Linköping Studies in Science and Technology
Thesis No. 1000

Adaptive Semi-structured Information Extraction

A User-Driven Approach to IE

Anders Arpteg

INSTITUTE OF TECHNOLOGY
LINKÖPINGS UNIVERSITET

LiU-Tek-Lic-2002:73

The Computer and Information Science Department
Linköpings university, SE-581 83, Linköping, Sweden
http://www.ida.liu.se/

Linköping January 2003

Printed by:
UniTryck, Linköping, Sweden
ISBN 91-7373-5892-2
ISSN 0280-7971

Distributed by:
Linköpings university
The Computer and Information Science Department
SE-581 83, Sweden

Abstract

The number of domains and tasks where information extraction tools can be used needs to be increased. One way to reach this goal is to construct user-driven information extraction systems where novice users are able to adapt them to new domains and tasks. To accomplish this goal, the systems need to become more intelligent and able to learn to extract information without need of expert skills or time-consuming work from the user.

The type of information extraction system that is in focus for this thesis is semi-structural information extraction. The term semi-structural refers to documents that not only contain natural language text but also additional structural information. The typical application is information extraction from World Wide Web hypertext documents. By making effective use of not only the link structure but also the structural information within each such document, user-driven extraction systems with high performance can be built.

The extraction process contains several steps where different types of techniques are used. Examples of such types of techniques are those that take advantage of structural, pure syntactic, linguistic, and semantic information. The first step that is in focus for this thesis is the navigation step that takes advantage of the structural information. It is only one part of a complete extraction system, but it is an important part. The use of reinforcement learning algorithms for the navigation step can make the adaptation of the system to new tasks and domains more user-driven. The advantage of using reinforcement learning techniques is that the extraction agent can efficiently learn from its own experience without need for intensive user interactions.

An agent-oriented system was designed to evaluate the approach suggested in this thesis. Initial experiments showed that the training of the navigation step and the approach of the system was promising. However, additional components need to be included in the system before it becomes a fully-fledged user-driven system.

Acknowledgments

This work has been supported by The Knowledge Foundation and the University of Kalmar in Sweden. I wish to thank my main supervisor Professor Erik Sandewall for his encouragement and wise guidance throughout this project. I would also like to give my thanks to Christer Lundberg and Professor Wlodek Kulesza.

I would probably not have started my Ph D studies without the encouragement and support given by Christer Lundberg. Professor Kulesza, who is also one of my supervisors, has also been very helpful to me. The general discussions and especially the financial support throughout the difficult economic situation at our department have been appreciated. I would also like to give my thanks to professor Valeri Marenitch for very interesting discussions and wise comments.

Final thanks go to my parents for always being there and supporting me.

Contents

Chapter 1

Introduction

The *information extraction* (IE) concept has been given a number of different definitions such as “the task of semantic matching between user-defined templates and documents written in natural language text”, “a process that takes unseen text as input and produces fixed-format, unambiguous data as output”, and “to extract relevant text fragments and piece them together into a coherent framework” [1, 2, 3]. The preferred definition for this thesis is to find subsets of relevant textual information for a given task or question and organize them into a clearly defined data structure. This is different from the area of *text understanding* that attempts to capture the semantics of whole documents, IE deals only with document subsets relevant for a given task or question.

Examples of applications of IE are shopping agents that locate information about products or services at different retailers and compare them to find the best retailer, event agents that collect information about events that occur at different locations and times, and news agents that collect news articles from different sources and present articles relevant for a specific user. Information stored in free natural text or with a semi-structural format would be too difficult to handle directly without IE for these applications.

The area of *information retrieval* (IR) has attracted a lot of attention recently due to the increased popularity of the World Wide Web. Services such as AltaVista and Google are known by most Internet users and are an essential part of the Web today. The main difference between IR and IE is that IR returns a set of documents rather than a set of answers or phrases related to the query. Thus, the information is not translated to a defined data structure in IR. The advantage of IR is that it is possible to cover a large number of domains, whereas IE typically requires domain-dependent knowledge and is therefore limited in the number of covered domains. These two areas should not be seen as predecessors or successors of each other, rather they can be combined and complement each other to provide more useful services.

The concept *structured text* as used in this paper refers to textual information stored in a clearly defined data model, for example in a relational database. The

advantage of clearly defined structural information is that the information can be automatically analyzed and processed more effectively. All the information has to adhere to the schema that defines the data model. *Semi-structured text* does not have the clear data model representation as structured text, but has more structural information than natural language text, e.g. HTML documents with presentational information combined with the content. The information does not have to adhere to a predefined schema. These semi-structural documents are often less grammatically correct than natural language texts with choppy sentence fragments [4]. The natural language processing (NLP) methods designed for free text do usually not work as well for semi-structural information sources.

A possible solution to this problem could be to simply remove the presentation related information and continue to use NLP techniques. The disadvantage of this solution is that valuable information for the extraction task would be lost. The problem of choppy sentence fragments would also still be present. It has also been shown that the extraction task can be performed with very high accuracy using only the semi-structural information, without the use of any NLP technique [4].

Web documents change rapidly, in both content and structure. To handle the dynamic nature of the Web, it is necessary to have adaptive and easily developed IE systems. A common task of web-based IE systems is to collect information from a web site where the semi-structural information is more or less constant and the content changes frequently. Thus, the IE system benefits from the use of semi-structural information to complete the task. This is not possible with free text information extraction, which lacks the kind of semi-structural information present in Web pages.

The term *wrapper* has been given different definitions depending on the context. In the database community, it represents a software component that converts data from one data model to another. In the Web context, it represents a software component that converts information in a Web page to a structural format, e.g. into a database. The latter corresponds to the preferred definition in this thesis. The term wrapper represents an IE software component that takes semi-structured textual input and generates structured text as output. *Automatic wrapper generation* and *wrapper induction* are terms that refer to the automatic construction of wrappers, for example using machine-learning techniques.

The performance of semi-structural IE systems (e.g. wrappers) is often measured differently than traditional systems. The precision and recall measure is typically very high and therefore not a useful measure of the system. Only systems with 100% precision and recall are of interest for sources with significant amounts of semi-structural information [5]. These systems are evaluated by their *expressiveness* and *efficiency*, which measures the coverage of the wrapper (percentage of sources that have 100% precision and recall) and how easily the wrapper can be adapted to new domains.

The *development* and adaptability of IE systems can be categorized in the three different approaches described below. The knowledge engineering approach is the traditional way to construct IE systems and the user-driven approach is novel and

still mostly a concept [6]. A truly user-driven IE system has the advantage of not requiring a domain or computer expert to adapt to new domains and tasks. Without the need for domain and computer experts, the availability and use of IE systems can be increased significantly.

Knowledge Engineering This approach requires a domain expert who is able to add extraction rules to the system. The difficulty of finding domain experts who also have sufficient computer knowledge makes it difficult to adapt to new domains.

Automatic Training The use of machine learning techniques can relax the requirement of computer knowledge for the domain expert and thus only require domain experts who can annotate documents for the domain. The system can automatically be trained using annotated documents. This makes the job of adapting an IE system to a new domain easier, but it still requires a significant amount of work to construct the training data for the system.

User-Driven IE UDIE differs from automatic training in that novice users rather than domain experts shall be able construct an IE system without need of large training sets. This makes it easy to adapt to new domains, although more intelligence is required for the system and higher requirements of the domain. For example, it may be necessary to have semi-structural documents instead of free natural language texts.

One of the main topics in this thesis deals with the adaptivity of the navigation task for an IE system. The navigation task involves how to navigate through the hypertext parse tree (or subset of that tree) from a given start point to desired extraction points. The navigation can cover multiple linked pages, thus the task defines which pages to download in addition to which nodes in the parse tree to extract.

It would require a lot of work for the user to manually define and maintain the path through the tree. It would be more efficient to require only a starting point and a few examples of what to extract from the user and to then automatically locate the optimal paths between the starting point and the extraction points. The level of adaptivity of the navigation tasks depends on the amount of work and expertise required for new domains and tasks. A high degree of adaptability is required to make the IE system user-driven.

Specialized techniques are necessary to be able to let novice users construct wrappers with few examples, handle complex structures such as nested and linked lists, and handle the dynamics in the source documents. The learning algorithms used in wrapper induction systems are seldom able to handle uncertainty or dynamic properties of the process. Furthermore, they use a character-based view of documents, rather than a tree-view; thus, the patterns that are identified do not take full advantage of the nodes and relationships between them. A tree-view of

the documents and the links between them would possibly increase the use of the existing structural information.

One approach to the problem is to view the corpus as a large tree where each node represents an element in the document[1]. The extraction process can then be viewed as a Markov process where a decision is made for each node as to whether it should be explored further and/or if it should be extracted. Since we do not have a large amount of training data or ability to know in advance which node that should be traversed, it would be appropriate to use reinforcement learning techniques to create and maintain a decision policy. The Q-lambda learning [7] which is based on traditional Q-learning and has the ability of off-line policy learning, does not need to know the complete process model, and relatively quickly approaches the optimal policy. These properties makes the algorithm suitable for the navigation and extraction type process described above.

With this model of the process, a learning algorithm and an interactive user interface during the learning phase, a new type of information extraction system is proposed. Advantages should be that novice users would be able to extract information from semi-structured sources in an easy and efficient fashion. Another feature to consider is to include an automatic re-calibration, i.e. re-training of navigation and extraction policy, if the structure dramatically changes.

1.1 Thesis Goals

The general IE task is an AI-complete problem. It would require full natural language understanding, something that we are far from being able to accomplish today. IE for limited domains and tasks does not require the same amount of knowledge and intelligence, and successful application of such IE systems exists today.

The overall goal of the work presented in this thesis is to improve knowledge management techniques, more specifically in the context of intelligent automated processing of information provided by World Wide Wed services. The following list describes the goals in more detail:

- Propose an approach for a user-driven information extraction system in semi-structural environments
- Suggest a suitable learning algorithm to handle the semi-structural information
- Find a way to handle the dynamic nature of the Web
- Implement a test-bed, or a prototype system
- Indicate future topics of research

[1] The node actually represents only block-level HTML elements, not character-level elements.

1.2 Thesis Overview

Chapter two presents background information for topics related to the main content of this thesis. The purpose is to "set the stage" for the rest of the thesis and show the position in a number of questions. A substantial introduction is given to the agent concept, which otherwise may not be as noticeable in the thesis but still central to the ideas and developments for the project.

A description of the complete information extraction task that is in focus for this thesis is given in chapter three. This chapter explains the different steps and components of the task and their relationships to each other. The ASIE (Adaptive Semi-structured IE) system that is used to evaluate the approach taken in this thesis is presented in chapter four. This chapter is the most practically oriented part of the thesis. It also describes the reinforcement learning algorithm that is used for the navigation step of the extraction task. The extensions made to the algorithm are also described and partly motivated in this chapter.

The implementation of the ASIE system is evaluated in chapter five. It describes the methodology used and details about experiments performed to put the system to test. Results of the experiments are also included in this chapter. Related work is described and compared to the approach in this thesis in chapter six.

The development of any powerful technique may be subject to abuse and possibly lead to undesired consequences. It is therefore important to consider the positive as well as negative effects of the techniques presented in this thesis. An introduction to ethics is given in chapter seven, together with ethical discussions of possible negative effects.

The final chapter provides conclusions of the thesis, what its contributions and limitations are, and identifies possible future work.

Chapter 2

Background

This chapter provides background information for topics that are used in this thesis. The chapter starts with theoretical information about general topics such as knowledge management and becomes more specific and practical at the end.

2.1 Knowledge Management

The major problem in the information society of today is how to handle the huge amount of information that is available. Computer-aided techniques that can automatically find relevant information, perform intelligent analyzes, and provide knowledge rather than information are becoming essential for everyday activities.

Knowledge Management (KM) tries to maximize the use of available knowledge in a system or organization. This implies that knowledge should be accessible, shared, reused, and embedded in the work context to increase the effectiveness of the organization. The knowledge management process can be divided into the following steps [8]:

- Knowledge Goals: Determine Goals for KM Activities
- Knowledge Identification: Create Overview of Available Knowledge
- Knowledge Structuring: Structuring and Integration of Knowledge
- Knowledge Capturing: Acquisition of Knowledge
- Knowledge Dissemination: Goal Oriented Dissemination of Knowledge
- Knowledge Usage: Productive Usage of Knowledge for the Company
- Knowledge Preservation: Storage and Maintenance of Knowledge
- Knowledge Assessment: Assessment of Current Knowledge and Compliance with Goals

The term ”knowledge” is usually interpreted differently from ”information” and ”data”. Data is just the bits or numbers without any defined meaning or context. Information is obtained by interpreting the data and thereby adding a semantic/meaning. For example, a message containing data in a defined syntax can be given a defined meaning/semantic by declaring a language and ontology for the message. The language together with ontologies gives the message semantics for a specific domain of discourse. The information is often descriptive by nature and describes events in the past. To have use of the information, it should also be able to predict things about the future, i.e. the information becomes knowledge. Knowledge can be thought of as a collection of information that is of use for the organization and is therefore predictive in nature [9].

2.2 Information Retrieval

The area of Information Retrieval (IR) [10] has already reached high levels of sophistication and is used by a very large amount of users today. Services such as AltaVista, Google, and Yahoo are known by most Internet users. The main advantage of this type of service is that it is able to handle generic domains and can cover the entire the Web in a single system. However, it is not sufficient for all types of questions or tasks that are needed. The lack of semantic understanding and ability to compile a directly useful answer, i.e. provide knowledge, makes semantically more difficult tasks not appropriate for this type of service.

2.3 Information Extraction

Information Extraction (IE) uses techniques different from Information Retrieval to obtain a higher degree of knowledge from textual information sources [3]. The basic IE system consists of a set of predefined templates containing a set of name-value pairs that are used to extract information from the corpus. The information obtained is structured and relatively easy to analyze to provide directly useful information. The disadvantage is that the templates are typically domain-specific; thus making an IE system less generic than a typical IR system.

Common approaches in IE consist of using statistical and linguistic knowledge to be able to find syntactical patterns that match the set of templates in the system. The use of resources such as WordNet [11] can give more semantically driven approaches by considering the underlying concepts and their relationships instead of the syntactical patterns.

2.4 Semantic Web

The task of IE would be greatly simplified if more semantics were included in the source documents. The documents on the Web today are primarily targeted

towards humans and not towards machines. The amount of background knowledge in humans makes it unnecessary to "tag" the information with semantic clues, it is sufficient with the content and information about how to present it. Machines of today require more information; they cannot interpret the complex meaning of natural language text or the "semi-structural" information[1] that is available on the Web. The Semantic Web initiative [12] tries to define how to include semantic clues in the document so that machines in the future will be able to interpret the information effectively.

One of the applications of IE in the future will be to translate existing documents automatically or semi-automatically into semantically enriched documents. Given that this semantic information is present, information retrieval systems could be greatly enhanced to provide services that are more intelligent.

2.5 Semi-structured IE

A problem with the IE techniques of today is that they are mostly targeted for natural language text, i.e. unstructured text. If more structural information such as tables or lists is present in the document, the system will often fail to employ the linguistic and statistical methods. The term "semi-structured" refers to information that contains more structural information than natural language text, but not as structured as in relational or object-oriented databases. Semi-structured documents may have an irregular structure that does not adhere to any predefined schema.

One of the main conferences for IE, the Text REtrieval Conference (TREC), focuses mainly on the ability to handle natural language text rather than semi-structural text. Many tasks and questions relevant for the user and suitable for IE deal with information that is presented in tabular format, e.g. price of products, schedule of events, and personnel lists.

The task of extracting information from semi-structured sources is not as difficult as extracting information from natural language text. There are more clues, i.e. the structure, to guide the process to the right extraction points. However, some of the common IE techniques such as linguistic rules become less useful and new techniques are needed to handle the structural information.

A large number of applications that work with this kind of task already exist today , e.g. a number of "shopping sites" that extract information about certain types of products from different retailers and compare them with each other to find the best retailer. The Buyer's Guide [13] is an example of such a site that works in the computer products domain in Sweden.

The current approach used in these applications is to use some kind of "knowledge engineering" approach, i.e. a domain expert constructs a set of extraction rules. This makes it difficult to construct and maintain the system. It would be better if the system itself could construct the rules and handle changes in the source

[1] Semi-structural in this context refers to the natural language text combined with information about how to present it

documents that might occur in the future. This suggests that a machine-learning approach would be appropriate.

There are a number of approaches to the task of automated template construction. The basic architecture of these systems consists of an IR module that selects relevant documents from a given corpus and another module that creates templates based on the objects/concepts that interact and their relationships and properties [14]. These systems primarily work with natural language text. The area of *wrapper induction* typically deals with automated construction of wrappers [5], i.e. templates for semi-structured documents. The basic approach consists of a variation of Head-Left-Right-Tail (HLRT) identification in the documents, i.e. to find the extraction points for some kind of list of items.

2.6 Agent-Oriented Development

The system described in this thesis (see chapter 4 on page 35) has been developed based on agent-oriented ideas. The use of agent-oriented design makes complex systems more robust and provides an intuitive view of the system. The main distinction between agents and non-agents is in our expectations and our point of view. Just as some algorithms can be more easily expressed and understood in an object-oriented representation than in a procedural one, so it may be easier for developers and users to interpret the behavior of their programs in terms of agents.

2.6.1 The Agent Concept

The initial meaning of an agent, originated by J. McCarthy in the mid-1950's and coined by O. Selfridge a few years later, was a system that given a goal could perform appropriate actions and that asks for advice when necessary. However, the term has later been used for a variety of purposes. Various definitions have been used such as "something acting on behalf of someone else", "serving as a mediating role between humans and computer software", "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effects", or simply a software entity that executes in the background or is scheduled to run later. These definitions are quite general and can be interpreted to allow anything be an agent.

A popular understanding in the web community of the term agent in recent years is that of a service that collects information from different web sites and presents a compiled result, e.g. an Internet shopping agent. These services could more appropriately be called information agents or wrappers. The term agent should be given a general definition that describes the characteristics of the entity in question rather than a specific task or environment where the entity resides. A preferred definition from Jennings [15] states:

> An agent is an encapsulated computer system situated in some environment and capable of reactive, proactive, and autonomous action in

that environment in order to meet its design objectives.

This definition is specific for software agents, i.e. computer systems, but it captures the most important properties that differentiate an agent from a non-agent. More information about such properties is given in chapter 2.6.3.

2.6.2 Agent-Oriented Programming

The concept of *agent-oriented programming*, as presented in the ground-breaking paper written by Shoham in 1993 [16], introduces a new language paradigm[2]. A substantial amount of work has been carried out since then trying to complete a formalism for intelligent agents. Shoham's view of a software agent consists of mental components such as beliefs, capabilities, choices, and commitments, i.e. *software with mental states*.

