Information Extraction from World Wide Web — A Survey —

Line Eikvil

July 1999

Contents

Chapter 1

Introduction.

The goal of information extraction (IE) is to transform text into a structured format and thereby reducing the information in a document to a tabular structure. Unseen texts are taken as input to produce fixed-format, unambiguous data as output. Specified information can then be extracted from different documents with a heterogeneous representation and be summarised and presented in a uniform way.

A uniform presentation of the data and their relations may be convenient for visual inspection and comparison of facts, for instance for uniform listings of job advertisements and listings of goods for comparison-shopping. When the data has a uniform representation, they may also be input to automatic analysis using for instance data mining techniques for discovery of patterns and further interpretation of these patterns.

IE systems do not attempt to understand the text in the input documents, but they analyse those portions of each document that contain relevant information. Relevance is determined by predefined domain guidelines which specify what types of information the system is expected to find.

Information extraction can be useful for any collection of documents from which you would want to extract specific facts. The World Wide Web is such a collection of documents. Here, information on a subject is often scattered among different Web servers and hosts, using many different formats, and it would be useful if this information could be extracted and integrated into a structured form.

Because the Web consists primarily of text, information extraction is central to any effort that would use the Web as a resource for knowledge discovery. An IE system can be thought of as an attempt to convert information from different text documents into database entries. Hence, successful information extraction from World Wide Web can turn the Web into a database.

Information extraction is a relatively new field which has developed over the last decade,

and which have met new challenges in the problem of extracting information from Web pages. This report will first, in Chapter 2, give an introduction to the field of information extraction and then, in Chapter 3, look at the development of the field of wrapper generation for Web sources. Chapter 4 gives an overview of systems developed for information extraction from Web sites, and Chapter 5 looks at different applications of the technology, and describes the first commercial systems that have appeared on this scene.

Chapter 2

Information Extraction

Information extraction (IE) is originally the task of locating specific information from a natural language document, and is a particular useful sub-area of natural language processing. IE systems have been developed both for structured text with tabular information and for free text such as news stories. A key element of IE systems is a set of text extraction rules or extraction patterns that identify relevant information to be extracted [52]. The dramatic growth in the number and size of on-line textual information sources has led to an increasing research interest in the information extraction problem.

This chapter will first focus on the development of the field of information extraction. Section 2.1 compares the field of IE to the more mature field of information retrieval (IR) and Section 2.2 gives a short history of IE. The two sections following explain the evaluation metrics applied in IE, which are inherited from IR, and gives an overview of the two different approaches commonly used for IE. The different types of documents for which information extraction has been applied are described in Section 2.5, and in Section 2.6 the characteristics of Web pages in this context are described.

2.1 IR and IE

Information extraction is different from the more mature technology of information retrieval (IR). Rather than to extract information the objective of IR is to select a relevant subset of documents from a larger collection based on a user query. The user must then browse the returned documents to get the desired information.

The contrast between the aims of IE and IR systems can be stated as follows: IR *retrieves relevant documents* from collections, while IE *extracts relevant information* from documents. Hence, the two techniques are complementary, and used in combination they can provide powerful tools for text processing [24].

Not only do IE and IR differ in aims; they also differ in the techniques usually deployed. These differences are due partly to their different aims and partly to the history of the fields. Most work in IE has emerged from research on rule-based systems in computational linguistics and natural language processing, while information theory, probability theory and statistics have influenced IR [24].

2.2 History of IE

While the science of automatic information retrieval is a mature field which has basically been around as long as databases of documents have existed, the field of automatic information extraction has only been driven forward in the last decade. Two factors have been important for the development of the field, the exponential growth in the amount of both online and offline textual data, and the focus on the field through the Message Understanding Conferences (MUC) during the last ten years.

A precursor of IE was and is the field of text understanding. AI researchers have been building various systems where the goal is to get an accurate representation of the contents of an entire text. These systems typically operate in very small domains only, and they are usually not very portable to new domains [53].

Since the late 1980's the US government has been sponsoring MUC to evaluate and advance the state-of-the art in information extraction. The MUC's involve a set of participants from different sites, usually a mix of academic and industrial research labs. Each participating site builds an IE system for a predetermined domain. The IE systems are all evaluated on the same domain and text collection, and the results are scored using an official scoring program.

The objective of the conferences has been to get a quantitative evaluation of IE systems, which prior to these conferences had been performed only sporadically and often on the same data on which they had been trained. The conferences provided the first large-scale effort to evaluate natural language processing systems. How to evaluate the systems has been a non-trivial issue, and through the conferences standard scoring criteria have been developed. The focus of the various MUC's has been tasks like Latin American terrorism, joint ventures, microelectronics, and company management changes.

Information extraction research has been rather successful in the past five or six years, and name recognition components of the leading systems have achieved near-human performance in English and Japanese. Through the MUC conferences it has been demonstrated that fully automatic IE systems can be built with state-of-the-art technology, and that, for some selected tasks, their performance is as good as the performance of a human expert [53]. However, in several of the subtasks, the scores in the top group in every MUC evaluation since 1993 have been roughly the same (bearing in mind however that the MUC tasks have become more complex). The primary advance has been that more and more sites are able to perform at this level, because the techniques used have converged. Moreover,

building systems that perform at this level currently requires a great investment in time and expertise. In addition, the vast bulk of the research so far has been done only on written text and only in English and a few other major languages.

2.3 Evaluation metrics

The necessity for evaluation metrics for the information extraction problem came with the message understanding conferences (MUC). The starting points for the development of these metrics were the standard IR metrics of recall and precision. However, the definitions of these measures have been altered from those used in IR, although the names have been retained. The alterations allow for reporting possible overgeneration in IE where, unlike IR, data not present in the input can be erroneously produced [24].

