guideline.	0	60	0	0	-
Overall	817	1134	1030	0.72	0.79

Table 23 Results from the Second Stage

At first sight, the results are a little bit surprising. Because of the used atomic approach, high recall and low precision values were expected, but on the contrariwise CPGPro got higher precision values than recall values. This phenomenon can be explained by constraints defined on the context information of each sentence in the extraction process, but more with not having an exhaustive lexicon. Indeed, test process showed that failure in recognition of relevant sentences is usually justified by not having appropriate medical terms in the underlying lexicon. The results show that CPGPro recognized 70% of all relevant sentences and that 92% of all extracted sentences were actually relevant. Both of the values are promising, especially the precision value. With the supply of a more appropriate lexicon, which covers the otolaryngology domain better, much better results in the recall value will be achieved, probably with a small decrease of the achieved precision value.

A failure in the first stage automatically implies failure in the second stage. Not identified sentences prevent more attributes and relations from being detected and "false positives" cause detection of irrelevant attributes and non-existing relations. This means, a lexicon with a better coverage will increase evaluation values in the second stage, too. Another factor, which complicates the detection of attributes and relations, is the occurrence of coreferences. Because of the absence of a coreference resolution module, CPGPro relies on intelligent guesses to resolve these coreferences. Overall, CPGPro has good scores in the second stage. Both of the evaluation values are satisfactory. It can be said, that CPGPro is a robust and effective system considered the structural and phrasal diversity among CPGs.

7 Conclusion

The object of this thesis was stated in the first chapter as building a framework to support the automation of guideline formalization by means of heuristics. In the light of the demonstrated results from otolaryngology specialty, it can be said that the task is for the most part accomplished.

The evaluation values are satisfactory, but more important underlying heuristics, which are implemented in an atomic approach, allow important performance improvements with the appropriate change of the underlying lexicon. Moreover, the system is fast and robust.

CPGPro heuristics use simple natural language analysis methods. The success of such simple rules is justified by delimiters made use of and the nature of guidelines that actions in these documents are usually expressed in small number of forms with common attributes. Unfortunately, the lack of a coreference module, which would need very complex natural language analysis methods, limits the accuracy of relation extraction.

Besides supporting guideline formalization tools, there are many possibilities to utilize the output of this system with appropriate post-processing. The evaluation shows that CPGPro can be applied to the task of guideline summarization, too. Moreover, it can be applied with appropriate modification to the task of guideline categorization for the sake of Information Retrieval.

8 References

[Aone & Bennett, 1995]: C. Aone and Scott W. Bennett. *Applying Machine Learning to Anaphora Resolution.* In Working Notes of the IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing, pages 151–157, 1995.

[Appelt & Israel, 1999]: D. Appelt, D. Israel: *Introduction to Information Extraction Technology.* IJCAI-99 Tutorial, Stockholm, Sweden, 1999.

[Appelt, 1999]: D. Appelt. *Introduction to Information Extraction.* AI Communications 12(3), 1999.

[Armstrong et al., 2004]: E. Armstrong, J. Ball, S. Bodoff, Debbie B. Carson, I. Evans, D. Green, K. Haase, E. Jendrock. *The J2EE 1.4 Tutorial*. http://java.sun.com /j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf, September 2004.

[BIG, 2004a]: *Business Informatics Group TU Wien: XML Grundlagen.* http:// www.big.tuwien.ac.at/teaching/offer/ss04/we_vo/weM4XMLGrundlagen.pdf, April 2004.

[BIG, 2004b]: *Business Informatics Group TU Wien : XML API-SAX* http://www.big. tuwien.ac.at/teaching/offer/ss04/we_vo/weM5SAX.pdf, April 2004.

[BIG, 2004c]: *Business Informatics Group TU Wien : XML API-DOM* http://www. big.tuwien.ac.at/ teaching/offer/ss04/we_vo/weM6DOM.pdf, April 2004.

[Callan, 2004]: J. Callan. *Information Extraction.* In Human Language Technologies Lectures, Carnegie Mellon University, November 2004.

[Cardie, 1993]: C. Cardie. *A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis*. In Proceedings of the Eleventh National Conference on Artificial Intelligence, pages 798–803. AAAI Press/The MIT Press, 1993.

[Cavaglia, 1999]: G. Cavaglia. *The Development of Lexical Resources for Information Extraction from Text Combining WordNet and Dewey Decimal Classification.* In Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics,1999

[Cowie & Lehnert, 1996]: J. Cowie , W. Lehnert. *Information Extraction.* Communications of the ACM, 39(1),1996.

[Cowie & Wilks, 2000]: J. Cowie, Y. Wilks. *Information Extraction*. Robert Dale, Handbook of Natural Language Processing, Hermann Moisl and Harold Sommers (Eds), Marcel Dekker, 2000.

[Cowie, 1983]: J. Cowie. *J.R. Automatic Analysis of Descriptive Texts.* In ACL Proceedings, Conference on Applied Natural Language Processing, Santa Monica, Calif, ACL, 1983.