An agent according to Shoham is an entity whose state is viewed as consisting of mental components, i.e. it is the view of the entity rather than the entity itself that makes it an agent or not. This view can be applied to small and well-known systems such as thermostats, but it is *most useful* when applied to complex systems that are largely unknown in structure. This can be illustrated through the light-switch example [17]:

> It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmitting current at will, that invariably transmits current when it "believes" that we want it transmitted and not otherwise; flicking the switch is simply our way of communicating our desires. However, while this is a coherent view, it does not buy us anything, since we essentially understand the mechanism sufficiently to have a simpler, mechanistic description of its behavior. In contrast, we do not have equally good knowledge of the operation of complex systems such robots, people, and, arguably, operating systems. In these cases it is often most convenient to employ mental terminology.

Agent-Oriented Programming (AOP) can also be seen as a specialization of the Object-Orientation Programming (OOP) paradigm where the object's (or rather the agent's) states have been fixed to a set of mental states. A new programming language called Agent-0 [18] defines language constructs for mental components such as belief, capability, obligation, and commitment. The control loop of an Agent-0 program starts with gathering incoming messages and updates the mental state, and then executes commitments using the capabilities.

Agent-0 should not be seen as a complete ready-to-use language, but as a base for other languages. However, few actual useful languages have been developed

[2]The AOP term was actually coined in 1989 by Shoham

that fulfill the promise of AOP. Agent frameworks based on traditional programming languages have now reached a higher level of usefulness and have become more popular (see section 2.6.7). This definition of agent-orientation that only addresses the view of a system helps in understanding how a system works but may be of little guidance when designing a system. A definition should also provide details about how a system *should* be designed to be agent-oriented, such as Jennings' definition.

2.6.3 Agent Properties

Agents can be classified by different properties such as the environment they live in, type of task they perform, or their architecture. For example, an information agent can be designed to operate in the Web environment dealing mainly with textual information. It should also be noted that the term agent represents more than just software agents, e.g. humans are also agents. Figure 2.1 shows some examples of different types of agents.

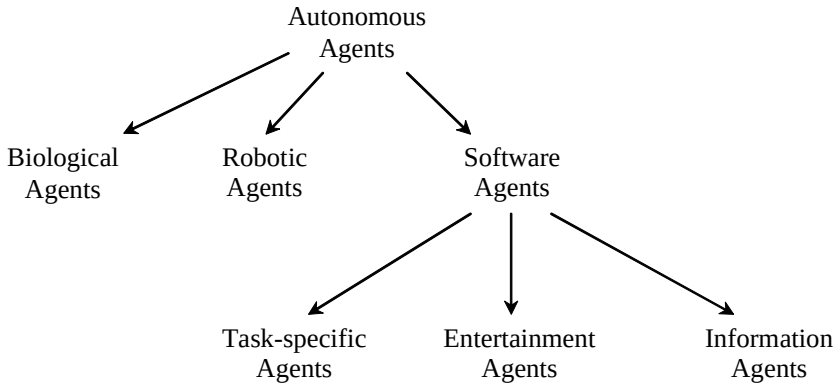

Figure 2.1: Agent Taxonomy (modified from Franklin and Graesser [19])

There are no sharp lines between what an agent is and what is not an agent. The set of properties listed in table 2.1 represents certain features that separate agents from non-agents, but it is not always possible to give an absolute answer as to whether a specific property applies or not for an entity. These properties do however show in an informal way what constitutes an agent.

The following simple example can be used to understand these properties. Consider the difference between a football and a football (soccer) player. The player is obviously an agent and the football is not. The player has reactivity since he/she operates in an environment where he/she can sense input such as vision and audio and he/she can act by muscular power in a timely fashion to reach the goal. The football can neither sense nor act in the environment. The player has autonomy since he/she can change his/her internal state, the football has not. Similar arguments can be made for the rest of the properties. Mobility, the last property, does

not apply to this example since it does not deal with a computer environment.

It should be noted that it is not necessary to fulfill all these properties to be called an agent and it can be difficult to give a definitive yes or no answer to whether an agent possesses a certain property. Few existing software agents have for example real deliberative capabilities. Service agents that do not interact with human users have no need for a human-like personality interface. One of the most important properties that separate agents from non-agents is that of autonomy. A software entity that is completely dependent on someone else to reach the goal, e.g. requires human input, should not be called an agent. However, it is of course allowed and often appropriate for an agent to ask for assistance, but without being completely dependent on the answer.

2.6.4 Agent Architectures

The deliberative and reactive properties previously described also show two classical types of agents. A classical deliberative agent contains an explicitly represented symbolic model of the world, where decisions are made via logical reasoning and theorem proving [21]. Two main challenges with this approach are how to translate the real world into an accurate, useful symbolic description and how to represent and reason with that symbolic information. Techniques such as vision, speech understanding, learning, commonsense reasoning, and automated reasoning need to be solved to be able to use this approach effectively. Current techniques are far from complete in these areas.

A reactive architecture contains no symbolic model of the world; it is much simpler in terms of the computation that they need to perform. Some researchers say that most everyday tasks are ”routine” in the sense that they require little, if any, new abstract reasoning [22]. Most tasks, once learned, can be accomplished in a routine way with little variation. These routines could be encoded into a low-level structure that only needs periodic updating. Agents may have both a reactive and deliberative behavior and perform quick actions for some events and plan for more long-term actions to be performed later. This is called *hybrid architecture*.

An early example of a deliberative architecture is the STRIPS planning system [23]. This system takes a symbolic description of the world, a desired goal state, and a set of action descriptions that characterize the pre- and post-conditions associated with various actions. It then attempts to find a sequence of actions that will achieve the goal, by using simple means-ends analysis, which essentially involves matching the post-conditions of actions to the desired goal. These very simple methods where shown to be ineffective on problems of even moderate complexity. Even with refinements to this method, theoretical results have been shown that indicate that these techniques will turn out to be unusable in any time-constrained system [24]. These results have had a profound influence on subsequent AI planning research and caused researchers to question the whole symbolic AI paradigm.

The mental categories beliefs, desires and intentions (BDI) can be used to describe the internal processing of a deliberative agent. This BDI architecture has

Table 2.1: Agent Properties (adapted from Bradshaw [20])

Reactivity the ability to selectively sense and act in a timely fashion

Autonomy have control over their own internal states and behavior

Goal Directedness the ability to fulfill given objectives, without being told how to fulfill them

Pro Activeness self-starting behavior, ability to initiate actions

Collaborative Behavior can work together with other agents to achieve a common goal

Knowledge Level the ability to communicate with persons and other agents with languages resembling human-like ”speech acts” different from typical symbol-level program-to-program protocols

Deliberative the ability to plan a sequence of actions to reach the goal; to be able to reason about the utility of the actions, perhaps using a symbolic model of the environment

Temporal Continuity persistence of identity and state over long periods of time

Personality the capability of manifesting the attributes of a human-believable character such as emotion

Adaptivity being able to learn and increase performance with experience

Mobility being able to migrate in a self-directed way from one host platform to another.

```
initialize();
while (true) {
   options = optionGenerator(eventQueue);
   selectedOptions = deliberate(options);
   updateIntentions(selectedOptions);
   execute();
   getNewExternalEvents();
   dropSuccessfulAttitudes();
   dropImpossibleAttitudes();
}
```

Figure 2.2: Abstract BDI Interpreter

become a popular model in the design of agents and been used since Bratman et al in 1987 [25]. A sample of an abstract BDI interpreter can be seen in figure 2.2 that gives a basic understanding of the process of a BDI agent [26].

Beliefs are usually modeled using possible-world semantics where a set of possible worlds is associated with each situation. The set of beliefs are dynamic and depend on sensory input from the world. The desires of an agent specify what preferences the agent has. The desires need not be consistent with what the agent believes to be possible, it can be a set of preferred future world states or courses of action. The goals of an agent are a subset of desires that are currently possible and can be used to define what options an agent currently has. The term *strong realism* refers to the goals also being a strict subset of the beliefs. The intentions of an agent are the subset of goals that the agent has the intention to perform. The agent can seldom perform all goals, even if they were consistent, due to resource limitations. This process of selection of goals/actions to perform is called the *formulation of intentions*. The plans of an agent consist of a sequence of intentions to achieve a higher goal, thus an intention can be seen as a partial plan.

A popular architecture for hybrid agents, i.e. agent with both reactive and deliberative capabilities, is the Reactive Action Packages (RAP) architecture [27]. This architecture consists of the three layers *planning*, *executor* and *controller*, see figure 2.3. This model of three layers is very characteristic for hybrid agents.

The planning layer manages the deliberative tasks such as planning and produces sketchy plans for achieving goals using internal knowledge about the world and a plan library. The executor fills in with details in the plans when they are about to be executed. The executor is also able to detect failure and to select alternative plans to achieve the goal. It is also able to provide primitive actions to the planner.

The controller provides sensing routines that give information about the world to the executor and behavior routines that handle the reactive nature of the agent. The behavior routines handle things like collision detection that need to be dealt with in a timely fashion. Firby also focused on the interface between continuous and symbolic robot control, i.e. how to turn symbolic actions into continuous processes. The planning layer deals with the symbolic representation of the world

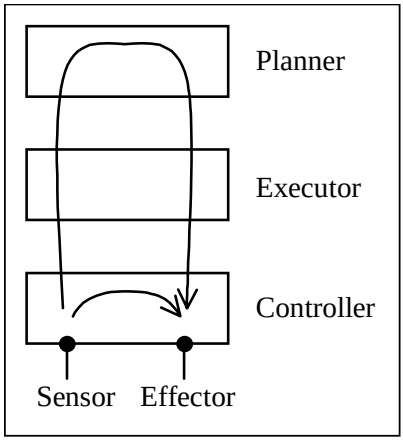

Figure 2.3: The RAP's Architecture

and the executor turns the symbolic and discrete actions into a sequence of continuous processes. This architecture does not include interaction with other agents, something that is required in the strong notion of an agent. There are extensions to this architecture, such as InterRAP [26], that extends the model with an additional cooperation layer that can generate plans that satisfy multiple agents and add inter-agent communication capabilities.

2.6.5 Agent Design

The agent-oriented paradigm is of particular use when designing complex systems. To be able to handle these complex systems effectively, they need to be *decomposed* into smaller sub-systems, *organized* to find appropriate relationships, and *abstracted* to find a suitable level of relevant details [28]. As seen in figure 2.4, complex systems can often be organized into a hierarchy that consists of sub-systems at different levels of abstraction. It is typical for these sub-systems to interact highly within themselves and only rarely with other sub-systems.

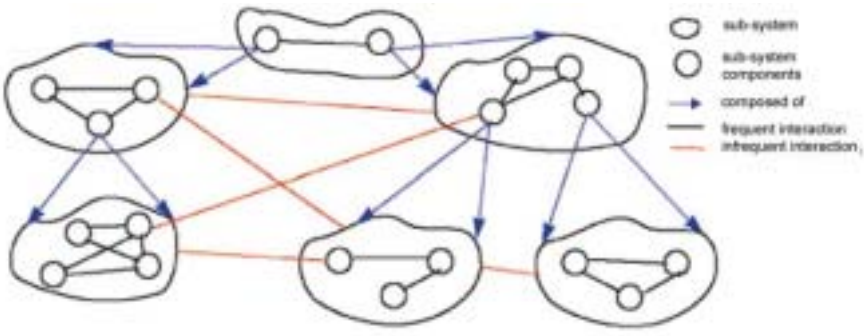

Figure 2.4: Relationships in Complex Systems [28]

The development of a system can be handled more effectively if the designer can focus at smaller sub-systems separately, i.e. the system needs to be decomposed. Complex systems can typically be decomposed into smaller parts that interact mainly with itself and only rarely with other sub-systems. This makes it possible to handle each sub-system separately as an autonomous and flexible unit. A sub-system works together with related sub-systems to reach the goal of the parent system. This motivates the choice to decompose a system by considering the goals or function they provide, rather than the data they provide as in traditional OO. Current trends in software engineering also motivate the design of proactive and autonomous entities [28, 29], i.e. agents with their own thread of control able to reach their goal and deal with unknown situations.

The abstraction of a system provides an intuitive model where the designer can focus on the relevant properties. The decomposition into sub-systems of different abstraction level is one way to abstract that is already motivated above. The interaction between sub-systems can also be abstracted as high-level social interactions, which is a natural way to view communication. The method-invocation type of communication can be difficult to use for complex systems. It removes the autonomy of the entities and requires the caller to consider not only failures for itself but also failures for the responder and all actions that it will perform. An agent-oriented approach using speech-act based communication protects the autonomy of the entities and allows entities to communicate using standardized protocols and languages (see chapter 2.6.6).

As seen in figure 2.4 above, a number of relationships at different levels of abstraction exist in complex systems. These relationships can be of different types such as part-of, type-of, client-server, peer, and team relationships. It is also possible that these relationships are dynamic, i.e. they change over time. The traditional class structures used in OO design provides little support for representing these types of relationships in an intuitive fashion. Agent-oriented frameworks such as JADE (see section 2.6.7) allow these relationships to be represented as behaviors implementing interaction protocols as defined by FIPA [30]. It is therefore possible to handle dynamic relationships of different types in an intuitive fashion.

2.6.6 Agent Communication

The ability to inter-operate and reuse software components is central to software engineering. This requires a way of interacting and communicating between the components. The use of method invocation as used in traditional object-orientation is limited to communication only between other components written in the same programming language and platform. Component-based communication, e.g. CORBA, COM, and JavaBeans, increases the ability to communicate to other components by introducing a standardized interface that makes it independent of the programming language used to implement the component. Agent-oriented high-level communication based on standardized protocols allows communication independent of programming language, platform, or network protocol. However,

this requires well-accepted agent communication language standards, something that has not yet been achieved [31].

Another problem with method-invocation communication is the lack of semantics included in the messages. The meaning and reason for the invocation is not clear from the invocation itself and thus an increased responsibility is placed on the designer to consider the effects of an invocation. Agent communication typically consists of speech-act based messages that stand a level above method-invocation communication. This does not replace existing techniques such as RMI or CORBA; they are still used during agent communication at a lower level and it is hidden from the designer. The use of specific content languages and ontologies in agent communication allows messages to contain a higher level of semantics and thus increase the ability to communicate with other agents.

The two major classical agent communication languages (ACLs) are KQML [32] and FIPA ACL [33]. According to the authors of KQML, agent communication languages should exhibit properties such as declarative form, separate content language and message language, clear semantics, efficient implementation, and be independent of network transport mechanism[32]. An ACL message typically consists of three abstraction layers as seen in figure 2.5.

Figure 2.5: ACL Message Format

The highest layer, the content layer, contains the actual meaning/purpose of the message. This content is encoded by specified language and uses a vocabulary that is specified by the ontology. The ontology specifies available concepts and relationships between these concepts. For example, if an agent is talking to other agents about a shopping domain, the ontology could contain concepts such as vendor, customer, products, price, order and rules for how propositions can be constructed with these concepts. The language specifies how to encode these propositions into a single serial digital unit, e.g. LISP or XML syntax is often used. It is important to have a clear semantic in the message to avoid incorrect interpretations by other agents. This will increase the ability to talk to agents that have been designed by other persons and organizations.

The communication layer adds information to the message such as the sender and receiver name for the message. The low-level message layer specifies which network protocol to use and how to encode the message before transmitting it. It also adds information about which language is used in the content of the message, which ontology to use, and which performative (speech act) is used. See figure 2.6 for an example of a KQML message.

```
(ask-one
     :sender joe
     :content (PRICE IBM ?price)
     :receiver stock-server
     :reply-with ibm-stock
     :language LPROLOG
     :ontology NYSE-TICKS)

(tell
     :sender stock-server
     :content (PRICE IBM 14)
     :receiver joe
     :in-reply-to ibm-stock
     :language LPROLOG
     :ontology NYSE-TICKS)
```

Figure 2.6: KQML Message Example [34]

The ACL developed by FIPA is similar to KQML in most aspects. The main difference is the semantics of the speech acts [34]. Therefore, it is not possible to match exactly the acts from KQML to FIPA ACL and vice versa. FIPA ACL use a content language called Semantic Language (SL) (as opposed to the KIF language commonly used in KQML) that is a form of multi-model logic with modal operators for beliefs, desires, uncertain beliefs, and intentions. FIPA agents need to have some understanding of this language to process ACL messages. However, by using frameworks such as JADE, most of this processing is handled automatically and hidden from the designer.

The XML language is proposed as a standard that shall make machine-readable documents by providing a clean and formal design. It should be noted that XML provides a low level syntactic standard that makes the documents machine-readable, not machine-understandable. Despite the presence of DTDs and XML Schemas that describe the XML document structure, this only describes the grammar for the document and not the semantics of the document. In addition, an XML document does not have to conform to a given DTD or XML Schema, the only requirement is that the document is well-formed. In practice, XML is being used as serialization syntax for other markup languages, as a data-exchange format for computer applications, and as content markup language for Web pages with a connecting XML style sheet to transform the document to suitable presentation format.

The Resource Definition Framework (RDF) [35] is a proposed standard for metadata, i.e. descriptions for resources on the Web, although it is not limited to only working with Web resources. The goal is to make machine-understandable documents by providing a higher level of semantics than present in for example XML documents. The basic structure of an RDF document is a set of triples where each triple consists of an object, an attribute, and a value. The triple is

normally written as A(O,V), for example `hasPrice('http://www.books.org/ISBN0012315866', '$62')`. This is actually all that is defined in the RDF model, i.e. the model itself is domain independent and does not place any restrictions on the names of the objects or attributes used. The RDF model does not specify any mechanism for reasoning; it can be characterized as a simple frame system, which can be used as a base for a reasoning system. The serialization syntax of RDF documents is proposed to be XML documents, but other formats are also possible.

RDF Schemas can be used to define a particular vocabulary that should be used for RDF attributes and it allows the specification of the kinds of object to which these attributes may be applied. This is similar to how XML Schemas define the vocabulary and rules for elements and attributes that should be used in XML documents. For example, two crucial RDF Schema constructions are subClassOf and subPropertyOf that define a hierarchical organization of types (classes) of objects and a hierarchical organization of properties of objects. Furthermore, constraints on properties can be specified using domain and range constructs.

The use of RDF and RDF Schemas for communication between agents and gain a semantic inter-operability has significant advantages over XML due to the entity-relationship model (which is a natural way of describing a domain) that is used in RDF documents instead of an arbitrary grammar defined in an XML Schema. Of course, this does not solve the general problem of finding semantic-preserving mappings between objects. However, the usage of RDF for data interchange raises the level of potential reuse much beyond parser reuse, which is all that one would obtain from using plain XML.

DARPA Agent Markup Language (DAML) is a part of the DAML program with the goal of creating technologies that will enable software agents to dynamically identify and understand information sources, and to provide inter-operability between agents in a semantic manner [36]. Besides creating an agent language, the program contains tasks to create tools that embed DAML markup in web pages and similar documents. Internet markup languages must move towards making semantic entities and markup. DAML will be a semantic language that ties the information on a page to machine-readable semantics.

A first draft release for the ontology core of the DAML language was released in October 2000. The draft defines classes, subclasses, properties and a set of restriction theorems. No explicit inference rules are included yet. The language core was developed by a group of researchers from MIT. A number of people joined the group later, including representatives from the OIL effort [37], the SHOE project [38], and the KIF work [39]. Recently, research groups from Europe have also joined the DAML effort, including Blekinge Institute of Technology, Sweden.