In the information extraction task, recall may be crudely interpreted as a measure of the fraction of the information that has been correctly extracted, and precision as a measure of the fraction of the extracted information that is correct. Recall then refers to how much of the information that was correctly extracted, while precision refers to the reliability of the information extracted. Precision and recall are defined as follows:

$$Precision = \frac{\#\text{ correct answers}}{\#\text{ answers produced}}$$

$$Recall = \frac{\#\text{ correct answers}}{\#\text{ total possible corrects}}$$

Both recall and precision are always on the interval [0, 1], their optimum being at 1.0. They are, however, inversely related to each other, meaning that by allowing for a lower recall you can achieve a higher precision and vice versa.

When comparing the performance of different systems, both precision and recall must be considered. However, as it is not straightforward to compare the two parameters at the same time, various combination methods have been proposed. One such measure is the F-measure, which combines precision, P, and recall, R, in a single measurement as follows:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

The parameter β determines how much to favour recall over precision. Researchers in information extraction frequently reports the F1 score of a system where $\beta = 1$, weighing precision and recall equally. Using the F-measure, the relative performance of systems reporting different values for recall and precision, can easily be compared.

2.4 Approaches to IE

There are two main approaches to the design of IE systems, which can be called the *knowledge engineering approach* and the *automatic training approach* [3].

In the *knowledge engineering approach* grammars expressing rules for the system are constructed by hand using knowledge of the application domain. The skill of the knowledge engineer plays a large role in the level of performance of the system, but the best performing systems are often hand crafted. However, the development process can be very laborious, and sometimes the required expertise may not be available.

For the *automatic training approach* there is not the same need for system expertise when customising the system for a new domain. Instead someone with sufficient knowledge of the domain and the task at hand annotates a set of training documents. Once a training corpus has been annotated, a training algorithm is run, training the system for analysing novel texts. This approach is faster than the knowledge engineering approach, but requires that a sufficient volume of training data is available.

2.5 Free, structured and semistructured text

Free text. Originally the aim of information extraction was to develop practical systems which could take short natural language texts and extract a limited range of key pieces of information from them. For example, the texts might be news articles about terrorist attacks and the key information might be the perpetrators, their affiliation, the location, the victims, etc. Or the texts might be pharmaceutical research abstracts and the key information might be new products, their manufacturers, patent information, etc.

IE systems for free text has generally used natural language techniques, and the extraction rules are typically based on patterns involving syntactic relations between words or semantic classes of words. Several steps are required including syntactic analysis, semantic tagging, recognisers for domain objects such as person and company names, and extraction rules. The rules or patterns can be hand-coded or generated from training examples annotated with the right label by a human expert.

The state-of-the-art in information extraction from free text is not comparable to human capability but still provides useful results. This is true whether the rules are hand coded or automatically learned [52]. Unrestricted natural language understanding is a long way from being solved. However, information extraction methods can work because they depend on strong a priori restrictions on the kinds of patterns they need to search for.

Structured text. Structured text is defined as textual information in a database or file following a predefined and strict format. Such information can easily be correctly

extracted using the format description. Usually quite simple techniques are sufficient for extracting information from text provided that the format is known, otherwise, the format must be learned.

Semistructured text. Semistructured data are an intermediate point between unstructured collections of textual documents and fully structured tuples of typed data. Such texts fall between structured and free text, and have previously been almost inaccessible to IE systems.

Semistructured text is ungrammatical and often telegraphic in style, and does not follow any rigid format. Natural Language Processing (NLP) techniques are deployed to design rules for extraction of information from free text. However, these methods which are appropriate for grammatical text will usually not work for semistructured text, which seldom contains full sentences. Hence, for semistructured texts the traditional techniques of IE can not be used, and at the same time simple rules used for rigidly structured text will not be sufficient.

Some form of structuring is, however, present in semistructured text, and extraction patterns are often based on tokens and delimiters like for instance HTML-tags. Syntactic and semantic information can only be utilised to a limited extent.

2.6 Web documents

The World Wide Web provides a vast source of information. This information is often semistructured, although you may also find both structured and free text. The information is also dynamic, it contains hyperlinks and can be represented in different forms and is globally shared over multiple sites and platforms. Hence, the Web provides a special challenge, and has been the driving force behind research on information extraction from structured and semistructured text.

Some define all Web pages as semistructured as they all contain some structuring information concerning display styles. However, Hsu [31] give a better categorisation of types of Web pages: A Web page that provides itemised information is *structured* if each attribute in a tuple can be correctly extracted based on some uniform syntactic clues, such as delimiters or the orders of attributes. *Semistructured* Web pages, however, may contain tuples with missing attributes, attributes with multiple values, variant attribute permutations, and exceptions. A Web page is *unstructured* if linguistic knowledge is required to extract the attributes correctly.

In the following we will talk about Web pages both as structured, semistructured and unstructured dependent on how the contents are organised. However, the structuredness of a Web page will always depend on the attributes the users want to extract. Usually, machine generated Web pages are very structured, while hand generated Web pages are

less structured, but there are plenty of exceptions.

When it comes to extracting information from Web pages, the same applies to Web documents as to other semistructured documents: The traditional NLP techniques are not well suited, as the sources often do not exhibit the rich grammatical structure such techniques are designed to exploit. In addition, NLP techniques tend to be slow, and this can be a problem as the volume of document collections on the Web can be large and the extraction is often expected to be performed on the fly.

A large portion of the data on the Web are rendered regularly as itemised lists of attributes, such as many searchable Web indexes. For these semistructured Web pages, the regularity of their appearance can be exploited for extracting data instead of using linguistic knowledge.

The organisation and the hyperlinking of documents is an important aspect when extracting information from Web pages that is not present in text documents for the traditional IE task. For instance it may be necessary to follow hyperlinks to obtain all the information you are looking for. The extraction rules will be dependent on the overall organisation of the Web page, and some rules will have limitations that prohibit them from being used on some types of pages.

Web pages that appear as results from queries to online databases often result in a set of hyperlinked pages. In [14], such semistructured Web pages are classified as either (i) *one-level one page* result, where one page contains all items related to the original query, (ii) *one-level multi-page* result, where more links must be followed to get the full listing of answers, and (iii) *two-level pages*, where for each item in the first level a link must be followed in order to navigate to a page containing all information related to the items.