[DaSilva, 1980]: G. DaSilva, D. Dwiggins. *Towards a Prolog Text Grammar.* SIGART 72, 1980.

[DeJong, 1982]: G. DeJong. *An Overview of the FRUMP System.* In Strategies for Natural Language Processing, W.G.Lehnert & M.H.Ringle (Eds), Lawrence Erlbaum Associates, 1982,

[Dolan et al., 1991]: Charles P. Dolan, Seth R. Goldman, Thomas V. Cuda, Alan M. Nakamura. *Hughes Trainable Text Skimmer: Description of the TTS System as Used for MUC-3.* In Proceedings of the Third Message Understanding Conference (MUC-3), Morgan Kaufmann, San Mateo, CA, 1991.

[Field, 1990]: M. Field, K. Lohr. *Clinical Practice Guidelines: Directions for a New Program.* National Academy Press, Washington D.C., 1990.

[Fisher et al., 1995]: D. Fisher, S. Soderland, J. McCarthy, F. Feng, W. Lehnert. *Description of the UMass System as Used for MUC-6*. In Proceedings of the Sixth Message Understanding Conf. (MUC-6), Morgan Kaufmann, Columbia, MD, 1995.

[Fuchs, 1999]: Matthew Fuchs. *Why XML is Meant for Java?* Web Techniques Magazine, 1999

[Grishman & Sundheim, 1996]: R. Grishman, B. Sundheim. *Message Understanding Conference 6: A Brief History.* In Proceedings of the 16th International Conference on Computational Linguistics, Springer, 1996.

[Grishman et al., 1991]: R. Grishman, J. Sterling, C. Macleod. *New York University: Description of the Proteus System as Used for MUC-3.* In Proceedings of the Third Message Understanding Conference (MUC-3), Morgan Kaufmann, San Mateo, CA, 1991.

[Grishman, 1995]: R. Grishman. *The NYU System for MUC-6 or Where's the Syntax*. In Proceedings of the Sixth Message Understanding Conference (MUC-6), Morgan Kaufmann, 1995.

[Grishman, 1997]: R. Grishman. *Information Extraction: Techniques and Challenges.* In Information Extraction: A Multidisciplinary Approach to an Emerging Information technology, M.T.Pazienza (Eds), International Summer School (SCIE-97), Frascati, Italy, Springer (Lecture Notes in Artifical Intelligence 1299), 1997.

[Grishman, 2002]: R. Grishman. *Information Extraction.* The Oxford Handbook of Computational Linguistics, Oxford University Press, New York, 2003

[Harold, 2002]: Elliotte R. Harold. *Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX.* Addison-Wesley Professional, 2002

[Hastings & Lytinen, 1994]: P. Hastings and S. Lytinen. *The Ups and Downs of Lexical Acquisition.* In Proceedings of the Twelfth National Conference on Artificial Intelligence, pages 754–759. AAAI Press/The MIT Press, 1994.

[Hobbs & Israel, 1994]: Jerry R. Hobbs, D. Israel. *Principles of template design.* In Proceedings of the Human Language Technology Workshop, Morgan Kaufmann, San Francisco, California, 1994

[Hobbs et al., 1996]: Jerry R. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, Mabry Tyson. *FASTUS: Extracting Information from Natural-Language Texts*. In Finite State Devices for Natural Language Processing, E. Roche and Y. Schabes (Eds.), MIT Press, 1996.

[Huffman, 1995]: Scott B. Huffman. *Learning information extraction patterns from examples.* InWorking Notes of the IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing, pages 127–134, 1995.

[Kaiser et al., 2005]: K. Kaiser, A. Cem, S. Miksch. *Gaining Process Information from Clinical Practice Guidelines Using Information Extraction*.Vienna University of Technology, Institute of Software Technology & Interactive Systems, Technical Report Asgaard-TR-2005-4, April, 2005.

[Kaiser, 2005]: K. Kaiser. *Modeling Computer-Supported Clinical Guidelines and Protocols: A Survey.* Vienna University of Technology, Institute of Software Technology & Interactive Systems, Technical Report Asgaard-TR-2005-2, March, 2005.

[Marsch, 1998]: E. Marsh, D. Perzanowski. *MUC-7 Evaluation of IE Technology: Overview of Results.* 1998. http://www.itl.nist.gov/iaui/894.02/related_projects/ muc /proceedings /muc_7_proceedings/marsch_slides.pdf, March 2005

[McCarthy & Lehnert, 1995]: J. McCarthy, W. Lehnert. *Using decision trees for coreference resolution.* In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pages 1050-1055, 1995

[McLaughlin, 2001]: B. McLaughlin. *Java & XML, 2nd Edition: Solutions to Real-World Problems*. O'Reilly, 2001.

[Miller, 1995]: George A. Miller. *WordNet: A Lexical Database for English.* Communications of the ACM, 38(11), 1995.