DAML-ONT markup (and later DAML+OIL [40]) is a specific kind of RDF markup, which in turn is written in XML using XML Namespaces and URIs. A DAML-ONT document consists of an Ontology root element, elements with meta information such as version and comments, elements for defining classes, subclass of, disjoint of class, and elements for defining properties, unique properties, unam-

biguous properties, sub properties, equivalent to, inverse of, transitive relation, and more. The expressiveness of DAML+OIL is significantly higher than the expressiveness of RDF.

2.6.7 Agent Frameworks

The development of agent systems is a complicated process that needs various tools and platforms to be efficient. A number of frameworks exist today that support building agent-systems. They provide features such as class libraries for development support, platforms for executing agents, and services to manage the agents. A large collection of links to different agent frameworks and other resources can be found at the AgentWeb, MultiAgent and AgentLink sites [41, 42, 43].

This chapter will give a short description of JADE that is the framework used in the system described in this thesis. A short introduction to the Via framework and other related techniques are also given for comparison.

Java Agent Development Framework (JADE) [44] assists in the development and execution of multi-agent systems. One of the main advantages of JADE is the comprehensive support of the FIPA specifications [30], especially in agent communication. It is ironic that although agents are supposed to increase the ability to communicate with other systems, a well accepted standard for agent communication does not exist [31]. The support of FIPA standards is therefore significant. The framework provides a class library for developing agents, a platform for execution of agents that includes white- and yellow page services, and a message transport and parsing service. The agent communication language is the FIPA ACL that is very similar to the widely known KQML language. It also supports using and building ontologies to support increased semantics in the agent communication. JADE is available for free download [45] under the LGPL license.

The JADE framework can also be combined with JESS and the Protege toolkits to provide support for agents with deliberate capabilities [46]. JESS is a Java-based rule engine that supports reasoning with declarative rules. Knowledge can be added and edited with the Protege tool that provides a graphical interface for editing ontologies and knowledge-bases.

The Via system [47] is a commercial software product that provides a framework for developing agents. It contains an API for building agents, a toolkit of multiple communication and network services, built-in user-tracking features, an agent server, and an easy-to-use graphical client interface. The system is built entirely in Java; allowing agents to operate on any platform with a Java virtual machine.

The graphical client interface allows novice users to manage available (i.e. already developed) agents in the server and the server loads and unloads agents from RAM when needed to optimize server resources. The actual agents normally consist of three parts: the agent, stimulus tasks, and action tasks. These parts have their own thread of execution and can do their job at the appropriate time. The stimulus tasks can be seen as the sensor of the agent that obtains information for

the agent. The action tasks can be seen as the effectors that perform some action in the environment. The Via system comes with a number of ready-to-use stimulus and actions that help the developer in the process of designing an agent.

The development of agents is very similar (syntactically) to the development of Java applets. The ViaAgent interface is inherited instead of the Applet interface, and similar methods such as start() and stop() are implemented. The developer chooses certain properties that are exposed to the user, e.g. an email address that will receive some notification when the agent wants to inform the user. The user of the agent can modify these properties with the graphical client interface. The package comes with a number of examples of simple agents, e.g. LoginWatcherAgent that notifies the user when users log in and out of a UNIX server. The system is free for educational purposes and can be obtained through Kinetoscope's homepage.

A related technique that shows much promise in becoming a popular way of interacting over the Internet is Web Services. A Web Service uses a set of XML-based languages to locate and communicate with other systems in traditional low-level method-invocation style. Support for building Web Services exists for most programming languages and platforms, including Java. Microsoft's .NET initiative [48] even provides new programming languages that have intrinsic support for Web Services, which makes the development of Web Services very efficient and intuitive. The main advantage of Web Services is the ease of development and use of W3C standardized XML-based languages. There is however no support for high-level declarative communication as in agent communication. The use of high-level speech-based agent communication is possible with Web Services as a lower layer; thus, they can complement each other to increase the inter-operability further.

2.7 The Buyer's Guide System

The interest in information extraction and the work for this thesis began with the development of the shopping agent site called "The Buyer's Guide" [13]. The purpose of that site is to guide and assist consumers of computer products in their purchase. It collects information about computer products from different retailers and compiles "tracks" a list of where to find and buy the best products. It is also possible to execute advanced queries to find appropriate products.

The motivation for constructing this web site back in 1997 was that no similar service existed at that time and it was very difficult to manually compare computer products from different retailers. The site has now been running for about 5 years, it collects information from about 40-50 retailers, has about 1 million page hits per month, and has been voted in the top 100 web sites in Sweden for the last four years by Internetworld. The following section gives a brief overview of the system and what problems that have been experienced administrating the service.

2.7.1 System Architecture

The information about computer products is extracted from the retailers' homepage. The retailer does not have to construct or change anything on the web site; the same pages as used by normal visitors are used. Figure 2.7 gives an overview of the main components of the system. The agents work independently of the web site and update information in a database. The information stored in the database can subsequently be used by other agents to analyze the information further and to present information in the web site.

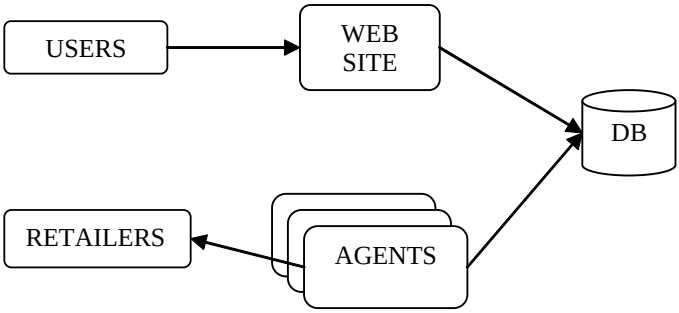

Figure 2.7: TBG Architecture

2.7.2 Extraction Approach

The technique used by the agents to extract information has basically been built using a knowledge engineering approach. A set of heuristic rules has been constructed that allows an agent to start on a given page and navigate through the retailer's site to find pages that contain different product categories. The information about computer products are normally classified into different categories by the retailers and the information about the assigned category is valuable to the agent. Figure 2.8 gives an overview of the extraction agent process.

One of the more important tasks of the extraction agents in the site is to classify the products into a common taxonomy. Thus, all the products from the retailers shall be classified into a common taxonomy even if the taxonomy used by the retailers differs substantially.

The information about product category, manufacturer, retailer, price range and the description of the product is used to be able to identify identical products from the retailers. The technique used to find identical products is again based on knowledge engineering heuristic rules.

2.7.3 Extraction Problems

One of the first problems that occurs with the agents in the system is how to locate the wanted information in the site. A specific agent is developed for each retailer. The addition of a new retailer involves starting with a generic agent and adding

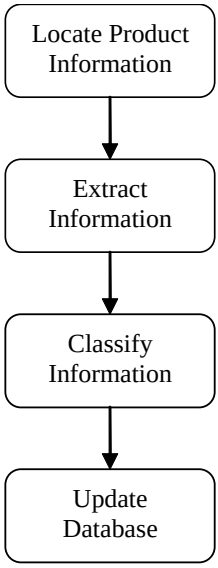

Figure 2.8: TBG Agent Flowchart

some retailer specific details to that agent. These details involve rules for how to navigate through the site and how to classify the products.

The rules used to navigate the retailers' site basically consist of Perl 5 regular expressions. By combining use of common and specific rules for the retailers and the power of regular expressions, the time required to add a new retailer is quite short. However, the development of these rules requires significant experience and knowledge in regular expression and programming.

The static nature of these rules also presents a problem when the structure of the retailer's web site is changed. The rules are made as generic as possible to allow for some changes in the web site, but they fail to work if the structure of the web site undergoes significant modifications.

One of the agents has the task of analyzing the information gathered by the extraction agents and identifying identical products. The detailed information that has been provided by the extraction agents, i.e. slots such as price, manufacturer, model, size and speed of the product, makes this task significantly easier than a pure natural language approach of the description of the product. This indicates the need for an intelligent extraction process. The rules for product identification are however static as well and it would be useful if they could handle the dynamics of the product information. For example, the prices of computer memory change rapidly and could decrease to 20% of the original price in as little as one month.

2.7.4 Conclusions

Even though the extraction rules work quite well, they do require significant knowledge to develop. If information extraction is to reach a similar level of coverage

as information retrieval, novice users must be able to adapt and create extraction services for new domains. This indicates the need for user-driven and intelligent extraction systems.

The system must also be able to handle the dynamic nature of the Web. As experienced with this service, it is common that retailers change the structure of their web site and the system must at least be able to handle small structural changes.

The use of artificial intelligence techniques in tasks such as product identification is essential and very effective, e.g. use of Bayesian networks to classify and identify products. Product identification would however be very difficult to perform without intelligent extraction techniques that provide sufficient information to work with.

Chapter 3

The Extraction Task

The need for efficient knowledge management tools is essential for the everyday activities of today and the need will continue to increase with the amount and flow of available information. Information Extraction is a type of knowledge management tool that given a set of slots, i.e. fields can be filled with specific information, produces a result set where these slots are filled with appropriate information from given documents. The resulting information is highly structured and can be easily analyzed by users of the system.

Information extraction can be used for different types of source documents and tasks. Chapter 3.1 describes the type of system that is the focus of this thesis. A system that performs this type of information extraction can be divided into different modes or stages such as a training mode and an extraction mode. Chapter 3.2 describes how such a system can be divided into different modes. The type of source documents that are the focus of this thesis need a model of how to represent the content of these documents. Chapter 3.3 describes how such a model can be designed and why it is appropriate for this type of extraction task.

One of the major parts of the extraction task that has been examined for this thesis is the navigation step. The navigation step involves how to find the optimal path from a given starting point to the desired extraction points. Chapter 3.4 describes what that step involves and how to handle it. A problem that occurs for this type of extraction task is how to manage the dynamic nature of the Web and specifically how to handle structural changes. Finally, to let the system be able to handle more complex extraction tasks and to be less dependent on structural information, additional techniques need to be implemented. Chapter 3.5 introduces the problem.

3.1 Extraction Task Types

The type of information extraction system that is the focus of this thesis is systems that deal with semi-structural documents, i.e. documents that contain more structural information than natural text but not as structured as in, for example,

relational or object-oriented databases. Such databases have a pre-defined schema to which the data must conform. Semi-structured documents may have missing or additional information, i.e. have an irregular structure that does not conform to a predefined schema. The main type of application for semi-structured IE is intelligent information agents that surf the World Wide Web and take advantage of the semi-structural information in HTML documents.

This type of extraction system can be designed for specific domains by programmers that posses skills in programming and have knowledge of the domain of discourse. It would be useful to have general-purpose extraction systems that can handle any domain, similar to how information retrieval systems like Google and AltaVista can handle nearly the entire Internet. However, the current state of the art in the information extraction area is not able to handle such wide domains. Since it would be too resource demanding to have programmers design extraction tools for every domain, it is necessary to allow users without programming skills to design or at least adapt systems to new domains. The term "user-driven information extraction" (UDIE) [49] refers to this type of system that allows users without expert skills to efficiently adapt and use such systems.

Information extraction systems that deal with natural language text have reached high levels of performance with the use of linguistic techniques such as part-of-speech tagging, named-entity taggers, co-reference analysis, and word sense disambiguation. Additional improvements can be achieved with shallow semantic analysis such as using WordNet to find synonyms and being able to merge similar appositives.

The additional structural information that exists in semi-structural documents provides a significant amount of knowledge of how to locate the wanted pieces of information for the extraction task. For some tasks, the structural information is sufficient by itself to complete the extraction task successfully without additional linguistic or semantic knowledge, e.g. extraction of news headlines from CNN's web site [4]. The linguistic techniques used in many natural language extraction systems unfortunately do not work as well in semi-structural documents. The text is often less grammatically correct and contains mostly choppy sentence fragments [4]. Therefore, the structural information becomes even more important when working with semi-structural documents.

For semi-structural extraction tasks, it is common that a set of documents are used repeatedly that have the same structure as in previous extraction. For example, when extracting news headlines from CNN's web site, the page structure remains constant over time and the textual content varies. It is therefore possible to reuse knowledge about how to extract information from previous extractions. Hence, the structural information provides crucial knowledge for the extraction task and the system can be effectively trained to extract information from certain domains using that information.

The advantage of using machine-learning techniques, as opposed to traditional knowledge-engineering methods, is that less work is required of the programmer. Thus, the system becomes more user-driven. The use of supervised learning tech-

niques to train extraction agents, such as in ShopBot [50] and Webfoot+CRYSTAL [4, 51], may still require a lot of work from a domain expert to produce a sufficient training set for each domain. By using reinforcement learning techniques, the extraction agent is able to learn from its own experience when interacting with the environment; thus reducing the amount of work from a domain expert to produce a large training set.

3.2 System Modes

It is possible to identify at least the following three different stages, or modes, of the extraction process. First, the system needs to be *trained* to extract the information. Second, the system must *extract* the information. Third and finally, the system can *answer* questions from the user. If the system is truly user-driven, a user without expert skills should be able to handle all of these stages. Figure 3.1 shows an example of the relationships between the different stages (or modes) of an extraction system.

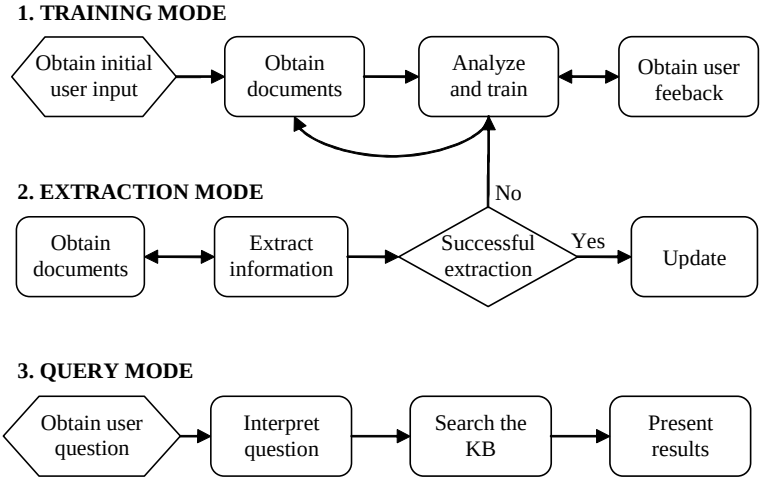

Figure 3.1: Extraction System Modes

3.2.1 The Training Mode

During the training mode, the agent finds an optimal decision policy to be able to extract wanted information. The process is initiated by giving essential user input such as the address where to start looking for information and a few examples of wanted information pieces. Given the starting address, the agent is able to download the initial document and start analyzing its contents. Given that the environment is hypertext documents, e.g. Web pages, the documents may contain links t other documents. If the agent during its training decides to examine the

child nodes of the link, it has to download these additional documents as well and analyze their content.

The exploration of the hypertext tree[1] may continue until a maximum depth of the tree has been reached or until no further documents are downloaded and analyzed. Additional details of how this navigation step works is given in section 3.4.

The navigation step alone is rarely sufficient to complete the extraction task, but it provides one useful step in the process that exploits the structural information available in semi-structural documents. If sufficient semi-structural information is present in the documents, none or only a small amount of extra work is needed to complete the extraction process.

There may also be a final step in the training mode where the user can examine the results of the training and give additional feedback. If insufficient user input was given at the start of the training, the user can provide additional examples of wanted information pieces and reject some information pieces that were suggested by the agent.

3.2.2 The Extraction Mode

When the agent has been trained to extract the information, it may start to execute the actual extraction of information. Remember that the typical semi-structured extraction task consists of a source of information where the structure remains mostly constant and where the content changes over time.

The extraction mode is similar to the training mode. It starts by downloading the initial document and analyzing its content. However, the agent no longer explores unknown paths of the hypertree as done during training. It exploits the knowledge stored in its decision policy to find the optimal path to the extraction points. The extraction mode is finished when no additional extraction points exist. The extracted information is either returned to the user or stored in a knowledge base where it can be further analyzed.

The navigation step alone may not be sufficient if the extraction task becomes more complex. It is often necessary to postprocess the text, for example, to remove remaining HTML elements. Other IE components such as named entity recognition, part-of-speech tagging, co reference resolution, and use of semantic resources would improve the utility of the extraction system. The navigation step provides the ability to locate relevant pieces of the source documents and return a hierarchical structure of these pieces.

A problem occurs if the structure of the source document suddenly changes. The decision policy no longer works and the agent has to start exploring for new paths to the desired information. If the structure changes, it is usually easy to detect the change since the path does not lead to wanted extraction points. More formally, the rewards returned and the total return obtained during the extraction does not

[1] See chapter 3.3 for more information about the hypertext tree

match the rewards and the total return obtained during training. As indicated in figure 3.1, the agent can automatically jump from extraction mode to training mode to find a new decision policy. This has not been investigated yet and needs further research.

3.2.3 The Query Mode

The extraction system may also have a user interface where the user is able to analyze the information that has previously been extracted. However, for some applications this may not be appropriate. For example, consider a news filtering application that is supposed to extract news articles from different sites and select those articles that are of interest to the user. The extraction system may only send an email or change the content of a web page when a new article is found, i.e. no user interaction is taking place. For other applications such as Buyer's Guide 2.7, i.e. shopping agents that compare and find the most suitable products for users, the users may want to be able to query the database of extracted information and select suitable products themselves.

The query mode is a standard search system where the user is presented a search form and the system interprets the query, executes the search, and presents the results. However, due to the high level of structure that exists in the extracted information, it is possible to provide a more advanced form with fields of specific data types that match different slots in the extracted information.

3.3 The Hypertext Model

The process of the information extraction task in semi-structural environments can be designed differently than in traditional natural-text environments. Instead of trying to map templates of different slots to a set of words or sentences, the document, or more specifically a set of connected nodes in a set of linked hypertext pages, can be viewed as a large parse tree where certain nodes shall be extracted. Similar models have been used by for example Rennie and McCallum [52] where each hyperlink represents a node. However, structural information other than the links is not used.

The model suggested in this thesis consists of a tree with nodes representing certain HTML elements in each page, including not only links but also other information. Figure 3.2 demonstrates how the hypertext linked pages are transformed into a hypertext tree. One step in the extraction task consists of locating the appropriate nodes that contain the relevant pieces of information, i.e. the *navigation task*.

A possible problem with using such a model of the source documents is that a single extraction point may be larger than the size of a single node, i.e. the wanted extraction pieces may span several nodes. A possible solution is to mark all of the nodes within the extraction unit, including child nodes, as desired extraction

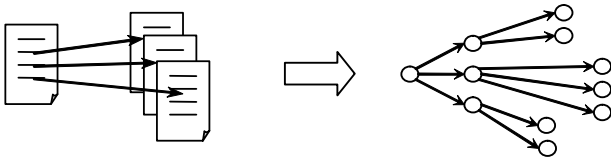

Figure 3.2: Hypertext Transformation

nodes. However, this may lead to unnecessarily large trees that will take a long time to train and require a large amount of memory.