2.7 Summary

The field of IE is a fairly new field that has evolved over the last decade. Both because of the focus on the field through the Message Understanding Conferences, where evaluation of proposed systems was focused, and because of the increasing amount of online documents.

Information extraction is performed both on free, structured and semistructured text. Techniques from Natural Language Processing are often used for free text. NLP techniques are however not well suited for structured and semistructured text, as these techniques require full grammatical sentences. Instead delimiter- and token-based methods are often used for information extraction from structured and semistructured text.

The Web is the source of an enormous amount of online documents that often contain semistructured text. Web documents are different from the documents traditionally used in IE; the volume is very large, new documents appear often, document contents change often, a large portion of the documents contain structured or semistructured text, and the

documents may contain hyperlinked information. Hence, Web documents have provided new challenges for the IE field.

Chapter 3

rapper Generation

The Internet presents a large and growing number of information sources, which can be found either by manual browsing or by using a search engine. These information sources live isolated, with no real connections with one another, and each service exists independently. This has created a need for information extraction from the Web, which can extract and collect information from independent sources.

The publication of structured and semistructured data on the Web is a trend that is likely to increase, and there is also a growth of the so-called hidden Web. This refers to Web pages generated on the fly from some database, based on user requests. In [37] it is claimed that 80% of the Web is already in the hidden Web. These pages are not available to Web crawlers for indexing and can not be reached through search engines. This means that there will be a special need for tools that can extract information from such pages.

Information extraction from Web sites is often performed using wrappers. In Sections 3.1 and 3.2 the concept of wrappers will be explained and the development of the field of wrapper generation will be described. In Section 3.3 we summarise different approaches to wrapper generation. As wrapper generation by hand is tedious, automatic wrapper generation has become important. Machine learning is an attractive alternative to creating rules by hand, and in Section 3.4 a short description of relevant techniques for inductive learning is given.

3.1 Wrappers

A wrapper can be seen as a procedure that is designed for extracting content of a particular information source and delivering the content of interest in a self-describing representation. In the database community, a wrapper is a software component that converts data and queries from one model to another. In the Web environment, its purpose should be to convert information implicitly stored as an HTML document into information explicitly

stored as a data-structure for further processing.

A wrapper for a Web source accepts queries about information in the pages of that source, fetches relevant pages from the source, extracts the requested information and returns the result. It consists of a set of extraction rules and the code required to apply these rules and is specific to one source. To extract information from several independent sources a library of wrappers are needed. Wrappers should execute quickly, because they are usually used online to satisfy users' queries. Ideally, the wrapper should also be able to cope with the changing and unstable nature of the Web, like network failures, ill-formed documents, change in the layout, etc.

There are two primary advantages to building wrappers around Web sources: The ability to obtain the relevant information from an individual source is enhanced, and all the sources for which a wrapper is build can be queried using a common query language. It is then possible to get an integrated access to several sources, and the Web sources may be queried in a database-like fashion using a common query language.

3.2 From IE to WG

The need for tools that could extract and integrate data from multiple Web sources led to the development of the wrapper generation (WG) field. This field appeared independently of the traditional IE community, and a typical WG application extracts data from Web pages that are generated online, based on user queries, using predefined HTML templates. In the wrapper generation community, such a collection of documents is called a semistructured source. In order to combine data coming from these sources, the relevant data must be extracted from the HTML templates. Hence, the wrappers are merely an IE application for this particular information source [40].

Traditional IE systems use extraction patterns based on a combination of syntactic and semantic constraints. However, as previously mentioned, for semistructured documents like those of a WG application, linguistic extraction patterns are usually not applicable. In order to deal with the new types of application domains, researchers introduced new sets of extraction patterns, and a typical WG system generates delimiter-based extraction patterns that do not use linguistic constraints. For this type of Web pages, which are generated on the fly, all documents are generated by filling in the same template. After seeing a couple of sample documents one can therefore usually identify the fragments of the template that represent the delimiters of each individual field [40].

Although the Web pages, which are the target of WG applications, follow syntactic regularities, the information extraction from these sources is not trivial. Scalability is for instance an important challenge, as there is both a large number of sites and a large variation in formatting styles. Flexibility is another challenge, as the format of sources may change.

3.3 Wrapper generation

The construction of a wrapper can be done manually, or by using a semi-automatic or automatic approach. The manual generation of a wrapper often involves the writing of ad hoc code. The creator has to spend quite some time understanding the structure of the document and translating it into program code. Though it is simpler to program the information extraction by hand for semistructured Web pages than for free text, the task is not trivial, and hand coding can be tedious and error-prone.

Tools for helping the *manual construction of wrappers* have been developed. Some approaches make use of expressive grammars in which the structure of a Web page may be described, and provide tools for generating code for the extraction based on the specified grammar. However, even specifying the grammars is tedious and time consuming and it requires a high level of expertise.

Although many wrappers to date are hand-written, manually constructed IE systems cannot adapt to domain changes, and must be adapted for each new problem domain. This means that manually created wrappers require high maintenance costs. For Web sources this is a problem as the number of sources of interest is often very large and the content and structure of different information sources may vary significantly. In addition, new sources appear frequently and the format of existing sources may change. Therefore, mechanisms and technology to aid the construction of wrappers are essential for automatic extraction of Web information.

Semi-automatic wrapper generation benefits from support tools to help design the wrapper. Some approaches offer a demonstration-oriented interface where the user shows the system what information to extract. Using a graphical interface, one may then perform programming by demonstration, showing the application what fields to extract. This approach means that expert knowledge in wrapper coding is not required at this stage, and it is also less error-prone than coding. However, it must be demonstrated for each new site and for each changed site how the data should be extracted as these systems can not themselves induce the structure of a site.