[Montgomery et al.,1991]: Christine A. Montgomery., Bonnie G. Stalls, Robert S. Belvin, Robert E. Stumberger. *Language Systems, Inc.: Description of the DBG System as Used for MUC-3.* In Proceedings of the Third Message Understanding Conference (MUC-3), Morgan Kaufmann, San Mateo, CA, 1991.

[MUC3, 1991]: *Proceedings of the Third Message Understanding Conference* (MUC-3). Morgan Kaufmann, San Mateo, CA, 1991.

[MUC4, 1992]: *Proceedings of the Fourth Message Understanding Conference* (MUC-4). Morgan Kaufmann, San Mateo, CA, 1992.

[MUC5, 1993]: *Proceedings of the Fifth Message Understanding Conference* (MUC-5). Morgan Kaufmann, San Francisco, CA, 1993.

[MUC6, 1995]: *Proceedings of the Sixth Message Understanding Conference* (MUC-6). Morgan Kaufmann, San Francisco, CA,1995.

[MUC7, 1997]: Proceedings of the Seventh Message Understanding Conference (MUC-7). National Institute of Standards (NIST), 1997. http://www.itl.nist.gov/iaui/894.02/related_projects/muc/, January 2005

[Ogden & Bernick, 1996]: W. Ogden, P. Bernick. *OLEADA: User-Centered TIPSTER Technology for Language Instruction.* In Proceedings of the Tipster Text Phase II Workshop, 1996

[Peleg et al., 2003]: M. Peleg, S. W. Tu, J. Bury, P. Ciccarese, J. Fox, R. A. Greenes, R. Hall, P. D. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. H. Shortliffe, M. Stefanelli. *Comparing Computer-Interpretable Guideline Models: A Case-Study Approach.* Journal of the American Medical Informatics Association (JAMIA), 10(1), pages 52–68, 2003.

[Phillips, 2001]: Lee A. Phillips. XML-Kompendium: Modernes Daten- und Dokumentmanagement, Markt+Technik, 2001

[Riloff, 1996a]: E. Riloff. *An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains.* Artificial Intelligence, 85, pages 101–134, 1996.

[Riloff, 1996b]: E. Riloff. *Automatically Generating Extraction Patterns from Untagged Text.* In Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press/The MIT Press, pages 1044-1049, 1996

[Riloff, 1999]: E. Riloff. *Information Extraction as a Stepping Stone toward Story Understanding.* MIT press, Montreal, Canada, 1999.

[Ruby, 2002]: D. Ruby, *Which Half is Which*. XML Magazine, March 2002

[Sager, 1981]: N. Sager. *Natural Language Information Processing: A Computer Grammar of English and Its Applications.* Addison-Wesley, Massachusetts, USA, 1981

[Soderland & Lehnert, 1994]: S. Soderland and W. Lehnert. *Wrap-Up: A Trainable Discourse Module for Information Extraction.* Journal of Artificial Intelligence Research (JAIR), 2, pages 131–158, 1994.

[Soderland, 1999]: S. Soderland. *Learning Information Extraction Rules for Semi Structured and Free Text.* Machine Learning: Special Issue on Natural Language Learning, 34, pages 233-272, 1999.

[Sundheim, 1995]: B. Sundheim. *Overview of Results of the MUC-6 Evaluation.* In Proceedings of the Sixth Message Understanding Conference (MUC-6), Columbia, MD, Morgan Kaufmann Publishers, 1995

[TIPSTER]: TIPSTER Text Program Homepage http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/, March 2005

[Warren, 1998]: Todd E. Warren. *Clinical Practice Guidelines.* Adis International, 1998

[Wilks & Stevenson 1996]: Y. Wilks, M. Stevenson. *The Grammar of Sense: Is Word-sense Tagging Much More than Part-of-speech Tagging?* Technical Report CS-96-05, University of Sheffield, UK, 1996

[Wilks, 1987]: W. Yorick. *Text Searching with Templates.* Technical Report ML 162, Cambridge, Language Research Unit, 1987.

[Wilks, 1997]: W. Yorick. *Information Extraction as a Core Language Technology.* In Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, M.T.Pazienza (Eds), International Summer School (SCIE-97), , Springer (Lecture Notes in Artifical Intelligence 1299), 1997.

[Yangarber & Grishman, 1997]: R. Yangarber and R. Grishman. *Customization of Information Extraction Systems.* In Proceedings of International Workshop on Lexically Driven Information Extraction, Frascati, Italy, 1997

[Yangarber & Grishman, 1998]: R. Yangarber, R. Grishman. *NYU: Description of the Proteus/PET System as Used for MUC-7 ST.* In Proceedings of the Seventh Message Understanding Conference (MUC-7), Morgan Kaufmann, 1998.

[Zajac & Vanni, 1996]: R. Zajac, M. Vanni. *The Temple Translator's Workstation Project.* In Proceedings of the Tipster Text Phase II Workshop, 1996

[Zarri, 1983]: G. P. Zarri. *Automatic Representation of the Semantic Relationships Corresponding to a French Surface Expression*. In ACL Proceedings, Conference on Applied Natural Language Processing, Santa Monica, Calif, ACL, 1983.