A more suitable solution is to for example only represent so called HTML block elements and treat character-level elements such as FONT and B elements as normal text connected to a node in the tree. This will reduce the size of the tree without losing valuable structural information. The character-level elements usually do not provide valuable structural information for the extraction task.

Each block element according to the HTML specification [53] is represented as a node in the tree with subsumed block elements as child nodes. The A HTML element is given special consideration since it is an in-line element, but still provides valuable structural information and is highly important for the extraction process.

3.4 The Navigation Step

Given that the source documents have been transformed into a tree where each node represents structural elements in the documents, it is now possible to use this tree to locate the extraction points.

The navigation task for information extraction involves finding the optimal path from a given starting point in the parse tree to the extraction points. It is defined as a set of states S, a set of actions A, a transition function $T : S \times A \rightarrow S$, and a reward function $R : S \times A \rightarrow R$. Given an initial state, the goal is to find the optimal transition function that maximizes the sum of rewards.

The intuition behind this approach is that it is similar to how humans would act if they were to complete the extraction task, i.e. to start from some node and traverse the hypertext information through the parts of the page and by following existing links up and down the tree. Of course, humans additionally interpret the information present in the source document in a much higher linguistic and semantic fashion. However, the main similarity is that a decision as to whether a node should be further explored is based on future actions. This model of the source documents allows the system to similarly trace future actions back to previous actions and improve performance with its own experience.

3.4.1 Learning to Navigate

During the training mode of the extraction system, a decision policy should be constructed that can be used to navigate to the wanted extraction points. Reinforcement learning techniques are suitable for the learning part since the utility of each action in the navigation task is based on future actions.

The tree model can be seen as a Markov decision process suitable for reinforcement learning algorithms. The extraction system is able to explore the tree structure and *navigate*, i.e. find the optimal path from a given starting point to desired extraction points. The learning algorithm used for this purpose is presented in a separate chapter since it is an important part of the system and vital to make the system more user-driven and maintain a sufficient performance. More information about the learning algorithm is given in chapter 4.4.

If the extraction task is not too complex, the extraction points could have been simply located by performing a breadth-first systematic search through the parse tree and skipping the training of a decision policy. Studies have shown that reinforcement learning techniques can outperform these techniques by a factor of three or more [52]. This becomes more evident when the complexity of the task increases. The advantage of having this kind of reinforcement learning with explicit modeling of future rewards is also that more user input as well as other heuristics and knowledge can be used at different points during the navigation process. Thus, the reward function can contain more feedback than just for example one for extraction nodes, -0.1 for links and zero for other nodes as used in the experiment described below. The intuition is that this process corresponds more closely to the behavior of human manual extraction. For example, consider the task of locating information about products at a vendor web site. These products are often grouped into categories and there are additional hints at the web site that lead the Surfer to the final destination. By incorporating this information into the reward function, the system will be able to skip irrelevant links and concentrate on paths that are more relevant.

The goal is to be able to complete the extraction task with as little input and direction from the user as possible. Besides the advantage of providing a user-driven IE system, this also enables the system to automatically handle structural changes in the source documents. If the source structure changes so that the decision policy suddenly does not lead to the extraction points, it can automatically start a new training phase to adapt to these changes.

3.5 Additional Steps

The work behind this thesis has so far primarily dealt with the navigation step that imitates the intuitive way of surfing through hypertext and takes advantage of the structural information. That step alone may not be sufficient to complete the extraction task. The information that has been located in the navigation step often

needs to be further analyzed. At the very least, the text should be post-processed to remove for example HTML elements.

If the task is more complex, it may also be necessary to implement traditional information extraction NLP techniques such as named entity recognition, part-of-speech tagging, co-reference resolution, and use of semantic resources such as WordNet [11]. The implementation of these techniques could also be integrated into the reward function to improve the performance of the navigation step further. However, as previously stated, natural language processing techniques unfortunately do not work as well in semi-structural documents. The text is often less grammatically correct and contains mostly choppy sentence fragments [4]

A common situation in semi-structured extraction tasks is that the information is grouped or split into several pieces in some way. For example, a list of computer hard drives may be grouped by manufacturer and split into multi-page tables to limit the length of a single page. The consequence of this is that items in subsequent pages are not considered to be "siblings" of the items in the first page. As shown in one of the experiments in chapter 5.2, the user may give some examples of wanted extraction pieces and specify that siblings should be extracted. Therefore, it would be advantageous if the items in subsequent pages in multi-page tables were considered as siblings. This could be accomplished by implementing heuristic rules that change the structural information accordingly. This is a topic of future research that needs further investigation.

Chapter 4

Design for the ASIE System

The Adaptive Semi-structured Information Extraction (ASIE) system was designed to evaluate the approach described in chapter 3 and to be used as a test bed for future developments. The system is still under heavy development and is not to be considered as a fully-fledged information extraction system.

The design of the system is based on agent-oriented ideas as described in the background chapter 2.6. The agent design and the platform used are described in chapter 4.2. The architecture of the system and its various components are presented in subsequent sub-chapters.

4.1 Methodology

The system described in this chapter is used to evaluate the approach described in the previous chapter. It would have been advantageous to use standardized test applications such as those defined in the TREC conferences. However, the type of application and approach makes these tests not currently suitable. One of the main properties that should be evaluated is the level of user-driven nature of the system and not performance such as precision and recall. This type of evaluation does not exist in these conferences.

The system is designed to initially confirm that the approach works with satisfactory performance and approach level of user interaction. It will also be used to conduct additional evaluations as the system evolves. These evaluations include tests of performance, adaptiveness to new domains and the user-driven nature of the system.

4.2 Agent Platform

The system is built with and upon the JADE framework [45]. The framework provides a package to assist the development of agents and a platform where the agents can be executed. One of its main advantages is its strong compliance with the FIPA standards [30] and that it is implemented in Java.

To assist in the execution of agents, the framework provides a distributed platform that can be split across several machines. Additionally, it provides a graphical management user interface, white-page and yellow-page services, and efficient means of agent communication based on traditional protocols such as CORBA. The underlying protocols such as CORBA are invisible to the developer who can only work with FIPA-compliant ACL messages. The Java classes provided with the framework simplify the development of the agent and support for example the development of custom communication languages and ontologies. Figure 4.1 shows the architecture of the JADE platform. The Agent Management System agent manages all the agents in the platform and provides a white page service. The Directory Facilitator agent provides a yellow page service where agents can search for other agents that for example provide a specific service.

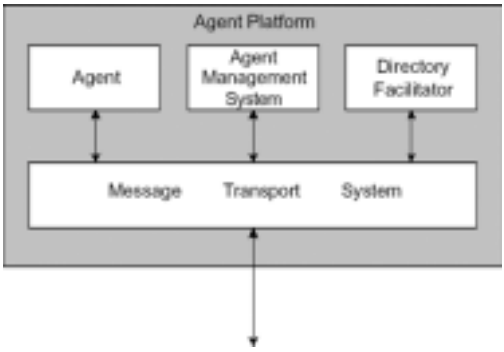

Figure 4.1: JADE Architecture

4.3 System Architecture

The architecture of the current implemented ASIE system, as shown in figure 4.2, consists of Surfer agents that are able to efficiently download and handle web pages on the Internet, Analyzer agents that take requests and analyze documents to find the relevant pieces of information, and Butler agents that communicate with the user. Currently all agents are located on a single machine, but since the JADE platform supports a distributed platform and mobile agents, the agents could be located on different machines and moved during execution if necessary.

4.3.1 The Butler Agent

The Butler agent handles communication between users and the extraction system. Currently, a user interface is implemented that allows for communication to human users. A Web service interface is also planned to be implemented in the Butler agent to allow machine users of the system as well.

The Butler agent allows users to train the extraction system for new domains and to schedule extractions and manage the trained agents. Currently, only the

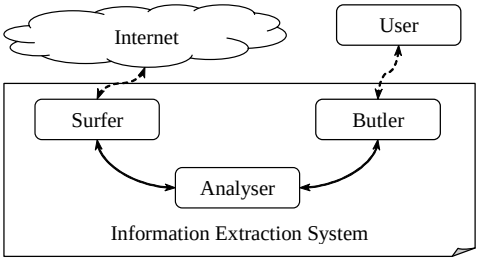

Figure 4.2: System Architecture

training of the agents interface is implemented, but future versions should also support management of the agents and possibly also provide support for direct queries of the data that has been extracted. A screen-shot of the Butler training interface is shown in figure 4.3. It is implemented as a Java applet embedded in a web page that communicates with the Butler agent over the network.

Figure 4.3: Butler Agent User Interface

4.3.2 The Surfer Agent

The Surfer agent has the capability to download web resources from the Internet. As shown in figure 4.1, it is used by the Analyzer agent. The Analyzer sends download requests of Web resources to the Surfer during the training and execution of extraction tasks.

The Surfer can be seen as a web client used by the system. It provides similar features such as a Web cache to increase the performance. Multiple Surfer agents can be alive simultaneously to decrease the download time, and each agent can also handle multiple download requests simultaneously.

4.3.3 The Analyzer Agent

The Analyzer agent is the brain of the system. The Butler agent can for example request start of training of a new extraction task to the Analyzer or start an extraction for an existing task. Multiple instances of the Analyzer agent can be executed simultaneously to increase the performance of the system.

The learning of a new extraction task is the process that is of most interest in the Analyzer agent. After a request has been received from the Butler agent with information such as the starting address and extraction patterns, the Analyzer initiates a new MDP (Markov Decision Process). The MDP represents the tree where the structural information will be stored. Initially, the tree only contains the given starting point and the Analyzer sends a request to the Surfer agent to start downloading the starting web page. When the Analyzer receives the page content from the Surfer, the Analyzer translates the page from the character level representation to a hierarchical parse tree representation according to the hypertext model described in chapter 3.3.

The Analyzer then examines the top nodes of the tree and makes a decision as to whether the child nodes should be traversed. Currently, "starting explore" is used when the process is initiated and therefore all child nodes will be examined during the training phase. This behavior is not suitable for complex extraction tasks and will change in future versions of the system. The $Q(\lambda)$ algorithm will subsequently be used to update the decision policy and find the optimal path to the extraction points. A reward function evaluates each node in the tree and returns an appropriate reward. For example, the reward function can return one if the current node contains matching extraction pattern and zero otherwise. The experiment described in chapter 5 also uses the rule that any sibling node to a matching extraction node should receive reward one as well as certain penalties for each download of a new page.

The process is finished when all nodes have been examined and the maximum page depth has been reached. An initial update to the decision policy has now been performed. The process is then repeated to reach a stable decision policy, according to the standard general policy iteration idea. About 10-20 repetitions may be necessary. Further information about the learning algorithm used in the training process is given in chapter 4.4.

4.4 The Learning Algorithm

One advantage of using reinforcement learning algorithms for the navigation task is that actions are assigned a utility based on the future result, i.e. actions that give no immediate benefit may still be considered important due to possible future states. The utility is measured in the *total return*, i.e. the estimated discounted sum of all future actions, instead of only the immediate reward. The goal of reinforcement learning is to obtain a policy that maps states to actions, $\pi : S \rightarrow A$. The total return, or the *value* of a specific state is defined as seen in equation 4.1 where γ

$\forall x, a\ [\hat{Q}(x, a) \leftarrow 0]$
$\forall x, a\ [Tr(x, a) \leftarrow 0]$
while (true) {
  $x_t \leftarrow$ the current state
  $a_t \leftarrow \arg\max_a \hat{Q}(x_t, a)$
  execute action a_t
  $r_t \leftarrow$ reward for action a_t
  $e'_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{Q}_t(x_t, a_t)$
  $e_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$
  $\forall x, a\ [Tr(x, a) \leftarrow \gamma \lambda Tr(x, a)]$
  $\forall x, a\ [\hat{Q}_{t+1}(x, a) \leftarrow \hat{Q}_t(x, a) + \alpha Tr(x, a) e_t]$
  $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_{t+1}(x, a) + \alpha e'_t$
  $Tr(x_t, a_t) = Tr(x_t, a_t) + 1$
}

Figure 4.4: The $Q(\lambda)$ Algorithm [7]

is the discount factor, r is the reward at time step t after given state s, and V^π is the optimal value function. This relates to the navigation task since it consists of a sequence of decisions to stop or continue through a specific path.

$$V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{4.1}$$

The algorithm used for the experiment is based on the $Q(\lambda)$-learning algorithm [7]. This algorithm combines the advantages of Q-learning and $TD(\lambda)$-learning. Q-learning is an off-policy method that allows non-greedy (exploratory) actions to be selected without losing the ability to obtain an optimal policy. $TD(\lambda)$-learning is a combination of Monte Carlo simulations and dynamic programming. Similar to Monte Carlo, it can learn a policy without complete knowledge of the environment dynamics, i.e. all states, actions and associated transitions. TD-learning also inherits the benefits of dynamic programming where the policy is updated directly after each time-step, as opposed to the end of the episode. The addition of *eligibility traces* assigns credit or blame to visited states and actions according to the λ factor. This will increase the learning speed and faster reach the optimal policy. Further details about these algorithms can be found in the textbook by Sutton and Barto [54].

An outline of the algorithm can be seen in figure 4.4. The $\hat{Q}(x, a)$ function represents the estimated total return for state-action pair (x, a) and $Tr(x, a)$ represents the eligibility trace value for (x, a). The $V(x, a)$ function returns the maximum Q value for given state, i.e. $V(x) = \arg\max_a Q(x, a)$. A common approximation that is also used in this system is to cut off the eligibility trace when the $Tr(x, a)$ value subsides beneath a given threshold, thus making it unnecessary to iterate through all state-action pairs.

4.4.1 Random Walk Experiment

A simple experiment was conducted to examine how the parameters of the $Q(\lambda)$ algorithm affect the estimated total return values. The Random Walk Markov process was used for the experiment. It consists of 20 states $S = s_1, ..., s_{20}$ and the only transitions that are allowed from state s_n are to state s_{n-1} or s_{n+1}. State s_1 and s_{20} are terminal states in the process. The goal is to reach state s_{20} and thus the learning algorithm shall find a decision policy that moves from the starting state s_{10} to terminal state s_{20}. Figure 4.5 shows the transitions allowed in the process.

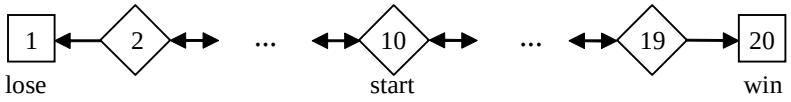

Figure 4.5: Random Walk Markov Process

For each action taken in the process, a reward of one is returned if state s_{20} is reached and zero otherwise, i.e. only when moving from state s_{19} to s_{20} is reward one returned. Due to its simplicity, the Random Walk process is commonly used when comparing or evaluating reinforcement learning algorithms. For the $Q(\lambda)$ algorithm used in this experiment, it is easy to calculate the optimal Q^* value for each state-action pair. As seen in equation 4.2, the Q^* value should be zero for all actions that move to the left and near one for actions that move to the right depending on the γ parameter.

$$Q^*(state, action) = \begin{cases} 0 & | \, action = left \\ \gamma^{19-state} & | \, action = right \end{cases} \tag{4.2}$$

For each experiment, the Random Walk process was executed through 60 episodes. Executing any further episodes does not improve the policy for most settings of the parameters. Table 4.1 lists the experiments that were performed on the Random Walk process and the setting of the parameters. For example, experiment 1 uses $\alpha = 0.5, \gamma = 0.5, \lambda = 0.5$ and varies ϵ from 0 to 1.0 in step of 0.05[1].

The ϵ (epsilon) parameter determines the probability that a random action will be selected, instead of the greedy action. If set to one, only greedy actions will be selected; thus, the policy will only be exploited and not explored. If the parameter is set to zero, a random action will always be chosen; thus, all paths of the tree will be explored but the behavior will be completely random.

Figure 4.6 shows the Q values of the 19 states [2] and how they depend on the value of the epsilon parameter for experiment 1. As seen in the figure, the optimal

[1] 0:1.0:0.05 is Matlab style for creating a set of values from 0 to 1 in steps of 0.05

[2] 19 states was actually used in the experiment instead of 20 states to find a starting state that is placed in the middle of the process

Table 4.1: Random Walk Experiments

num	ϵ (epsilon)	α (alpha)	γ (gamma)	λ (lambda)	# episodes
1	0:1.0:0.05	0.5	0.5	0.5	60
2	0.5	0.05:0.95:0.05	0.5	0.5	60
3	0.5	0.1:0.9:0.1	0.5	0.5	10:100:10
4	0.5	0:1.0:0.05	0.5	0.5	0
5	0.5	0.5	0:1.0:0.05	0.5	60
6	0.5	0.5	0.5	0:1.0:0.05	60

Q^* value of one for state-action pair $(17, right)$[3] is reached for all settings of the epsilon parameter after 60 episodes.

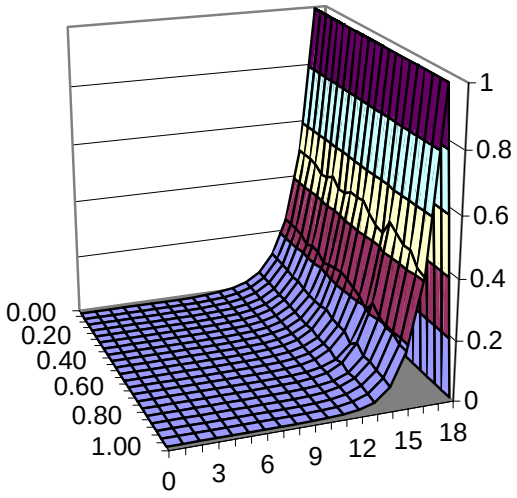

Figure 4.6: The State and Epsilon Parameter

It even finds the optimal policy for $\epsilon = 0$ when only the greedy action will be chosen, but that depends on the fact that the default action is to go right when both go left and right have the same Q value. The optimal policy would not have been found if the default action had been to go left. A more suitable implementation would have been to choose a random action when two actions have the same Q value, since the optimal policy would then always be found given sufficient time.

Figure 4.7 shows the Q value for state-action pair $(16, right)$ for experiment 1. A slight trend can be seen in the figure that the optimal Q value is not found after 60 episodes when ϵ approaches 1. Thus, for this process, it will reach the optimal policy faster if the current policy is exploited and not only explored. In fact, it could even be concluded from the figure that the best value of ϵ is zero, i.e. no random actions. Even if that may be true for this process, it will not necessarily be true for more complex and dynamic processes. As seen in later experiments that

[3]The states are numbered 0-18; thus $(17, right)$ corresponds to state 18 and action right

perform actual navigation tasks, too little exploration may cause the process to be stuck in wrong final states and never to find the appropriate states.