Automatic wrapper generation uses machine-learning techniques, and the wrapper research community has developed learning algorithms for a spectrum of wrappers - from the very simple to the relatively complex. Even automatic generation systems do however require a minimum of intervention of human experts. The systems must usually go through a training phase, where it is fed with training examples, and in many cases this learning has to be supervised.

Wrapper induction is a technique for automatically constructing wrappers, where inductive learning methods are used to generate the extraction rules. The user labels the relevant data on a set of pages and the system learns extraction rules based on these examples. The accuracy of these rules will depend on both the number and the quality of the examples. The examples are of good quality if they are representative of the pages to be processed.

3.4 Inductive learning of wrappers

Various machine learning techniques like symbolic learning, Inductive Logic Programming (ILP), wrapper induction, statistical methods, and grammar induction have been applied for the problem of information extraction. In wrapper induction, the problem of wrapper construction is posed as one of inductive learning.

At the highest level, inductive learning is the task of computing from a set of examples of some unknown target concept, a generalisation that explains the observations. The idea is that a generalisation is good if it explains the observed examples and makes accurate predictions when previously unseen examples are encountered. Inductive learning has proven useful for classification problems, knowledge acquisition, knowledge discovery from large databases, and program construction from partial specifications.

Inductive learning is accomplished through inductive inference, the process of reasoning from a part to a whole, from particular instances to generalisation, or from individual to universal. A teacher provides examples for the learner, and the learner generalises the given examples to induce the general rules. Because human learning is mostly based on experimental observation (empirical learning), it is easier for us to produce good examples than to generate explicit and complete general rules. In general, inductive methods can be characterised as search methods over a hypothesis space.

Inductive learning methods for supervised learning can be classified as zero-order or first-order. The principal difference between zero-order and first-order supervised learning systems are the form of training data and the way that a learned theory is expressed.

Data for the *zero-order learning* programs such as CART and C4.5 comprise preclassified cases; each described by its values for a fixed collection of attributes. These systems develop theories in the form of decision trees or production rules that relate a case's class to its attribute values. Unfortunately, decision tree learners lack expressiveness because they are based on propositional logic. They cannot learn concepts including relations between objects such as family relationship. From the database point of view, they can deal only with one relation consisting of attribute-value pairs.

The relational *first-order learning* methods allow induction over structured examples that can include first-order logical predicates and functions and unbounded structures such as lists or trees. In particular inductive logic programming studies the induction of rules in first-order logic and the learning of logic programs and other relational knowledge from examples.

Inductive logic programming research stands at the intersection of the traditional fields of machine learning and logic programming. Most other machine learning algorithms are restricted to finite feature-based representations of examples and concepts and cannot learn complex relational and recursive knowledge. However, due to the expressiveness of first-order logic, ILP methods can learn relational and recursive concepts that cannot

be represented in the attribute-value representations assumed by most machine-learning algorithms. ILP allows learning with much richer representations, and can learn much more complex concepts than decision tree learners. Hence, ILP techniques have been applied to the problem of learning to extract information from documents with more complex structures and relations.

ILP algorithms use two different approaches to induction: *bottom-up (generalisation)* or *top-down (specialisation)*. The bottom-up approach is data-driven, and starts with selecting one or more examples and formulates a hypothesis to cover those examples and then generalises the hypothesis to cover the rest of the examples. The top-down approach starts with the most general hypothesis available, making it more specific through the introduction of negative examples. In general top-down algorithms can induce a large class of logic programs, but need a relatively large number of samples. On the other hand, bottom-up algorithms work with a very small number of examples but induce only a small class of programs.

Several experimental ILP systems have been developed, where FOIL [47] and GOLEM [39] are the best known. FOIL was developed by Quinlan in 1989, and is a top-down ILP-algorithm. Based on a training set of positive and negative facts, a clause that covers some positive and no negative facts is found, and all the facts covered by this clause from the training set is removed until there are no positive facts in the training set. GOLEM (Muggleton and Feng 1990) uses a greedy covering algorithm. The clause construction is bottom-up, based on the construction of least-general generalisations of more specific clauses. Generalisation is performed until all positive examples are covered, and non-of the negative examples are implied.

3.5 Summary

The publication of structured data on the Web is expected to increase. There is also a growth in the number of Web pages that are generated on the fly based on user queries to a database. These pages will not be indexed by search-engines, and there will be an increasing need for tools that can help extract relevant information from these and other collections of Web documents.

A wrapper for a Web source is a program specific to that source, which extracts the requested information and returns the result. This is useful for getting an integrated access to different information sources. As the need for such tools has increased, the wrapper generation field has evolved independently of the traditional field of IE, deploying techniques that are less dependent on full grammatical sentences than NLP techniques.

Wrappers can be constructed manually by directly writing the necessary code for extracting the information or by specifying the structure of a Web page through a grammar and automatically translating the grammar to program code. In any case, the procedure of creating wrappers are tedious, and as Web pages change and new pages appear a new

wrapper must be created. Therefore, information extraction from the Web has led to research into semi-automatic and automatic creation of wrappers.

Wrapper induction is a method for automatic wrapper generation that uses machine-learning techniques. In wrapper induction, the problem of wrapper construction is posed as one of inductive learning, where the task is to compute from a set of examples, a generalisation that explains the observations. A teacher provides examples for the learner, and the learner generalises the given examples to induce the general rules.

Inductive logic programming stands at the intersection of the traditional fields of machine learning and logic programming, and uses rules in first-order logic. Due to the expressiveness of first-order logic, ILP methods can learn relational and recursive concepts that cannot be represented in the attribute-value representations assumed by most machine-learning algorithms. ILP techniques have therefore been applied to the problem of learning to extract information from documents with more complex structures and relations.