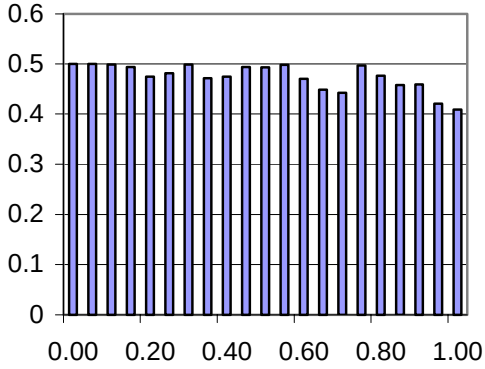

Figure 4.7: The Epsilon Parameter

Experiment 2 evaluates the learning rate parameter α (alpha). Figure 4.8 shows the Q value for state-action pair $(16, right)$ for different α settings. After 60 episodes, the optimal Q value of 0.5 is found when α approaches one. When $alpha = 1$, the complete error between the last reward the current estimate of the reward for the state-action pair is used in policy updates. For this simple and static process, a high learning rate works very well. For more complex and dynamic processes, setting α too high may cause an unstable policy and it may take a longer time to find the optimal policy than with a lower learning rate.

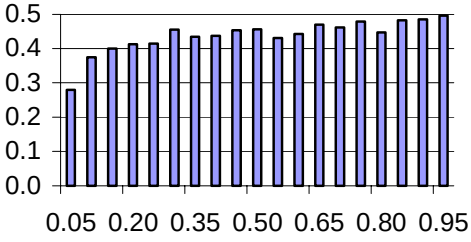

Figure 4.8: The Alpha Parameter

Experiment 3 evaluates the correlation between the number of episodes executed and the learning rate parameter. As expected, the optimal Q value can be reached even with a low learning rate if a sufficient number of episodes are executed, or with a low number of episodes and high learning rate as seen in figure 4.9.

Experiment 4 evaluates the γ (gamma) parameter. The γ parameter determines how much previously executed state-action pairs shall be blamed or credited for current reward. Setting $\gamma = 1$ results in a full update of all previous state-action pairs for each error. Figure 4.10 shows that the optimal Q value for state-action pair $(16, right)$ is found for each setting of γ. The γ parameter does not change

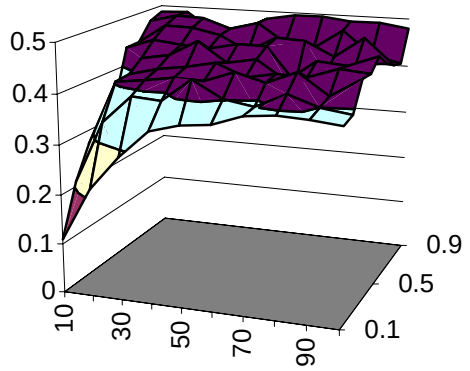

Figure 4.9: The Alpha Parameter and Number of Episodes

how the policy is updated, rather it changes what the optimal Q value should be. As seen in equation 4.2, the optimal Q value for state-action pair $(16, right)$ is γ^1 (remember that only 19 states were used in the experiment and they are numbered 0-18). If $\gamma = 1$ then the Q value would propagate all the way back to state 1; thus the optimal policy would contain value 1 for action $right$ and value 0 for action $left$ for all states in the process. Even though this may be appropriate for this process, a more complex process with different rewards in several state-action pairs could suffer severely from a too high γ setting. This is a difficult parameter to set, it may sometimes be appropriate to populate the error far back through previous state-action pairs and sometimes it may interfere with neighbor paths that lead to wanted states. This is especially true when not selecting the greedy action, i.e. when exploring the process.

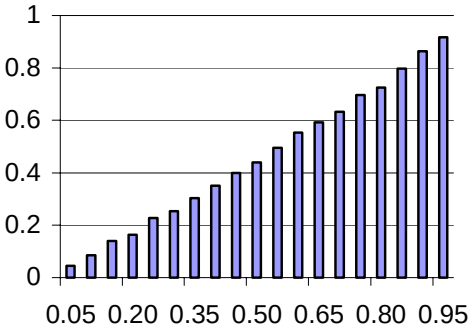

Figure 4.10: The Gamma Parameter

The final parameter that was evaluated was the λ parameter. It determines how fast the error should be distributed through the eligibility trace. The eligibility trace propagates the error back through previous states for each policy update. Without the trace, it would take a longer time to find the optimal policy. As with the γ parameter, setting λ too high may cause interference with neighborhood states in the

process. The Watkins version of $Q(\lambda)$ [55] cuts off the trace for each state that performs an exploratory action. Peng's $Q(\lambda)$ algorithm treats exploratory actions the same as greedy actions and most studies have shown that it performs significantly better than Watkins' algorithm [54].

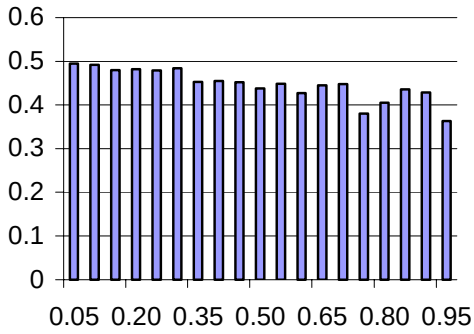

Figure 4.11: The Lambda Parameter

As seen in figure 4.11, the optimal Q value for state $(16, right)$ is reached after 60 episodes for low settings of λ. Higher settings cause later inappropriate states, mainly through exploratory actions, to interfere with the Q value. Since 60 episodes were used in the experiment, the optimal policy was reached even with low settings of the λ parameter. If the process were bigger or a fewer number of episodes were executed, the optimal policy would not be reached for low settings of the parameter.

4.4.2 Learning Algorithm Extensions

As indicated in the evaluation chapter of this thesis, the algorithm needs to be extended if the extraction task shall reach satisfactory levels of performance.

Firstly, the eligibility trace is cut off for non-greedy actions. The update of the $Tr(x, a)$ value is therefore replaced as shown in equation 5.4. Secondly, the error values that are used to update the $\hat{Q}(x, a)$ value are changed so that it becomes dependent on the state-action pair that it shall update. Therefore, it needs to be calculated for each node in the trace according to equation 5.5. More information about their extensions can be found in chapter 5.3.

4.4.3 Algorithm Complexity

The space requirements of this algorithm depend mainly on the number of state-action pairs that exist in the process. The $\hat{Q}(x, a)$ needs to store the estimated total return for each pair in the process. The $Tr(x, a)$ does not need to save a trace value for each pair in the process, the trace can be cut off for value below a given threshold and a value of zero can be assumed for pairs that do not exist in the trace array. Thus, the space complexity is linear in the number of pairs in the process.

The most time expensive operation in the algorithm is the update of the $\hat{Q}(x, a)$ values. The Q value for each state-action pair needs to be updated for each action executed by the agent. Additionally, the extended version of the algorithm needs to calculate the error for each such update. The time complexity is therefore $O(nm)$ where n is the number of pairs in the process and m the average number of actions per state.

Chapter 5

Evaluation

The approach of the extraction task described in chapter 3 was evaluated with the ASIE system described in chapter 4. This chapter describes how the experiment was setup and the results of the experiment.

The choice of experiment is motivated and explained in section 5.1. Details about experiment setup and algorithm parameters are given in section 5.2. The results of the experiment and comments about them are given in section 5.3.

5.1 Experiment Overview

A simple information extraction task was selected to evaluate the navigation step using reinforcement learning techniques. The experiment that was selected was to extract the set of undergraduate programs available in the department web site of the University [56] of Kalmar. This is not a complex task, in terms of linguistic or semantic knowledge requirements. This is appropriate in this case since the use of structural information is in focus.

The goal of the Analyzer agent is to find the optimal decision policy to be able to navigate quickly to the extraction points in the tree. For the chosen extraction task, about 20 pages need to be downloaded and about 1000 nodes are present in the parse tree. Figure 5.1 shows target text pieces that should be extracted in the final page for this experiment. The given start page for the extraction task is the start page for the department and the final page can be found within two links from that start page. However, about 20 pages needed to be downloaded to find that page for this task.

To obtain a user-driven system, as little work and expertise as possible should be required of the user to complete this task. The input from the user for this task consists of a starting URL and a small set of example extraction patterns. It was actually only necessary to have one simple pattern to complete this task, but input that is more complex may of course be necessary to complete more difficult extraction tasks.

An important part of the navigation step is how the rewards are calculated for

Figure 5.1: Extraction Targets (text inside shaded box)

the actions performed in given environment. Since the extraction task has the goal of finding the information pieces to extract, the rewards of the state-action pairs are high if they lead to states where information to extract exists and low otherwise, e.g. the reward can be set to one if a relevant information point has been found and zero otherwise. Additional assignments of the reward should also be used to guide the agent in the right direction and avoid resource and time-consuming paths.

The selected extraction task can be easily completed with little user input. The reason that such a simple task was selected was that the main objective with the experiment is to evaluate the learning algorithm used for the navigation task. For more advanced extraction tasks, additional user input, feedback as well as intelligent calculation of the reward function will be necessary. The level of intelligence implemented in the reward function is crucial to maintain a user-driven extraction system for more advanced extraction tasks and will be the focus of future research.

5.2 Experiment Setup

Table 5.1 list the input given for the experiment. The start URL and maximum page depth represent the obvious start address for the extraction task and maximum consecutive page links. The given pattern states that the agent should look for a node that contains the text “120p” and extract the all the content of those nodes. The pattern also states that any siblings that exist to a matching node should also be extracted. This type of rule is common in semi-structured extraction tasks since the wanted information is often placed in tables or bullet lists as in this case. It reduces the burden of creating necessary and sufficient textual patterns.

Table 5.1: User Input for the Navigation Experiment

Start URL	http://www.te.hik.se/
Page Depth	3
Pattern 1	Contain “120p” + siblings

The reward function for this experiment is defined according to equation 5.1. When a desired extraction node s' is located after executing action a in state s, the positive value is returned by the reward function and subsequently distributed back through the process according to the learning algorithm. A negative reward is given when a hyperlink is followed since the jump to another page results in a subsequent resource demanding page download. If another path to the extraction node can be found without making as many downloads, then that path should be preferred.

The specific values of zero and one that have been chosen correspond to standard reward values for reinforcement learning algorithms. Other values could have been chosen as long as desired states received higher rewards.

$$r(s,a) = \begin{cases} 1 & | \ s' \text{ is an extraction node} \\ -0.1 & | \ s \text{ is a hyperlink} \\ 0 & | \ \text{otherwise} \end{cases} \tag{5.1}$$

The Q values for the decision policy are initiated to one according to equation 5.2, i.e. the estimated total return for all state-action pairs equals one in the beginning of the process. This is called *starting explore* and will make the process explore all the nodes in the process due to the high initial Q values. The policy will subsequently approach the optimal policy Q^*.

$$\forall s, a \ \hat{Q}(s,a) = 1 \tag{5.2}$$

The parameters for the $Q(\lambda)$ algorithm in this experiment are $\lambda = 0.9, \gamma = 0.99, \alpha = 0.5$. The motivation for the high gamma (γ) parameter, i.e. the discount factor, is that the process is large and not very dynamic and it is important for this experiment to make the correct decision in the top nodes to reach the extraction nodes. The lambda (λ) parameter influences the decay rate for the eligibility trace and can make the process converge faster but also increases the computational demands on the process. It also represents how significant the rewards in future states are and they are highly important for this experiment. The alpha (α) represents the learning rate and influences how much of the error that should be adjusted for each step. Too high a value may cause the policy to fluctuate thereby increasing convergence time and too low may cause increased convergence time.

5.3 Results

The agent was able to navigate successfully to the desired extraction points after executing a number of episodes. The interesting result of this simple task is how the update of the policy worked. It should be noted that for more complicated extraction tasks, the use of only the semi-structural information as currently implemented would not be sufficient. However, the idea is that it will provide a foundation for an extraction system that can be extended with additional techniques.

An example of how the Q values are updated during the policy iteration can be seen in figure 5.2. The two series in the graph represent two FRAME HTML elements near the top of the tree. These two nodes are especially important because they are far from the final nodes in a tree that consists of about 1000 nodes. The Q value of these two nodes will therefore receive a lot of influence from nodes further down the tree. The frame named FRAME2 leads to a set of desired extraction points and the other does not.

As seen on the right, on the final episodes, the policy is stabilizing and approaching the optimal decision policy. The optimal Q value for $FRAME2$ can be calculated as the discounted sum of rewards from the frame state and forward according to equation 5.3. The optimal value for the $FRAME2$ state with the rewards in the given experiment is $-0.1 + -0.1 \cdot 0.99^6 + 1 \cdot 0.99^{16} + -0.1 \cdot$

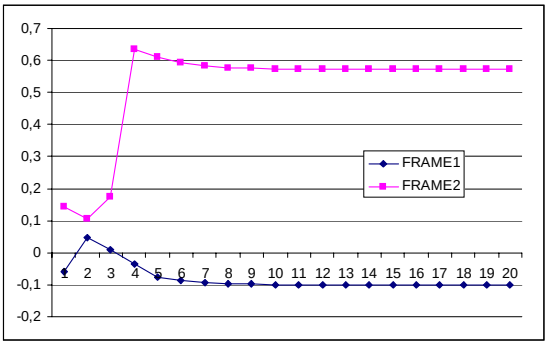

Figure 5.2: Q Values of Two Top Frame Nodes

$0.99^{17} \approx 0,573$. This value is obtained within three significant figures after about 10 episodes as seen in figure 5.2. The first -0.1 reward represents the link in the frame element itself, the second and last -0.1 reward represents anchor element links in the tree, and the 1 reward represents a matching extraction node.

$$Q^{\pi}(s_t, a_t) = \sum_{n=0}^{\infty} \gamma^n R(s_{t+n}, a_{t+n}) \tag{5.3}$$

5.3.1 Local Optima Problem

As seen in the first episodes in figure 5.2, there is a dip in episode two where the Q values do not approach the optimal value. The reason for this dip is that the rewards from the extraction nodes have not reached the top nodes yet. The greedy action in the state that represents the FRAME's parent node is therefore incorrectly set to FRAME1 node. Fortunately, the rewards will have propagated all the way back to the top nodes in the tree after a few episodes and the decision policy will converge to the optimal policy.

If the task had been more complex, the extraction nodes may not have been found and the agent could have been stuck in local optima due to a faulty decision in the beginning of the process. Of course, in an environment as complex as the Internet, it is impossible to guarantee that the wanted nodes will be found. If the environment is restricted in some way, for example by limiting the page depth as done in this experiment, the policy will easier approach the optimal policy. The convergence of Peng's $Q(\lambda)$ has not been proven yet; however, most empirical studies have shown it performs better than other off-policy algorithms [54].

The problem of local optima can also be reduced by modifying the parameters of the algorithm. By increasing the γ factor that regulates how far the error should be propagated back through the process, the rewards of states far away will have a higher influence thus reducing the risk of local optima. However, too high γ values will result in other problems such as top nodes being too highly blamed for errors even though some other wanted child nodes exist. This problem indicates

the need for more intelligent error propagation in the process. This topic needs further investigation as indicated in the future work chapter 8.5.

5.3.2 Non-greedy Action Problem

The first runs of the experiment did not result in a stable policy that approaches the optimal policy as shown in figure 5.2. In fact, some state-action pairs receive Q values of about 200, which is far from the optimal Q value that usually resides between zero and one.

A problem with the Peng $Q(\lambda)$ algorithm is that λ is not set to zero when non-greedy actions are chosen, i.e. the eligibility trace is not cut off for non-greedy actions. This could possibly increase the convergence speed [7, 57]; however, this is not the case for this process.

Since a high γ value had been chosen, the tree paths that did not lead to an extraction point significantly reduced the Q values of the top nodes. That caused the process never to converge to the optimal policy. In particular the top nodes will have Q values far from the optimal values. The trace factor that regulates how the error is distributed through the eligibility trace was therefore modified according to equation 5.4 to cut off the eligibility trace for non-greedy actions, similar to the Watkins' algorithm [58]. The n in the equation represents the number of time steps back through the process. Thus, errors are only propagated back through the last greedy actions in the eligibility trace. The parent of a non-greedy action is therefore not blamed for any subsequent errors. This solution significantly increases the performance of the algorithm; however, a more intelligent propagation of the error is still desirable.

$$t(s_{t-n}, a_{t-n}) = \begin{cases} (\lambda\gamma)^n & \mid \neg\exists nongreedy(a_m) \wedge 0 < m < n \\ 0 & otherwise \end{cases} \tag{5.4}$$

5.3.3 Penalty Accumulation Problem

Another problem with the $Q(\lambda)$ algorithm is that the top nodes accumulate penalties from errors in the nodes further down the eligibility trace. The error is calculated for the bottom nodes and that error is then distributed back through the trace according to the standard $Q(\lambda)$ algorithm. The problem is that the top nodes will receive a huge amount of penalty if the process is large and a high gamma value is used in the algorithm. For this experiment, *starting explore* was used to force the process to explore a large number of nodes in the beginning and become more exploitive when the policy becomes stable, i.e. the Q values are initiated to value 1. The error for the first episodes will consequently be quite large and negative since the value one is far from the optimal value. The top nodes could therefore reach Q values down to -200 after the first episode, which is of course far from the optimal value. Even if the policy does converge given enough episodes, it will take an unnecessarily long time to find the optimal value.

A solution to this problem is to calculate the error for each node in the trace instead of distributing the error for the bottom node. The rewards for each node in the trace, i.e. state and actions, are known since all of these actions have previously been executed. This information should be used to update the Q value more precisely. The error function for updates of Q values in the trace was therefore modified according to equation 5.5. The first term represents the discounted sum of rewards from the given state-action pair in the trace (t_1) to the end of the trace (t_2). The second term represents the estimated total return after the trace, and the final term represents the current total return estimate of the state-action pair that shall be updated.

$$\begin{aligned} e(s_{t_1}, a_{t_1}) = & \sum_{n=t_1}^{t_2} \gamma^{n-t_1} R(s_n, a_n) + \\ & \gamma^{t_2+1-t_1} \hat{Q}^*(s_{t_2+1}, a_{t_2+1}) - \hat{Q}^*(s_{t_1}, a_{t_1}) \end{aligned} \tag{5.5}$$

This modification increases the performance of the algorithm. The Q values of the top nodes will now stay close to the optimal value even in the first episodes, as seen in figure 5.2.

Chapter 6

Related Work

The information extraction area is quite new; it has existed for about seven years. However, a variety of types of systems and extraction tasks already exist. This chapter describes and discusses a set of systems and work related to this thesis. These systems will be evaluated mainly from two aspects: (1) the amount and type of knowledge used, and (2) the level of adaptiveness of the system.

As stated previously, the amount and type of knowledge used in semi-structural systems is usually quite different from natural language IE systems. More linguistic and semantic knowledge are typically used in natural language systems, whereas pattern matching and structural information are used in semi-structural systems. The following classification of knowledge will be used for the systems described in this chapter.