Chapter 4

Systems for rapper Generation

The early approaches to structuring and wrapping Web sites were essentially based on manual techniques. Here it is assumed that a human programmer examines a site and manually codes wrappers that abstract the logical features of HTML pages and store them in a database. One of the first approaches to a framework for manual building of Web wrappers is the TSIMMIS system [13, 25, 28, 29]; The Stanford-IBM Manager of Multiple Information Sources (1995). The goal of TSIMMIS is to provide tools for accessing, in an integrated fashion, multiple information sources, and to ensure that the information obtained is consistent. The focus is on development of languages and tools to support this wrapping process.

For Web sites where the volume is large and the content and structure change dynamically, there is however a need for a more effective construction of wrappers. Generally the database communities have focused on how the heterogeneous information can be integrated, while wrappers are designed manually. On the other hand the AI communities have focused on how machine learning can be used to automate the learning of Web sites. In this chapter systems of the latter category, using machine learning techniques for semi-automatic and automatic generation of wrappers, will be described.

The complexity of a wrapper and the wrapper generation will depend on the level of structure in the Web sites. In Section 4.1 systems based on techniques that are tailored for relatively well-structured Web pages will be described. These systems typically arise from the wrapper generation community. Then, in Section 4.2, systems based on techniques intended for Web pages with a less rigorous structure are presented. The latter systems are more influenced by the traditional IE field.

4.1 Structured and semistructured Web pages

In this section the systems ShopBot, WIEN, SoftMealy and STALKER are described. These systems can be said to belong to the field of wrapper generation, where the typical application is to extract data from Web pages that are generated online based on queries to a database. These systems use delimiter-based extraction patterns and do not use syntactic or semantic constraints, and they are limited to work on fairly structured data.

4.1.1 ShopBot

R. B. Doorenbos, O. Etzioni, D. S. Weld (1996/1997) [17, 18]

ShopBot is a comparison-shopping agent, specialised to extract information from Web vendors. As such ShopBot is more restricted than most of the other systems that will be described. The algorithm focuses on vendor sites with form-based search pages returning lists of products with a tabular format. Information is extracted from the resulting pages using a combination of heuristic search, pattern matching and inductive learning techniques.

The ShopBot operates in two phases; the learning phase, which is performed offline, and the online comparison-shopping phase. During the learning phase, the vendor sites are analysed to learn a symbolic description of each site. In the comparison-shopping phase, the learned vendor descriptions are used to extract information from the sites and find the best price for the product specified by the user.

In the learning phase some simple heuristics are used to determine the correct search form, and how to pose a query to this form. Then the learner must determine the format of the result page. The result pages for a query are assumed to consist of a header, a body and a tail, where the header and tail are consistent across different pages, and the body contains all the desired product information. The format of the result page is determined in 3 steps. First the failure template, appearing for non-existing products are learned by analysing the result pages of dummy queries for non-words (ex. "xldccxx-no-product"). Then by performing several queries for potential products the head and tail of the result pages are identified, assuming matching prefixes and suffixes over pages.

Finally the format of the body containing the product description is determined. Initially, a number of possible abstract formats of product descriptions are defined and represented as sequences of HTML tags and text strings. The body is broken into logical lines representing vertical-space-delimited text, and the learning algorithm compares the different abstract formats with the logical lines to find the best match. In this way the algorithm is able to induce the format of the descriptions of the products, but it does not induce the labels of the information slots. The price, which is the most important information in comparison-shopping, is extracted using special hand-coded techniques.

4.1.2 WIEN

N. Kushmerick (1997) [33, 34]

The Wrapper Induction ENviroment is a tool for assisting with wrapper construction. The Wrapper Induction is designed for automatic learning of Web pages, and is strongly influenced by ShopBot. However, Kushmerick is the first to introduce the term wrapper induction. The method is not designed for a specific area. It works on structured text containing tabular information, and it is demonstrated for, but not restricted to, HTML-documents.

The approach can handle Web pages that have what they call a HLRT organisation, where there is a Head delimiter, a set of Right and Left delimiters for each fact to be extracted, and a Tail delimiter at the end. It looks for uniform delimiters that identify the beginning and end of each slot and for delimiters that separate the tabular information from surrounding text. Pages that conform to this regularity are nearly always automatically formatted responses to a search index or a listing of an underlying database.

Kushmerick seeks to automate wrapper construction as much as possible, and thereby avoiding the task of hand labelling a set of example documents. Hence, a set of techniques for automatically labelling example documents has been developed. The labelling algorithm takes as input domain-specific heuristics for recognising instances of the attributes to be extracted. The system requires that these heuristics are provided as input, but is not concerned with how they are obtained. They may have been defined manually; however, this is still a smaller job than hand-labelling each example site.

Inductive learning is used to perform the wrapper induction, generalising from example query responses. The induction algorithm takes as input a set of labelled pages. It then searches the space of wrappers defined by the HLRT wrapper model and iterates over all possible choices of delimiters until a HLRT wrapper which is consistent with the labelled pages is found. A model based on computational learning theory is used to predict how many examples the learning algorithm must observe to ensure that the probability of failure for the resulting wrapper is below a specified limit.

WIEN uses only delimiters that immediately precede and follow the data to be extracted [41], and cannot wrap sources in which some items are missing or sources where the items may appear in a varying order. Multi-slot rules are used for the extraction. This means that the extraction rules are able to link together related information, as opposed to single-slot rules that can only extract isolated data (e.g. in a document that contains several names and addresses, single-slot rules can not specify which is the address of a particular person). [41]

4.1.3 SoftMealy

C-H. Hsu (1998) [30, 31]

After Kushmerick's system WIEN several systems seeking to improve his wrapper induction have appeared. SoftMealy is a system that learns to extract data from semistructured Web pages by learning wrappers specified as non-deterministic finite automata. This representation and learning algorithm is, as opposed to Kushmerick's system, said to handle missing values, slots with multiple values, and variable permutations.

An inductive generalisation algorithm is used to induce contextual rules from training examples. A training example provides an ordered list of the facts to be extracted together with the separators between these facts. The wrapper induction takes as input the set of labelled tuples, providing information on the positions of the separators and the permutations of the facts to be extracted. The algorithm takes these positions as training examples and generalises their context into contextual rules, which are the output.