Pattern Matching Use of syntactic character-level pattern matching, e.g. regular expressions and named-entity identification

Structure Matching Use of structural information in the process, e.g. use of HTML structure and relationships between the nodes

Linguistic Knowledge Use of linguistic related heuristics, e.g. part-of-speech taggers and grammatical analysis

Semantic Knowledge Use of semantic/conceptual processing, e.g. use of shallow semantic resources such as WordNet

A very important aspect of an IE system is how easy it is to adapt to new domains. The level of adaptiveness is therefore especially evaluated for the systems described in this chapter. The *development* and adaptability of IE systems can be considered in the three different approaches described below. The same classification is described in the introduction of the thesis, but it is repeated here for clarity. The knowledge engineering approach is the traditional way to construct IE systems and the user-driven approach is novel and still mostly a concept.

Knowledge Engineering This approach requires a domain expert who is able to add extraction rules to the system. The difficulty of finding domain experts that also have sufficient computer knowledge makes it hard to adapt to new domains.

Automatic Training The use of machine learning techniques can relax the requirement of computer knowledge for the domain expert and thus only require domain experts that can annotate documents for the domain. The system can automatically be trained given the annotated documents. This makes the job of adapting an IE system to a new domain easier, but it still requires a significant amount of work to construct the training data for the system.

User-Driven IE UDIE differs from automatic training in that novice users rather than domain experts should be able construct an IE system without need of large training sets. This makes it easy to adapt to new domains, although more intelligence is required for the system and higher requirements of the domain. For example, it may be necessary to have semi-structural documents instead of free natural language texts.

6.1 Existing Semi-structural IE Systems

This section will briefly describe a number of IE systems that are able to process semi-structural texts. Some of these systems are pure *wrappers* and others are combinations of natural language and semi-structural IE systems, e.g. the `Webfoot` and `CRYSTAL` systems.

6.1.1 Ashish and Knoblock's Wrapper Generation Toolkit

The main idea in the system described in the paper by Ashish and Knoblock [59] is to exploit the semi-structural information present in Web pages to facilitate the extraction process. The construction of a wrapper starts with *identifying the relevant structure* of a page, building a parser based on given structure, and finally adding communication capabilities to the wrapper to be able to find different sources of information and give the result to a mediator.

A set of heuristic rules are used to identify sections and subsections in the web pages. These rules are basically regular expressions that exploit HTML knowledge to find the structure. In addition, heuristics such as font size are used to determine the hierarchical level of the structure.

There is no training in the system, although the user is able to correct erroneous guesses through a graphical user interface. The heuristics basically employs pattern matching rules to identify sections and subsections, with assistance of HTML knowledge. The actual structural relationships present in the source pages are not used for the identified output structure.

6.1.2 Rapper: A Wrapper Generator with Linguistic Knowledge

The approach used in the `Rapper` system [60] applies linguistic techniques to the wrapper generation task for semi-structured documents. The use of regular-expression pattern matching rules is insufficient to handle changes in word use and the order of the words. The addition of linguistic knowledge should increase performance and make the development of pattern-matching expressions easier.

The parsing starts with a preprocessing phase, syntactic analysis, domain independent rules, domain specific rules, co reference, and finally tag clean-up. The algorithm first makes a shallow parse and checks if it is enough. If more processing is required, it continues with deeper parsing. The preprocessing and syntactic analysis phases can for example add part-of-speech tagging to the text. The domain independent rules include techniques such as number rules that recognize and provide a standard form of numbers (e.g. three thousand becomes 3000, 3K, etc). Other rules such as date-time rules, units of measure, and named entities (names for people, organizations, and locations) are also included. Domain-specific rules exploit knowledge about a specific application domain to do deeper parsing, for example find the number to the left of the word personnel, and give it the label personnel-strength.

The system uses the same techniques as in the Ashish system [59] and adds algorithms that employ linguistic knowledge. These extensions increase the cost of adapting the system to new domains, although they increase the accuracy for implemented domains. As stated in the paper, the construction of wrappers is a non-trivial task even with these tools. A significant amount of knowledge is still required to construct a wrapper.

6.1.3 Wrappers in the TSIMMIS System

Hammer et al have developed a *wrapper implementation toolkit* to make the development of wrappers more efficient [61]. The toolkit consists of a set of templates that contain common types of answers. Application developers can translate and match queries against these templates and swiftly be able to extract information if a suitable set of existing templates are found. A post-processing engine is also part of the toolkit that is able to translate the answer to the desired format for the application.

A declarative, rule-based query language called `MSL` is used in the `TSIMMIS` project [62]. The data is represented according to a model called OEM that is similar to wrapper templates. MSL queries sent to the system extract OEM objects that match the patterns in the query.

The system relies on a predefined set of templates that are matched to the query and used to extract relevant information. This makes the development easier given that predefined templates exist. The adaptation to new domains involves construction of new templates, which can be a non-trivial task. The possibility to integrate the new templates into the existing system makes it viable to obtain a larger

more domain-independent solution. The templates basically seem to rely on pattern matching rules.

6.1.4 The Webfoot preprocessor

It is difficult to combine the benefits of semi-structured oriented IE systems and NLP oriented systems. Soderland tries to solve the problem by preprocessing the documents with the Webfoot preprocessor [4] and then passing the output to an NLP system named CRYSTAL [51].

As stated in the paper, it is important to include the structural relationships and not only the facts from the documents. For example, the relationships between the day, location, and weather conditions provide crucial information to be able to answer advanced queries. The Webfoot system transforms the document using structural information from the HTML layout to a sequence of segments. The goal is that relevant facts should be grouped together into a single segment. Each segment is further divided into fields using a set of delimiters, both domain-independent and domain-dependent.

The sequence of segments produced by the Webfoot system is passed on as input to the CRYSTAL NLP system that learns text extraction rules from examples. A domain-dependent lexicon is also included in the input together with the sequence of segments from Webfoot.

The Webfoot preprocessor relies on a set of predefined heuristic rules, thus no training is involved in this first stage. The CRYSTAL system employs a supervised learning that requires texts that have been manually annotated to match terms in the documents to concepts in the system. In addition, a lexicon has to be constructed for the domains of interest. Both linguistic knowledge and some semantic knowledge from the lexicons are used. The structure of the source document is used, although only one level hierarchy is constructed.

6.1.5 The ShopBot Comparison Shopping Agent

The goal of the ShopBot [50] system is to be able to compare computer product prices for different vendors. The user of the system should be able to enter product keywords and receive a sorted list of products from different vendors. The system should automatically be able to learn where search forms are located at each vendor and interpret the response from queries executed at each vendor.

The agent learns how to handle the search forms at a specific vendor during the “learning phase” of the system. After the learning phase, users of the system can send queries that will be executed for all vendors included in the system and in real-time see the result of the query. A set of heuristic rules are used to locate the correct search form at the vendor. The learning algorithm does not require any annotation of the documents, but it requires a “domain model” that contains a set of example products with their associated attributes. These examples are used together with HTML heuristics to induce the extraction rules for each vendor.

The system is heavily targeted to the vendor-product domain, although it would work for other product types than computer products. Adaptation to new vendors within the same product domain seems to work with very little effort. Adaptation to new types of products requires more work, e.g. construction of the domain model that requires domain expertise. There is practically no use of linguistic or semantic knowledge, with the possible exception of the domain model. The extraction rules basically rely on pattern matching and some minor structural matching.

6.1.6 The WYSIWYG Wrapper Factory (W4F)

The `W4F` toolkit [63] assists users in the construction of wrappers with a graphical user-interface. A set of wizards presents web pages that are annotated with extraction rules. The user can click on specific parts of the page to find extraction rules for that piece of text. These rules can be further generalized and modified by the user to construct a wrapper for a specific target site.

As written in the paper, the design of wrappers should be written in a layered architecture where the retrieval, extraction, and mapping of the output are carried out separately. This will increase the ability to adapt the wrappers to new domains and target sites.

The toolkit transforms the HTML pages into a parse tree according to the HTML specification. Instead of considering the HTML page as a string; it is viewed as a tree where each node corresponds to an HTML element. The advantage is that the structural information in the document can be used in the extraction rules, instead of just character-level patterns. The rules are written in a declarative language for each layer, i.e. the retrieval, extraction, and mapping layer. In addition, non-declarative user-defined mappings in the Java language are provided which makes it very easy to include the wrapper in Java applications.

The graphical user-interface makes it easy to construct wrappers for new domains, even though no automatic training is used in the toolkit. The user is still required to have knowledge about how the extraction language works and about for example regular expressions. The toolkit relies heavily on the structural information in the web pages and on some pattern matching.

6.1.7 Head-Left-Right-Tail (HLRT) and Related Wrappers

Six different classes of wrappers are evaluated in the paper by Kushmerick [5] with focus on their efficiency and expressiveness. A wrapper is formally defined as a function from a page to a label, i.e. the result of executing a wrapper on a page is a label. The simple `LR` wrapper class uses left- and right-hand delimiters to extract the relevant pieces of text. More than one attribute can of course be extracted, each having associated pairs of left- and right-hand delimiters.

The *wrapper induction* problem is defined as finding a wrapper of given class that maps the given set of example pages and associated labels. The labeling of the example pages used for the supervised training is often performed manually.

However, since the goal is to automate the wrapper construction, some initial experiments have also been performed on ways to label the examples automatically.

The other classes of wrappers presented in the paper extend the `LR` class in different ways. The `HLRT` is a useful class that identifies the delimiters for the head and tail of the page in addition to the left and right for each attribute, thus providing support for more sophisticated pages. The `OCRT` class differs from `HLRT` in that the opening and closing delimiter is identified for each group of attributes instead of for each page. The `N-LR` class extends the `LR` class to support nested attributes, i.e. name, street, and city can be sub-attributes of an address attribute. The `HLRT` and other classes could also be extended to support nested attributes, e.g. class `N-HLRT`.

One of the purposes of the paper is to compare the performance of the wrapper induction task for the different classes, thus automated training is used for the construction of the wrappers. There seems to be little more than pattern matching rules that are being induced by the learning algorithms presented.

6.2 Discussion

The systems selected for this paper by no means constitute a complete list of systems that can handle semi-structural text. They have mainly been selected to show different approaches for systems related to this thesis. This section will discuss the differences in the systems by looking at the type of knowledge that is used and the level of adaptiveness. Figure 6.1 gives an overview of the differences in the amount and type of knowledge used.

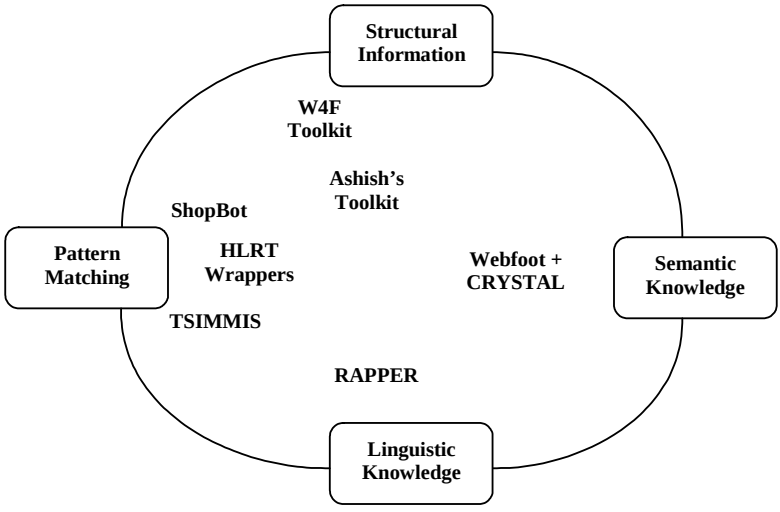

Figure 6.1: Amount and Type of Knowledge

The `ShopBot` system, the `HLRT` related wrappers, and the `TSIMMIS` system rely mostly on pattern matching capabilities and do not take significant advantage

of structural, linguistic, or semantic knowledge. Even though HTML heuristics are included in some of these systems, the web pages are basically viewed as a character-based representation rather than a hierarchal structure. The W4F toolkit makes more use of the structural information by transforming the pages into a parse tree that is used to navigate through the text. The Webfoot system together with the CRYSTAL system makes an ambitious attempt to take advantage of linguistic, semantic, as well as structural knowledge. The level of structural knowledge used is still not as rigorous as in the W4F toolkit.

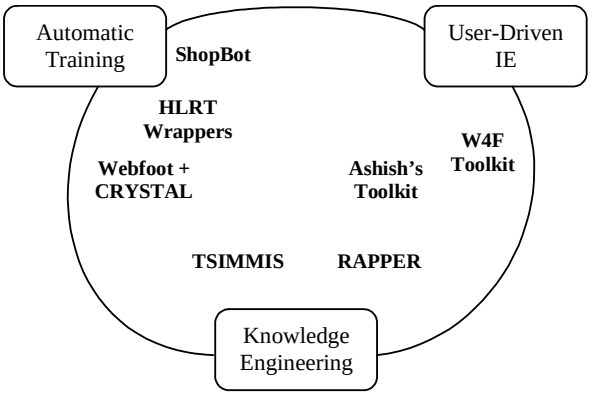

Figure 6.2: System Adaptiveness

An overview of the adaptiveness of the systems is shown in figure 6.2. The TSIMMIS and RAPPER systems rely on traditional knowledge engineering methods for adapting to new domains. The ShopBot and the HLRT related wrapper classes rely on automatic training, although some expert work may be needed to annotate documents or provide new domain knowledge. The W4F toolkit has no training and relies on a user/expert to create the extraction rules with valuable assistance from the graphical user interface. The optimal solution would be to be able to have intelligent training guided by novice users, i.e. a truly user-driven IE system. In addition, the current border between semi-structural IE and natural language IE should be removed in the future to let techniques from both areas complement each other.

Another system that is related to this thesis, even though it is not an information extraction system but rather a web spider, is the Cora project [52]. A Web spider is an agent, a.k.a. robot, that collects documents for an information retrieval system. The Cora system uses reinforcement learning techniques to choose efficiently which links to follow. Thus, a policy is trained for the hyperlink structure but the semi-structural information present in the document is not used since the goal is not to extract information. However, the use of reinforcement learning techniques to navigate through some structure is similar to the approach of the ASIE system.

Chapter 7

Ethical Considerations

Novel scientific knowledge and advances in technology give more power to the users of these tools. This power can not only be used but also abused for destructive and morally questionable goals. It is therefore important to consider the possible and probable consequences of these new technologies. The purpose of this chapter is to identify and discuss these consequences.

7.1 General Effects of Knowledge Management

A serious problem in the information society of today is the *information overflow* problem. A large amount of money is spent on knowledge management (KM) tools that are effective in producing large quantities of information. These tools are also efficient in distributing this information around an organization, even to users that are not interested in the information. When users receive too much irrelevant information, the relevant information will be lost. Thus, the goal of the KM tools to increase the effectiveness of the organization [8] is not achieved. In extreme cases, these tools can even decrease the effectiveness.

KM tools should produce information with high quantity and quality, i.e. as large an amount of information as possible that is relevant to the specific user and as small an amount of irrelevant information as possible. Tools used for applications such as data warehousing and data mining are able to produce large quantities of data, but unable to make a semantic interpretation of the data to judge whether or not the information is relevant. More tools that are intelligent should be used where a user can easily choose relevant information topics and that automatically adapt to the user's needs. A crude example is the comparison between information retrieval (IR) and information extraction (IE) services. IR services such as AltaVista and Google produce a large amount of information, including a large amount of irrelevant information. IE services extract only the relevant part of the documents and present a smaller amount of information but with higher quality. The problem with IE tools is that they are more difficult to build.

I hope that the type of service that is presented in this thesis will not only in-

crease the quantity of information, but also increase the quality and make it possible for users to obtain relevant and interesting information.

Another consideration is that of personal privacy. Privacy refers to the ability for an individual to control information about him- or herself, consistent with his and her expectations and values. Loss of privacy may result in ethical and social problems.

American law classifies privacy into the following four types: disclosure, intrusion, false light, and appropriation [64]. Disclosure happens when potentially harmful information about an individual is given to someone else without given consent. Intrusion occurs when the personal sphere of an individual has not been respected. False light is another form of invasion of privacy where false information is spread about an individual. It may seem questionable as to whether false light is really a privacy issue, since it does not reveal true privacy information. However, since the false information may cause serious harm to the individual, it is an ethical problem. Appropriation is the act of taking on the identity of others for purposes of one's own. An example of this type of invasion of privacy is carding, i.e. using other people's credit cards. It should be noted that these different types of invasion of privacy are not independent of each other. Some have even suggested that all privacy issues can be reduced into disclosure or intrusion.

The introduction of knowledge management tools brings these privacy issues into new situations. The magnitude of information that can be collected, analyzed, and distributed has increased significantly due to these computer-aided tools. For example, the use of data mining tools to identify zip codes for where to send advertisements can be seen as a positive example of how KM tools decrease unwanted advertisement. However, the same tools can also be used to refuse orders from certain low-income zip codes. This may increase the revenue for the company in question, but it may also increase segregation and cause increased harm to society. KM tools not only increase the effect of existing privacy problems, new KM applications introduce new privacy concerns in situations that were not possible before. An example of such an application is transaction generated information (TGI). Information can be gathered from various transactions that individuals perform such as credit card purchases and telephone calls. Advanced KM tools make it feasible to gather and analyze this huge amount of information. These tools can produce personal profiles that contain personal information and potentially cause privacy problems.

The popularity of the Internet has also significantly changed these privacy issues. The Internet is quite different from other existing forms of communication. In-Real-Life (IRL) communication can be categorized as "few-to-few" where only a small number of people can send and only a small number of people can receive the information. Newspapers, radio, and television are examples of "few-to-many" communication where a small number of people can broadcast information to a large number of people. The Internet makes it possible for a large number of people to send information to a large number of people, thus it is "many-to-many" communication [65].

It may not be obvious why this increased access to information and possibility to analyze and distribute information may be harmful. Some people say that only individuals who have something to hide will get hurt. One way to argue against this opinion is to compare it to the extreme example of a panopticon. A panopticon is a prison where the cells are arranged in a circle and the side facing the center of the circle is made of glass. A guard tower in the middle can observe everything that happens in the cells. Even though the prisoners will not be observed at all times, they do not know themselves when they are being watched. Since the prisoners think they are being observed, they will change their behavior as if someone were watching them. If all information about ourselves were accessible, such as credit card purchases and telephone calls, we would also change our behavior. This change of behavior can lead to loss of autonomy, fear of acting against the general opinion, and problems with the development of our democratic society [66].

To protect our privacy, the use of knowledge management tools and the information they produce needs to be regulated. Even the development of these tools may need to be regulated in extreme cases. How this should be regulated is a complex issue and not the main topic of this thesis. It is however important to be aware of these problems and what consequences the use of these tools can have.

An interesting example of how privacy can be protected by regulation is Swedish legislation. Sweden has been called one of the most open societies in the world. The so called "publicity principle" states that there should be free access to official documents and anyone who wants access has the right of anonymity and does not have to state his/her purpose. Sweden was also the first country in the world that enacted a Data Protection Law in 1973 that regulated the use of personal data files. This law was superseded in 1998 by PUL, an implementation of an EU directive that regulates use of personal information. The purpose of this directive was to adjust the legislation to the IT society of today. Unfortunately the directive was interpreted in an extreme way by the Swedish government so that it was not even allowed to print other people's names without given consent. It was no longer possible even to have a list of employees in the company homepage without first given explicit consent. Even though this certainly protects our privacy, it also prevents legitimate and non-harmful activities. The law was later changed to be less restrictive.