The induced wrapper is a non-deterministic finite automaton where states represent the facts to be extracted and state transitions represent contextual rules defining the separators between them. State transitions are determined by matching contextual rules that characterise the context delimiting two adjacent facts. To extract the facts, the wrapper recognises the separators surrounding it.

The general rules of SoftMealy allow for wildcards and handle both missing items and items appearing in various orders. However, in order to deal with items of various orders, SoftMealy has to see training examples that include each possible ordering of the items. The extraction patterns of SoftMealy are more expressive than the ones defined for WIEN [41].

4.1.4 STALKER

I. Muslea, S. Minton, C. Knoblock. (1998) [42, 43, 44]

STALKER is a supervised learning algorithm for inducing extraction rules. Training examples are supplied by the user who has to select a few sample pages and mark up the relevant data (the leaves of a so-called EC tree). When the page has been marked, the sequence of tokens that represent the content of the page, together with the index of the token that represents the start of an item are generated. The sequences of tokens (words, HTML tags) and wildcards, can then be used as landmarks to locate an item to be extracted on a page, and the wrapper-induction algorithm, generates extraction-rules that are expressed as simple landmark-grammars. The extraction process handles text, but no link information.

Web documents are described by the so-called Embedded Catalog (EC) formalism. An EC description of a page is a tree structure in which an internal node is either a homogeneous

list of items or a heterogeneous tuple of items. The content of the root node in the EC tree is the whole sequence of tokens (the entire document), while the content of an arbitrary node represents a subsequence of the content of its parent. The end-nodes are the items of interest to the user (i.e. the data to be extracted).

STALKER uses a sequential covering algorithm, meaning that it starts by generating linear landmark automata which can generate as many as possible of the positive training examples. The linear landmark automaton is a non-deterministic finite automaton where a transition between states takes place only if the input string is accepted for the transition between the current and the next state. Then it tries to generate new automata for the remaining examples, and so on. When all the positive examples are covered, STALKER returns the solution, an SLG (Simple Landmark Grammars) where each branch corresponds to a learned landmark automaton.

STALKER is able to wrap information sources that have an arbitrary number of levels. Each node is extracted independently of its siblings, hence the ordering of the items in the document need not be fixed, and it is possible to extract information from documents with missing items or documents with items appearing in various orders. This makes STALKER more flexible than WIEN which is only able to handle sources with a fixed ordering of items. As opposed to SoftMealy, which has the same ability to handle missing items and items appearing in varying order, STALKER does not need to see training examples that include each possible ordering of the items.

STALKER rules are, unlike WIEN, single-slot. However, this does not represent a limitation, because STALKER uses the Embedded Catalog Tree to group together the individual items extracted from multi-slot templates [41].

4.2 Semistructured and unstructured Web pages

In this section the systems RAPIER, SRV and WHISK are described. These systems belong to a more sophisticated class of wrapper generators than those presented in the previous section and they can handle a wider range of texts. Although they work on multiple types of text, they do not rely on the use of semantic and syntactic information, but can make use of it when it is available and are able to employ hybrid extraction patterns.

These systems are closer to the traditional IE approach, and can be said to lie between the field of IE and WG as their focus is to develop machine learning methods for the IE problem. The methods are based on inductive logic programming or relational learning, and are related to induction algorithms like FOIL (SRV, WHISK) and GOLEM (RAPIER).

4.2.1 RAPIER

E. Califf (1997) [11, 12]

RAPIER, the Robust Automated Production of Information Extraction Rules, learns rules for the complete IE task, and works on semistructured text. It takes pairs of documents and filled templates indicating the information to be extracted and learns pattern-matching rules to extract fillers for the slots in the template. The human interaction of the system consists of providing texts with filled templates.

The learning algorithm incorporates techniques from several inductive logic programming systems and learns unbounded patterns that include constraints on the words and part-of-speech tags surrounding the filler. The learning algorithm consists of a specific to general search (bottom-up), beginning with the most specific rule that matches a target slot in the training. Pairs of rules are chosen at random and a beam search is conducted to find the best generalisation of the two rules, taking a least general generalisation, then adding constraints until no progress is made in several successive iterations.

The extraction patterns of RAPIER are based both on delimiters and content description, using rules represented by patterns that exploit the syntactic and semantic information. A part-of-speech tagger is used to obtain the syntactic information, while a lexicon of semantic classes is used to obtain the semantic information. The tagger takes sentences as input and labels each word as noun, verb, adjective etc. A part-of-speech tagger is faster and more robust than a full parser, but does not give as much information.

The rules for the information extraction are indexed by a template name and a slot name and consist of 3 parts: *a pre-filler:* a pattern that should match the text preceding the target text, *a filler:* a pattern that should match the target text, and *a post-filler:* a pattern which should match the text immediately following the target text.

Each pattern is a sequence of *pattern items*, matching exactly one word, or *pattern lists*, matching N words. The text must fulfil the requirements of the patterns to give a match. Possible requirements are that the text must be (i) a list of words, where one word must match the document text, (ii) a list of syntactic tags, where one tag must match the tag of the document text, or (iii) a list of semantic classes, where the document text must belong to one of these classes.

The approach with extraction of fields centred around the target phrase means that the system can only perform single-slot extraction. However, it might be possible to perform multi-slot extraction by dividing the text into more than three fields [52].

4.2.2 SRV

D. Freitag (1998) [21, 22, 23]

The Sequence Rules with Validation (SRV), is a top down relational algorithm for information extraction. Input to SRV is a set of pages, labelled to identify instances of the fields to extract, and a set of features defined over tokens. The output is a set of extraction rules.

In SRV the IE problem is viewed as a classification problem where all possible phrases from the text (up to a maximum length) are considered as instances. Every candidate instance in a document is presented to the classifier, and the system assigns to each of these phrases a metric indicating confidence that the phrase is a correct filler for the target slot. The original version of SRV uses a classifier that is a relational rule learner that does a top-down induction similar to FOIL. In [23] two additional classifiers, a "rote" learner and a naive Bayes classifier, are compared to the original SRV.