Another example of Swedish extreme protection of privacy was a recent court decision that held the publisher of a major Swedish newspaper responsible for content written in a public unmoderated web forum [67]. This decision resulted in the forum being shutdown and many people lost the ability to give their opinion and take advantage of the possibilities of the Internet. This is similar to the famous German case where the managing director of Compuserve was held responsible for pornographic material present in a Compuserve newsgroup [68].

New laws need not always be legislated to regulate the use of IT tools. In a personally related case, misleading articles that were written by the Swedish press caused a public Internet chat service at University of Kalmar to be shutdown[69]. A

DALnet IRC[1] server that I co-administrated was used by a group of Nazis. A major Swedish newspaper wrote a headline article where they said that the University of Kalmar supported Nazis, only because a few Nazis were using the server together with over 10.000 other users. This is a misleading article implying that University of Kalmar is pro-Nazi, and wrong from several aspects such as freedom of speech and democratic values. We should be careful not to be overly restrictive in the use of IT tools. It is important to protect privacy, but we should be careful not to restrict legitimate use as well. On a positive note, the server was later brought back online after long discussions with the head of the University.

A problem with the legislation of today is that it tries to define what should be allowed instead of what should be disallowed. This is called the *use-model* and the *misuse-model* respectively [70]. The conclusion from a Swedish investigation in 1993 found that the current use-model system with licenses and permissions did not function as intended. The immense work brought about by applications prohibits the Data Protection Agency from making the necessary inspections out in the field. Only about 10% of the available personal data files in the country were legal. A conclusion of the investigation was that only a Misuse-model was feasible in today's society, but the new EU directive was a perfect example of the use-model and the Swedish government felt obliged to implement it. The IT of tomorrow will provide even more intelligent and effective tools. We need to find effective ways to regulate the use of these tools if our privacy is to be protected.

7.2 Intellectual Property Rights

Knowledge management tools such as semi-structured IE make it easier to automatically extract relevant pieces of information from web sites around the Internet. This may tempt people to abuse these tools to take advantage of services and information in malevolent ways. The use (abuse) of these tools may also be illegal if they violate for example copyright restrictions placed on the source documents.

It may not be obvious why it is important and beneficial for the society to have copyright laws and patents. They are after all infringements of our rights of freedom of speech. It may seem that these laws only benefit the inventor or the company holding the patent at the expense of the public at large. However, if one considers that the invention or the information may not have been possible to create unless these laws existed; it is easy to see that they benefit the society in the end.

The following personally related incident is an example of the problem of balancing the copyright restriction and public benefits for a specific case. Microsoft has a knowledge base [71] that contains about 100.000 short articles that describe and give suggestions of how to handle problems with their products. The search

[1] An Internet Relay Chat (IRC) server provides a service where users can join and talk to each other in different channels to discuss various topics. DALnet is a global network of connected IRC servers that shares channels and users around the world.

engine that Microsoft provided unfortunately did not work very well[2]. Articles that should have been returned were not returned, i.e. the search engine had a low recall. Therefore, I personally created a service that collected the articles and provided my own search engine for the knowledge base.

It can be difficult to see who was hurt by this action. The users of Microsoft products will benefit from the service since they more easily solve problems that occur and Microsoft itself should increase their status among their customers. However, this was not the reaction of Microsoft when they became aware of the service. They threatened with legal action unless the service was shutdown due to copyright infringements. Even though I had asked permission and received an acceptance from the European Microsoft headquarters previously, I was forced to shutdown the service according to American headquarters.

The American headquarters was after all correct, it was an infringement of the copyright statement and it should therefore be shutdown. This is also an example of how knowledge management tools can be misused, even though it was for a beneficial purpose. Copyright laws restrict the rights to reproduce and publish information and patents prevent others from making, using, and selling inventions. This trade off between freedom and welfare is worth taking since it is beneficial in the end.

This also opens the question of how to uphold these laws. The same problem exists in the information retrieval area where some sites do not want to be indexed by any search engine. This resulted in a standard for how web sites should inform search engines (a.k.a robots, spiders) that they want to be excluded from the index [72]. A similar solution should be appropriate for information extraction (IE) tools as well. If such a standard became widely accepted, then it would be easier to prevent abuse of IE systems.

The copyright laws prevent people from republishing the same information at a different site. However, they do not explicitly prevent publishing of derived works, i.e. the result of analyzing and processing the source information. Some content providers may not want any information, even derived information, to be published elsewhere whereas other providers allow a certain amount of derived works. This problem of derived works also needs to be considered and regulated.

7.3 Ethical Theories

The main ethical theory that is used to judge what is right and wrong in this chapter is Mill's utilitarianism. Utilitarianism is a theory where an action is judged on the consequences of performing the action. The basic principle is that *everyone ought to act so as to bring about the greatest amount of happiness for the greatest number of people* [73]. The ultimate intrinsic good in the world is total happiness, i.e. not happiness of one specific individual. A common critique of utilitarianism is that it may seem to favor the decision to kill one individual to save ten other individuals,

[2]This occurred in 1998, it works better today

e.g. for organ transplants. However, if the long-term consequences of this action are considered then utilitarianism will not favor such an action. If it were allowed to kill one individual to save other individuals, then we would live in a society where most people would feel unsafe and the total happiness would be decreased.

Another question is whether the intentions of the individual should be considered or the actual consequences of the action. If the individual could not foresee the actual consequences of an action but had good intentions, then the individual still has high moral character. If we were judging the moral character of the individual, then it would be appropriate to consider the intentions of the individual. To put it in agent terms, the individual/agent can only maximize his *expected* utility. However, we are actually interested in whether it is morally correct to perform an action and therefore the consequences of the action should be considered and not the intentions of the individual or agent.

According to the theory of ethical relativism, the question of what is right and wrong might have different answers for different individuals. This means that there cannot exist any universal law or individual-independent morality, which makes the theory very impractical. It also focuses on the moral aspects of individuals rather than on actions. If we are interested in the moral aspects of an action, then the action should be dependent on the individual. It may however be dependent on the context of the action. If the action is later performed in the same context then the same moral value should apply.

Kant's deontological theory of ethics emphasizes the act itself rather than the consequences of performing the act. Some acts are never good no matter what the consequences are, for example killing another individual. It states that one should never treat humans as a means to an end, they are ends in themselves. This prevents the organ transplant problem previously explained since one cannot use one individual as a means to save other individuals. The categorical imperative states that one should only perform actions that you would like to be a universal law. Due to the complexity of our world, it is very difficult to find universal laws, e.g. Thou shalt not lie. Each action needs to be considered separately in the right context and it is difficult to find general context-independent laws.

Given that we have an ethical theory that we can use to evaluate actions, when is it appropriate to use it? Since our world evolves and the people who enact laws makes mistakes as we all do, it may be morally justified to break the law in some cases. Assume the following scenario. We have an IE system that is able to collect and process information from various sources. Let us also assume that it never publishes information from sites that have copyright restrictions. Would it be morally justified to collect and process information but not publish any information? Since the collection and processing actions do not hurt any other individual, what would be the harm?

Let us add some more information to the system, and assume that it has access to various Internet backbones and can gather a huge amount of information. The system is now able to analyze everyday traffic, e.g. transaction generated information, and create detailed profiles of specific individuals. This type of use of IE

raises various privacy concerns. Even though the information is not published, the bare knowledge that such a system exists may influence our behavior. The situation is similar to that of the panopticon that was described earlier in this chapter and leads to loss of autonomy.

Would this type of use of IE be morally justified if it could prevent another crime? It may be possible to use the system to detect carding, i.e. abuse of other people's credit cards. Would the appropriation abuse outweigh the loss of autonomy in the society? The loss of autonomy may seem to cause greater harm in this case, but would that still be the case if the system could prevent a terrorist attack? The terrorist attack may kill thousands of people and cause wide spread fear in the society. These difficult questions show some of the possible consequences that can be the result of use of knowledge management techniques.

Chapter 8

Conclusions

This chapter discusses the results of the work presented in this thesis. The problems and solutions of the experiments are initially discussed, followed by a discussion that is more general about information extraction and knowledge management. Additionally, the ethical considerations are discussed as well as limitations of the current approach and future work.

8.1 Intelligent Navigation

Training of the navigation task can be successfully accomplished using reinforcement learning techniques. One of the main advantages of reinforcement learning techniques, as opposed to supervised learning techniques, is the ability to learn from experience while interacting with the environment. Thus, the goal of a user-driven system, where the user is not required to prepare large training sets, is approached.

One of the problems identified in the experiment was that the agent might be stuck in local optima and not find the optimal path or the desired extraction points. This is not a problem if the environment can be restricted so that every node can be evaluated. However, it should be noted that the convergence of the $Q(\lambda)$ algorithm has not been proven. For tasks that are more complex where the environment can not be sufficiently restricted, the agent must choose which paths to investigate and leave other paths unexplored. The problem of local optima becomes more difficult for these tasks. These types of tasks require further investigation to show that they can be handled with satisfactory performance.

The $Q(\lambda)$ algorithm described by Peng that propagates errors through the eligibility trace from non-greedy actions proved to present some problems for this application. These problems involved how the non-greedy action error was propagated and how the error was calculated. The performance of the learning algorithm was significantly increased by cutting off the eligibility trace for non-greedy actions. Due to the high value of the discount factor chosen for this experiment, the γ (gamma) parameter, the iteration of the policy did not benefit from errors from

non-greedy actions. As stated in other papers [54], the use of non-greedy errors might very well benefit the training of the decision policy but this is not always the case.

Another problem experienced in the experiment was the problem of accumulating traces. Since the top nodes in the process received the same error, discounted by gamma, the estimation of the total return in the decision policy does not work well. The problem can be reduced by changing the way the error is calculated and re-calculating the error for each top node in the trace. This gives a better estimate of the total return and makes the policy converge faster to the optimal policy.

One of the goals behind this thesis was to be able to handle the dynamic nature of the Web. Since the approach proposed for the ASIE system involves high use of the structural information, it becomes sensitive to structural changes. A change in the structure can be easily detected by comparing the received total return with the estimated total return in the decision policy during the extraction mode of the system. If a change occurs after the training mode, the received total return will differ substantially from the estimated total return. A possible way to handle the change is to automatically switch to training mode again and try to find the similar extraction points again. The old patterns given by the user can be used, or possibly the content of the last successful extraction. The content extracted from the last successful extraction will probably still exist, but with a different structure. It should be possible to use this information to identify the new structure. However, it may be necessary to interact with the user in some cases to handle the structural change. This problem requires additional research.

8.2 Information Extraction

The area of semi-structural IE differs from natural language based IE in a number of ways, e.g. the source text is often not as grammatically correct or structured in the same way. Thus, techniques designed for natural language text may not work well for semi-structured text and vice versa. The performance of wrappers that work with semi-structured text is typically very high if the source text contains enough structural information that can be used for the extraction task. As stated in the introduction, the problem with these systems is how to adapt quickly to new domains and thus how to construct truly user-driven IE systems that can easily adapt to new domains. The usefulness of IE systems will not reach the same level of popularity as IR until they can handle a similar coverage of domains. For semi-structural information extraction systems, this can be possible in the near future. Natural language IE has already reached surprisingly high levels of accuracy, near human level performance for some specific tasks such as named entity recognition. However, the concept of user-driven IE systems for free text is still far away due to the demand for a large amount of linguistic and semantic knowledge.

The goal of user-driven information extraction systems says that minimal input from the user should be required, both in required time and in knowledge. The

navigation task is only one part of the complete information extraction system, but an important task that takes advantage of the semi-structural information. If the source documents contain a large amount of semi-structural information, no other processing may be required to complete the task [4].

The intuition behind the approach described in this thesis is that it should provide a base for future intelligent techniques and simulate human "surfing" behavior. The current implementation with the hypertext tree model and the learning algorithm with its reward function and error propagation only considers the syntactic and structural information. The performance of the system, and ability to handle more than just rigorous semi-structural domains, could benefit from additional use of linguistic and semantic knowledge. This will also be one of the topics for future research.

The current implementation, which mainly deals with semi-structural domains, can still provide several important applications. The Buyer's Guide shopping-agent system is one example. Another example that I have been involved in is Structure Knowledge Initiative - SKICal [74]. That initiative has the goal of increasing the knowledge management of public events. One task in that project is to collect and compile information from calendars that exist in different Web sites. This is an excellent example of the application of semi-structural information extraction. Other smaller applications that are of personal use are a TV-planner agent and a news agent. The TV-planner collects information from different TV program listings, learns personal preferences, and alerts the user when the show starts. The news agent simply extracts news headlines from about 10 news agencies and compiles a compact list of headlines with links to the articles. Several other similar applications can be envisioned and the need for these services will increase further as they become sufficiently effective.

The overall goal of the thesis is to improve techniques in the knowledge management area. The semi-structured information extraction technique is one part of the knowledge management area that can be of significant use in the near future. The amounts of information that exist in semi-structured environments have already reached high levels today. The work of the Semantic Web initiative, that has the goal to present information with sufficient semantics so that the information shall be machine-understandable, also requires effective information extraction techniques. With such information extraction systems and the use of standardized languages and ontologies, new services can be designed with machine-understandable semantic information.

8.3 Ethics

The purpose of the ethical chapter in this thesis was to consider possible consequences of fulfilling the goals of this thesis. The preferred ethical theory used to decide what is right and wrong is that of Mill's utilitarianism. It is the consequences of performing an act that should be evaluated and how these consequences

might hurt individuals in our world, both the short- and long-term consequences.

The first problem that was considered was that of information overflow. By increasing the performance of knowledge management tools, the flow of information might increase so that individuals get even more filled with information and unable to handle it. Firstly, knowledge management does not simply strive to increase the flow of information; the real goal is to increase the effectiveness of an organization/individual by better information handling. Thus, knowledge management tools should strive to provide knowledge and not simply information. The use of information extraction tools will certainly benefit the direct user of the system, but the question is if the user will later use that information to common good or not. For example, it might be possible that someone uses information extraction tools to collect email address information and then uses that information to send spam email. There are of course many other examples of benevolent use of the information as well. If the primary use of the information extraction technology was malevolent, then the development of such information extraction techniques should be seriously questioned. The number of applications that can be envisioned, both in the short and long-term, should however provide more benevolent than malevolent use. For example, applications such as shopping agents benefit users since they are able to find the best retailer and the competition among the retailers is increased.

Another problem is that of privacy. Information extraction tools can certainly be used in a malevolent way to collect personal information without given consent. If this abuse continued without improved legislation, the total happiness would decrease. Therefore, the question is whether it is probable that the legislation will be improved or whether the development of information extraction techniques is morally questionable. The best solution were if the legislation would be improved, e.g. by application of the misuse-model, since the benevolent use of information extraction techniques would be increased. Also, as indicated in the end of the chapter, we need to regulate the use of knowledge management tools and access to information.

The third problem deals with protection of intellectual property. As described in chapter 7.2, I have personal experience with how easy it is to get in to such problems. If it is easy to extract information from other services, it is easy to violate copyrights even if the intent is benevolent. If a standard for what is allowed to be extracted was developed, similar to the robot exclusion standard used for information retrieval services, the problem would be reduced. However, it would not stop users of information extraction systems who intend to violate copyrights. Again, the need for improved legislation is indicated.

8.4 Limitations

The approach proposed in this thesis is suitable for semi-structured information extraction tasks in hypertext environments. Natural language processing techniques should be used for other information extraction tasks such as extracting informa-

tion about different companies from news articles. The approach in this thesis is also dependent on high existence of structural information in the source documents. The desired information should preferably be present in lists or tables for the task to work satisfactorily.

It should also be noted that the current system is not a fully-fledged extraction system, but rather the foundation that additional components shall be built upon.

8.5 Future Work

The system described in this thesis is still in the beginning of the development cycle. Even though it can already handle simpler tasks, the goal is that it should be able to handle complex tasks and still maintain the user-driven property. The following list presents some topics and problems that need further work to reach that goal.

- Additional experiments should be performed to evaluate the system further in different environments and levels of complexity.
- The problem of dynamic structures need further investigation and experimentation. The detection and automatic retraining of the agent are two such topics that should be investigated.
- The navigation step should be given the ability to handle various types of structures that needs special handling. For example, it is common to have multi-page tables, i.e. lists that are split across several pages since they will not fit on a single page, and special consideration should be given to the structural information for these lists.
- The reward function should be improved to guide the agent intelligently through the hypertext environment. Linguistic and semantic techniques - could be included in this function to be able to give more input from the environment to the learning algorithm. Various such techniques exist today, and the addition of them to the system would make it more general and able to handle other domains.
- The error propagation should be further developed so that errors are more intelligently blamed through the eligibility trace. The current solution to simply cut off the trace for non-greedy actions and use a discounted distribution for greedy actions can be improved. This could reduce the problem of local optima and make the algorithm converge to the optimal policy more quickly.
- The users of the system should also be given additional ability to interact and give feedback to the system. This includes both human and machine users.

Bibliography

[1] N. Guarino, *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, vol. 1299 of *Lecture Notes in Artificial Intelligence*, ch. 8. Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration, pp. 139–170. Frascati: Springer, July 1997.

[2] H. Cunningham, Information Extraction: A User Guide (Revised Version), Report CS-99-07, Department of Computer Science, University of Sheffield, May 1999.

[3] J. Cowie and W. Lehnert, Information Extraction, *Communications of the ACM*, vol. 39, no. 1, pp. 80–91, 1996.

[4] S. Soderland, Learning to Extract Text-Based Information from the World Wide Web, in *Proceedings of the 3rd International Conference on Knwoledge Discovery and Data Mining (KDD-97)*, 1997.

[5] N. Kushmerick, Wrapper Induction: Efficiency and Expressiveness, *Artificial Intelligence*, vol. 118, pp. 15–68, 2000.

[6] Y. Wilks and R. Catizone, *Information Extraction: Towards Scalable, Adaptable Systems*, vol. 1714 of *Lecture Notes in Artificial Intelligence*, ch. Can We Make Information Extraction More Adaptive?, pp. 1–16. Frascati, Italy: Springer, June 28 – July 2 1999.

[7] J. Peng and R. J. Williams, Incremental Multi-Step Q-Learning, *Machine Learning*, vol. 22, pp. 283–290, 1996.

[8] G. Probst, S. Raub, and K. Romhardt, *Managing Knowledge*. London: Wiley, 1999.

[9] N. J. Kock, R. McQueen, and J. Corner, The Nature of Data, Information and Knowledge Exchanges in Business Processes: Implications for Process Improvement and Organizational Learning, *The Learning Organization*, vol. 4, no. 2, pp. 70–80, 1997.

[10] M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY: McGraw-Hill, 1983.

[11] G. A. Miller, WORDNET: A Lexical Database for English, *Communications of ACM*, vol. 11, pp. 39–41, 1995.

[12] T. Berners-Lee, Semantic Web Road Map, 1998. `http://www.w3.org/DesignIssues/Semantic`.

[13] A. Arpteg, *The Buyers Guide*. Master's thesis, Sweden, 1997.

[14] R. Collier, *Automatic Template Creation for Information Extraction*. PhD thesis, University of Sheffield, UK, 1998.

[15] N. R. Jennings, Agent-Based Computing: Promise and Perils, in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, (Stockholm, Sweden), pp. 1429–1436, 1999.

[16] Y. Shoham, Agent-oriented programming, *Artificial Intelligence*, vol. 60, no. 1, pp. 51–92, 1993.

[17] Y. Shoham, Time for action: On the relation between time, knowledge, and action, in *Proceedings of IJCAI-89*, pp. 954–959, 1989.

[18] Y. Shoham, AGENT-0: A simple agent language and its interpreter, in *Proceedings of the Ninth National Conference on Artificial Intelligence*, vol. 2, pp. 704–709, 1991.

[19] S. Franklin and A. Graesser, Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents, in *Proceedings of the Thrid International Workshop on Agent Theories, Architectures, and Languages*, (New York), pp. 21–35, Springer-Verlag, 1996.

[20] J. M. Bradshaw, ed., *Software Agents*. AAAI Press/The MIT Press, 1997.

[21] M. J. Wooldrige and N. R. Jennings, Agent Theories, Architectures, and Languages: A Survey, in *Proceedings of the ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, vol. 890 of *Lecture Notes in Artificial Intelligence*, pp. 1–39, Springer-Verlag, 1995.

[22] D. Chapman and P. Agre, Abstract reasoning as emergent from concrete activity, in *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*, Morgan Kaufmann Publishers, Inc., 1986.

[23] R. E. Fikes and N. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, vol. 5, no. 2, 1971.

[24] D. Chapmen, Planning for conjunctive goals, *Artificial Intelligence*, vol. 32, 1987.

[25] M. E. Bratman, Intentions, Plans, and Practical Reason, report, Harvard University Press, Cambridge, MA, 1987.

[26] J. P. Müller, *The Design of Intelligent Agents: A Layered Approach*, vol. 1177 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1996.

[27] J. Firby, *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Department of Computer Science, Yale University, 1989.

[28] N. R. Jennings and M. Wooldridge, *Handbook of Agent Technology*, ch. Agent-oriented Software Engineering. AAAI/MIT Press, 2000.

[29] H. V. D. Parunak, *Multi-Agent Systems*, ch. Industrial and Practial Applications of DAI. MIT Press, 1999.

[30] The Foundation for Intelligent Physical Agents. `http://www.fipa.org/` (2002-05-10), 2002.

[31] S. Virdhagriswaran, D. Osisek, and P. O'Connor, Standardizing agent technology, *ACM StandardView*, vol. 3, no. 3, pp. 96–101, 1995.

[32] T. Finin, R. Fritzson, D. McKay, and R. McEntire, KQML as an Agent Communication Language, in *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, (Gaithersburg, MD, USA), pp. 456–463, ACM Press, 1994.

[33] FIPA ACL Message Structure Specification. `http://www.fipa.org/specs/fipa00061/` (2002-05-01), 2000.

[34] Y. Labrou, T. Finin, and Y. Peng, Agent Communication Languages: The Current Landscape, *IEEE Intelligent Systems*, vol. 14, no. 2, pp. 45–52, 1999.

[35] W3C, Resource Description Framework. `http://www.w3.org/RDF/` (2002-05-01), 1999.

[36] The DARPA Agent Markup Language Homepage. `http://www.daml.org/` (2002-05-01), 2000.

[37] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, and M. Klein, OIL in a Nutshell, in *Proceedings of 12th International Conference on Knowledge Engineering and Knowledge Management*, 2000.

[38] J. Heflin, J. Hendler, and S. Luke, SHOE: A Knowledge Representation Language for Internet Applications, Report CS-TR-4078, Department of Computer Science, University of Maryland, 1999.

[39] M. Genesereth and R. Fikes, Knowledge Interchange Format (Version 3.0) - Reference Manual, report, Computer Science Department, Stanford University, 1992.

[40] DAML+OIL Homepage. `http://www.daml.org/2001/03/daml+oil-index.html` (2002-05-01), 2001.

[41] T. Finin and Y. Labrou, UMBC Agent Web. `http://agents.umbc.edu/` (2002-05-01), 2002.

[42] J. M. Vidal, MultiAgent.com. `http://www.multiagent.com/` (2002-05-01), 2002.

[43] AgentLink.org. `http://www.agentlink.org/` (2002-05-01), 2002.

[44] F. Bellifemine, A. Poggi, and G. Rimassa, JADE — A FIPA-compliant agent framework, in *Proceedings of the 4th International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-99)*, pp. 97–108, 1999.

[45] F. Bellifemine and T. Trucco, Java Development Framework. `http://sharon.cselt.it/projects/jade/` (2000-11-27), 2000.

[46] O. Hoffmann, JadeJessProtege. `http://sourceforge.net/projects/jadejessprotege/` (2002-05-01), 2001.

[47] Kinetoscope, Via 1.1 Developer's Guide. `http://www.kinetoscope.com/shared/docs/userguide.pdf` (2000-11-02), 1998.

[48] Microsoft .NET. `http://www.microsoft.com/net/` (2002-05-01), 2002.

[49] D. Day, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, and M. Vilain, Mixed-Initiative Development of Language Processing Systems, in *Proceedings of the 5th Conference on Applied NLP Systems (ANLP-97)*, 1997.

[50] R. Doorenbos, O. Etzioni, and D. Weld, A Scalable Comparison-Shopping Agent for the World-Wide Web, in *Proceedings of the First International Conference on Autonomous Agents*, pp. 39–48, 1997.

[51] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert, Crystal: Inducing a conceptual dictionary, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1314–1319, 1995.

[52] J. Rennie and A. K. McCallum, Using reinforcement learning to spider the web efficiently, in *Proceedings of the 16th International Conference on Machine Learning*, pp. 335–343, Morgan Kaufmann, San Francisco, CA, 1999.

[53] W. W. W. Consortium, HTML 4.01 Specification, 1999. `http://www.w3.org/TR/html4/` (2001-12-01).

[54] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1999. ISBN 0-262-19398-1.

[55] C. J. C. H. Watkins and P. Dayan, Q-learning, *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[56] University of Kalmar, Department of Technology, 2001. `http://www.te.hik.se/` (2001-12-01).

[57] G. Rummery and M. Niranjan, On-Line Q-Learning Using Connectionist Systems, Report CUED/F-INFENG/TR166, Cambridge University Engineering Department, 1994.

[58] C. Watkins and P. Dayan, Q-Learning, *Machine Learning*, vol. 3, no. 8, pp. 279–292, 1992.

[59] N. Ashish and C. Knoblock, Wrapper Generation for Semi-structured Internet Sources, in *Proc. Workshop on Management of Semistructured Data*, (Tucson), 1997.

[60] D. Mattrox, L. J. Sligman, and K. Smith, Rapper: A Wrapper Generator with Linguistic Knowledge, in *Workshop on Web Information and Data Management*, pp. 6–11, 1999.

[61] Y. Papakonstantinou, A. Gupta, H. Garcia-Monlina, and J. Ullman, A Query Translation Scheme for Rapid Implementation of Wrappers, in *International Conference on Deductive and Object-Oriented Databases*, pp. 97–107, August 1995.

[62] J. Hammer, M. Breunig, H. Garcia-Monlina, S. Nestorov, V. Vassalos, and R. Yerneni, Template-Based Wrappers in the TSIMMIS System, in *Proceedings of the Twenty-Sixth SIGMOD International Conference on Management of Data*, (Tucson, Arizona), May 12–15 1997.

[63] A. Sahuguet and F. Azavant, WysiWyg Web Wrapper Factory, 1999.

[64] G. Collste, ed., *Ethics in the Age of Information Technology*, ch. 2. Information Technology and Values, pp. 59–88. Linköping, Sweden: Linköpings University, 2000.

[65] D. G. Johnson, *Computer Ethics*, ch. 4. Ethics and the Internet I: Ethics Online, pp. 81–108. Upper Saddle River, New Jersey: Prentice Hall, third ed., 2001.

[66] D. G. Johnson, *Computer Ethics*, ch. 8. Ethics and the Internet II: Social Implications and Social Values, pp. 199–230. Upper Saddle River, New Jersey: Prentice Hall, third ed., 2001.

[67] F. Rundkvist, (Swedish) Aftonbladet.se stänger Tyck till-sidorna. `http://www.aftonbladet.se/vss/nyheter/story/0,2789,137738,00.html` (2002-05-10), 2002.

[68] Cyber-Rights and Cyber-Liberties, Felix Somm Decision in English. `http://www.cyber-rights.org/isps/somm-dec.htm` (1999-04-24), 1999.

[69] D. Lagerlöf, (Swedish) Nazisterna chattar på högskolans nät. `http://www.expressen.se/article.asp?id=77501` (2001-11-10), 2001.

[70] G. Collste, ed., *Ethics in the Age of Information Technology*, ch. 4. Information Technology and Society, pp. 143–243. Linköping, Sweden: Linköpings University, 2000.

[71] The Microsoft Knowledge Base. `http://support.microsoft.com/default.aspx?scid=fh;en-us;kbinfo` (2002-04-22), 2002.

[72] M. Koster, A Standard for Robot Exclusion. `http://www.robotstxt.org/wc/norobots.html` (2002-04-22), 1994.

[73] D. G. Johnson, *Computer Ethics*, ch. 2. Philosophical Ethics, pp. 26–53. Upper Saddle River, New Jersey: Prentice Hall, third ed., 2001.

[74] G. FitzPatrick, Structure Knowledge Initiative - SKICal. `http://www.skical.org/` (2002-05-01), 1998.

List of Figures

List of Equations

LINKÖPINGS UNIVERSITET

Avdelning, institution
Division, department

Institutionen för datavetenskap

Department of Computer and Information Science

Datum
Date

2002-12-16

Språk
Language

[ ] Svenska/Swedish
[X] Engelska/English
[ ] _____

Rapporttyp
Report category

[X] Licentiatavhandling
[ ] Examensarbete
[ ] C-uppsats
[ ] D-uppsats
[ ] Övrig rapport

_____

URL för elektronisk version

ISBN 91-7373-5892-2

ISRN LiU-Tek-Lic-2002:73

Serietitel och serienummer
Title of series, numbering

ISSN 0280-7971

Linköping Studies in Science and Technology

Thesis No. 1000

Titel
Title

Adaptive Semi-structured Information Extraction

Författare
Author

Anders Arpteg

Sammanfattning
Abstract

The number of domains and tasks where information extraction tools can be used needs to be increased. One way to reach this goal is to construct user-driven information extraction systems where novice users are able to adapt them to new domains and tasks. To accomplish this goal, the systems need to become more intelligent and able to learn to extract information without need of expert skills or time-consuming work from the user.

The type of information extraction system that is in focus for this thesis is semi-structural information extraction. The term semi-structural refers to documents that not only contain natural language text but also additional structural information. The typical application is information extraction from World Wide Web hypertext documents. By making effective use of not only the link structure but also the structural information within each such document, user-driven extraction systems with high performance can be built.

The extraction process contains several steps where different types of techniques are used. Examples of such types of techniques are those that take advantage of structural, pure syntactic, linguistic, and semantic information. The first step that is in focus for this thesis is the navigation step that takes advantage of the structural information. It is only one part of a complete extraction system, but it is an important part. The use of reinforcement learning algorithms for the navigation step can make the adaptation of the system to new tasks and domains more user-driven. The advantage of using reinforcement learning techniques is that the extraction agent can efficiently learn from its own experience without need for intensive user interactions.

An agent-oriented system was designed to evaluate the approach suggested in this thesis. Initial experiments showed that the training of the navigation step and the approach of the system was promising. However, additional components need to be included in the system before it becomes a fully-fledged user-driven system.

Nyckelord
Keywords

information extraction, artificial intelligence, semi-structured data, reinforcement learning, knowledge management

Department of Computer and Information Science
Linköpings universitet

Linköping Studies in Science and Technology
Faculty of Arts and Sciences - Licentiate Theses

No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)

No 28 **Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.

No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.

No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.

No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.

No 60 **Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.

No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.

No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.

No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.

No 74 **Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.

No 104 **Shamsul I. Chowdhury: S**tatistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.

No 108 **Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.

No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.

No 113 **Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.

No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.

No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.

No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.

No 139 **Christer Bäckström:** Reasoning about Interdependent Actions, 1988.

No 140 **Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.

No 146 **Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.

No 150 **Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.

No 165 **Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.

No 166 **Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.

No 174 **Yngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.

No 177 **Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.

No 181 **Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.

No 184 **Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.

No 187 **Simin Nadjm-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.

No 189 **Magnus Merkel:** Temporal Information in Natural Language, 1989.

No 196 **Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.

No 197 **Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.

No 203 **Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.

No 212 **Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.

No 230 **Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.

No 237 **Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.

No 250 **Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.

No 253 **Torbjörn Näslund:** SLDFA-Resolution - Computing Answers for Negative Queries, 1990.

No 260 **Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.

No 283 **Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge-Bases, 1991.

No 298 **Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.

No 318 **Lena Strömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.

No 319 **Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.

No 326 **Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.

No 328 **Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.

No 333 **Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.

No 335 **Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Sytems, 1992.

No 348 **Ulf Cederling:** Industrial Software Development - a Case Study, 1992.

No 352 **Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.

No 371 **Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.

No 378 **Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.

No 380 **Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.

No 381 **Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.

No 383 **Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.

No 386 **Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.

No 398 **Johan Boye:** Dependency-based Groudness Analysis of Functional Logic Programs, 1993.

No 402	**Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
No 406	**Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
No 414	**Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agentteoretiskt perspektiv, 1994.
No 417	**Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
No 436	**Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
No 437	**Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
No 440	**Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
FHS 3/94	**Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
FHS 4/94	**Karin Pettersson:** Informationssystemstrukturering, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
No 441	**Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
No 446	**Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
No 450	**Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
No 451	**Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
No 452	**Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
No 455	**Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
FHS 5/94	**Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
No 462	**Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
No 463	**Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
No 464	**Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
No 469	**Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
No 473	**Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
No 475	**Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
No 476	**Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
No 478	**Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
FHS 7/95	**Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylkprogram, 1995.
No 482	**Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
No 488	**Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
No 489	**Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
No 497	**Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
No 498	**Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
No 503	**Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
FHS 8/95	**Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
FHS 9/95	**Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
No 513	**Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
No 517	**Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
No 518	**Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
No 522	**Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
No 538	**Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
No 545	**Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
No 546	**Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
FiF-a 1/96	**Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
No 549	**Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
No 550	**Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag. 1996.
No 557	**Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
No 558	**Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 561	**Anders Ekman:** Exploration of Polygonal Environments, 1996.
No 563	**Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.
No 567	**Johan Jenvald:** Simulation and Data Collection in Battle Training, 1996.
No 575	**Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.
No 576	**Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.
No 587	**Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.
No 589	**Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
No 591	**Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.
No 595	**Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.
No 597	**Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.

No 598 **Rego Granlund:** C^3Fire - A Microworld Supporting Emergency Management Training, 1997.
No 599 **Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.
No 607 **Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.
No 609 **Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.
FiF-a 4 **Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
FiF-a 6 **Tommy Wedlund**: Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.
No 615 **Silvia Coradeschi**: A Decision-Mechanism for Reactive and Coordinated Agents, 1997.
No 623 **Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multiflex? 1997.
No 626 **David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.
No 627 **Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.
No 629 **Gunilla Ivefors:** Krigsspel coh Informationsteknik inför en oförutsägbar framtid, 1997**.**
No 631 **Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997
No 639 **Jukka Mäki-Turja:**. Smalltalk - a suitable Real-Time Language, 1997.
No 640 **Juha Takkinen**: CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.
No 643 **Man Lin**: Formal Analysis of Reactive Rule-based Programs, 1997.
No 653 **Mats Gustafsson**: Bringing Role-Based Access Control to Distributed Systems, 1997.
FiF-a 13 **Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
No 674 **Marcus Bjäreland:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.
No 676 **Jan Håkegård**: Hiera rchical Test Architecture and Board-Level Test Controller Synthesis, 1998.
No 668 **Per-Ove Zetterlund**: Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
No 675 **Jimmy Tjäder**: Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.
FiF-a 14 **Ulf Melin**: Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.
No 695 **Tim Heyer**: COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.
No 700 **Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.
FiF-a 16 **Marie-Therese Christiansson:** Inter-organistorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
No 712 **Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.
No 719 **Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.
No 723 **Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.
No 725 **Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.
No 730 **Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.
No 731 **Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.
No 733 **Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.
No 734 **Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.
FiF-a 21 **Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.
FiF-a 22 **Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.
No 737 **Jonas Mellin:** Predictable Event Monitoring, 1998.
No 738 **Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.
FiF-a 25 **Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.
No 742 **Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.
No 748 **Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.
No 751 **Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader,1999.
No 752 **Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.
No 753 **Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.
No 754 **Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.
No 766 **Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
No 769 **Jesper Andersson:** Towards Reactive Software Architectures, 1999.
No 775 **Anders Henriksson:** Unique kernel diagnosis, 1999.
FiF-a 30 **Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
No 787 **Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
No 788 **Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
No 790 **Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
No 791 **Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
No 800 **Anders Subotic:** Software Quality Inspection, 1999.
No 807 **Svein Bergum**: Managerial communication in telework, 2000.

No 809 **Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.
FiF-a 32 **Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
No 808 **Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.
No 820 **Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.
No 823 **Lars Hult:** Publika Gränsytor - ett designexempel, 2000.
No 832 **Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.
FiF-a 34 **Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
No 842 **Magnus Kald:** The role of management control systems in strategic business units, 2000.
No 844 **Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.
FiF-a 37 **Ewa Braf**: Organisationers kunskapsverksamheter - en kritisk studie av ”knowledge management”, 2000.
FiF-a 40 **Henrik Lindberg:** Webbaserade affärsprocesser - Möjligheter och begränsningar, 2000.
FiF-a 41 **Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.
No. 854 **Ola Pettersson:** Deliberation in a Mobile Robot, 2000.
No 863 **Dan Lawesson**: Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.
No 881 **Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.
No 882 **Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.
No 890 **Annika Flycht-Eriksson:** Domain Knowledge Management inInformation-providing Dialogue systems, 2001.
Fif-a 47 **Per-Arne Segerkvis**t: Webbaserade imaginära organisationers samverkansformer, 2001.
No 894 **Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.
No 906 **Lin Han**: Secure and Scalable E-Service Software Delivery, 2001.
No 917 **Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.
No 916 **Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.
Fif-a-49 **Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.
Fif-a-51 **Per Oscarsson:**Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
No 919 **Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.
No 915 **Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter. 2001.
No 931 **Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.
No 933 **Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.
No 938 **Bourhane Kadmiry**: Fuzzy Control of Unmanned Helicopter, 2002.
No 942 **Patrik Haslum**: Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.
No 956 **Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.
FiF-a 58 **Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.
No 964 **Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.
No 973 **Gert Jervan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.
No 958 **Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.
Fif-a 61 **Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.
No 985 **Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.
No 982 **Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.
No 989 **Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.
No 990 **Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.
No 991 **Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.
No 999 **Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002
No 1000 **Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.