The features used in SRV come in two varieties, simple features and relational features. Given a token drawn from a document, a number of obvious features may be used, such as length, character type, orthography, part of speech, or lexical meaning, and also relational features that encode token adjacency. The relational features are what give SRV its relational character.

SRV's example space consists of all text fragments from the training document collection as long (in number of tokens) as the smallest field instance in the training corpus but no longer than the largest. The extraction process involves examining every possible text fragment of appropriate size to see whether it matches any of the rules.

SRV starts (as FOIL) with the entire set of examples, positive and negative, where a negative example is any fragment that is not tagged as a field instance. Induction proceeds by matching all examples not covered by previously learned rules. When a rule is good enough, i.e. it either covers only positive examples or a further specialisation is judged to be unproductive, all positive examples matching it are removed from the training set, and the process is repeated.

SRV's rule representation is expressive and able to incorporate orthographic features and other evidence such as part of speech or semantic classes when available. No prior syntactic analysis is required. SRV is similar to STALKER and RAPIER, in that it is able to extract particular items independently of other relevant items. The relational learner is also as RAPIER only designed to extract single slots. This is different from for instance WIEN, which learns rules to extract multiple slots.

4.2.3 WHISK

S. Soderland (1998) [52]

The WHISK system is designed to handle all types of extraction problems, from very structured text to the semistructured text common to Web documents, and used in conjunction with a syntactic analyser and semantic tagging, it also handles extraction from free text such as news stories. When applied to structured or semistructured text WHISK does not require prior syntactic analysis, but for free text WHISK works best with input annotated by a syntactic analyser and semantic tagger.

WHISK is a supervised learning algorithm and requires a set of hand-tagged training instances. The tagging process is interleaved with the learning. In each iteration WHISK presents the user with a batch of instances to tag, then WHISK induces a set of rules from the expanded training set.

It starts with a set of untagged instances and an empty training set of tagged instances, where an instance is a smaller unit of a document, like a sentence. At each iteration a set of untagged instances are selected and presented to the user to annotate. The user adds a tag for each case frame to be extracted from the instance. What constitutes an instance depends on the domain. For structured or semistructured text, a text is broken into multiple instances based on the HTML tags or other regular expressions. For free text, a sentence analyser segments the text into instances where each instance is a sentence or a sentence fragment.

The tags of the training instances are used to guide the creation of rules and to test the performance of proposed rules. If a rule is applied successfully to an instance, the instance is considered to be covered by the rule. If the extracted phrases exactly match a tag associated with the instance, it is considered correct.

WHISK belongs to the family of machine learning algorithms known as covering algorithms, and is related to algorithms that learn classification rules by top-down induction. First, the most general rule that covers the seed is found, then terms are added to a rule one at a time until the errors are reduced to zero or a pre-pruning criterion has been satisfied. The metric used to select a new term is the Laplacian expected error of the rule, $\frac{e+1}{n+1}$, where n is the number of extractions made on the training set and e is the number of errors among those extractions. The learning process is repeated until a set of rules has been generated that covers all positive extractions from the training. Post-pruning is then performed to remove rules that may be overfitting the data.

WHISK shares SRV and RAPIER's ability to handle either structured or semistructured text, but does not have their limitation of single-slot extraction. Like WIEN, WHISK associates related information into multi-slot case frames. WHISK also differs from SRV and RAPIER in that rather than operating on documents WHISK operates on instances which are typically sentences or similarly sized units.

Name	struct	semi	free	Multislot	Missing items	Permutations
ShopBot	X	-	-	-	-	-
WIEN	X	-	-	X	-	-
SoftMealy	X	X	-	-	X	X*
STALKER	X	X	-	*	X	X
RAPIER	X	X	-	-	X	X
SRV	X	X	-	-	X	X
WHISK	X	X	X	X	X	X*

Table 4.1: *This table summarises the characteristics of the systems treated in this section. The first three columns indicate the type of text that is handled by the systems, the fourth column says whether the systems perform multislot extraction i.e. if they just extract isolated facts or if they are able to link together related information. STALKER uses single-slot rules but is able to group related information together after extraction. The last two columns treat the ordering of the facts. Systems that can handle permutations of facts, but are required to be trained on all possible permutations are marked with X*.*

WHISK handles variable permutations of slots, but has the same drawback as SoftMealy; it needs to see training examples of all possible orderings of items. WHISK is also unable to represent negated features, and due to less expressive feature set, WHISK is said to have a lower performance than SRV.

4.3 Summary

In this chapter different systems for learning wrappers automatically have been reviewed. Table 4.1 summarises the characteristics of the functionality of each of these systems. The X's mark a systems ability to perform extraction on the data type in question. The table gives an overview of the type of texts the systems can handle, whether the systems extract only isolated data or whether they are able to link together related information (multislot extraction), and whether the systems are able to handle missing items or variably ordered items.

The systems presented have a background in the information extraction (IE) and wrapper generation (WG) communities. Both communities use machine learning algorithms to generate extraction patterns for online information sources. ShopBot, WIEN, SoftMealy and STALKER belong to a group of systems that generate wrappers for fairly structured Web pages using delimiter-based extraction patterns. RAPIER, WHISK and SRV handle a wider range of texts, and are closer related to the field of traditional IE. All three use relational learning systems, where SRV and WHISK are both covering algorithms with top-down search, while RAPIER is primarily bottom-up, but with a top-down component.

Chapter 5

Applications and Commercial Systems

The World Wide Web makes enormous amounts of information available to users regardless of location. Agents that can perform information extraction are useful for automatically gathering and analysing this information on a user's behalf, and the number of applications for this technology is large.

In this chapter, the first section gives examples of different application areas for which information extraction has been tested, and in Section 5.2 the first commercial systems that have appeared on the scene are presented.

5.1 Examples of applications

The Internet presents numerous sources of useful information like telephone directories, product catalogues, stock quotes, weather forecasts, etc. The data in the Web pages may originate from different sources. Often the original source is a database, and the Web page presented to the user is a result of forming a query to the database and using a cgi-script to automatically generate the result page.

In many cases it can be useful to combine information from several such sources to obtain the complete picture. However, the Web's browsing paradigm does not readily support retrieving and integrating data from multiple sites. Tools for information extraction can therefore be useful for a number of applications where the objective is to extract and collect specific information from a set of different Web pages.

The systems presented in the previous chapter have been tested for a number of different types of Web information, and some of these are reviewed in the following.

- *Product descriptions.*
  ShopBot is specially designed for this type of application [17] [18], and compiles an overview of product offers by extracting product information from several Web vendors and summarising the result for the user. For comparison-shopping, the extracted product descriptions are sorted by price.

- *Restaurant guides.*
  Information on restaurants can be spread out over different Web sites. In experiments with STALKER, information is extracted from a set of Web pages where each HTML page contains exactly one restaurant review. The information extracted consists of items like the name of the restaurant, the type of food, cost, cuisine, address, phonenumber and the review. [42, 43].

- *Seminar announcements.*
  Here the task can be to extract information like speaker, location and time from a collection of Web pages with seminar announcements. This is one of the experiments performed with SRV [22].

- *Job advertisements.*
  Such advertisements can be found several places on the net, for instance as newsgroup postings. This application is among others tested for RAPIER and WHISK, which extracts information like title, salary, location etc. [11].

- *Executive succession.*
  This represents a task for the more traditional IE field, as it includes information extraction from free text. In an experiment with WHISK, a collection of Wall Street Journal articles was analysed searching for management succession events [52]. The target was to extract the name of the corporation, the position, the name of the person moving into that position and the name of the person moving out of the position.

This is just a small selection of applications for which this technology can be useful, and there are obviously many more. Some additional examples are: rental ads, geographic information, holiday and travel information, weather reports, bibliographic information etc. In general agents that perform information extraction and collection can be used for any list-type of data where information is spread out over a set of Web pages.

5.2 Commercial systems

Of the applications mentioned in the previous section, comparison-shopping is mainly where this technology has been used in commercial systems. One reason for the commercial focus on this application is the general focus on electronic commerce and the general growth in applications on the Internet connected to this.

Another reason is that such sites are well suited for the task because they are designed for users to find things quickly and they have a uniform look and feel. The online vendors have found it useful to obey certain regularities to facilitate sales to human users. This also facilitates the operation of systems for comparison-shopping, and the possibility of automating the process.

As different vendors often sell the same products, a service that can collect and compare information from several vendors is useful for Web shoppers. Often, the information that these vendors provide, is fetched from a database and generated on the fly based on user queries. This makes the information part of the hidden Web, which can not be indexed by search engines. Agents for comparison-shopping are therefore the only alternative to manual collection of product information.

In the following, three commercial comparison-shoppers will be presented: Junglee, Jango and mySimon. These are currently the most prominent systems on the market, and each system uses a somewhat different approach to the problem of product information extraction.

Jango and mySimon use a real-time approach, gathering product information whenever a shopper makes a request, while Junglee takes a different approach, collecting a database of product information locally and updating it when necessary. Each shopping-agent takes user inputs and returns a list of matching products, where the user can compare the offers on this list, and decide from which merchant to make the purchase of the product. In the following a brief description of each system will be given.

5.2.1 Junglee

Junglee was founded in June 1996 by graduates from Stanford University, and bought by Amazon for approximately $ 180 million dollars in August 1998. Their comparison-shopper uses a so-called Virtual Database (VDB) technology and exploits a mix of HTML and XML to present data from multiple sources [46, 48].

The VDB gathers, structures and integrates the data from disparate data sources and provides the application programmer with the appearance of a single unified database system. Wrappers interface with the data sources, transforming them into a database.

The Virtual Database has two main components, the data integration system and the data publishing system. Data are extracted from different sources by the integration system and kept in a database. The updating of this database is scheduled by the publishing system.

The data integration system has 3 elements: a set of wrappers, a mapper and an extractor. *The wrappers* give a common interface to the different Web sites, and are created using descriptive programming in a language specifically designed to capture the structure of Web sites and the relation between hyperlinked sites.

The mapper transforms the extracted data to a common format using defined mapping rules. *The extractor* extracts structure from unstructured textual data using dictionaries and linguistic rules to describe how to extract the required features from the text. In both cases specifically designed languages are used to formulate the rules. There will be one wrapper for each Web site while an extractor is typically created for an entire collection of Web sites with similar information.

5.2.2 Jango

Jango was previously known as Shopbot and is a product of NETbot. It is based on research by Oren Etzioni and Dan Weld from the University of Washington [17, 18]. Excite bought NetBot for $ 35 million in October 1997 and integrated Jango into its shopping channel.

Jango consists of 4 components [8]; (i) a natural language front-end that transforms a request to logical product descriptions, (ii) a query router that determines product category and associates it with a set of Web sites, (iii) an aggregation engine that queries the selected sites in parallel, and finally (iv) the filter that extracts the information from the Web sites using a Shopbot-like analysis (see Section 4.1).

From the URL of a store's home page and from knowledge about product domains, Jango learns how to shop at the stores during a learning phase. In this phase it learns the format of the product descriptions of each vendor and learns to extract product attributes like price. During the shopping phase, the learned descriptions are used to extract information on products specified by the user. This information is extracted in parallel from each online vendor, and the results are sorted by ascending order of price and presented to the user.

5.2.3 MySimon

mySimon was co-founded by Michael Yang and Yeogirl Yun in April 1998. A so-called Virtual Learning Agent (VLA) technology, which was developed by Yeogirl Yun, is used to learn Web sites [54].

The Virtual Learning Agent (VLA) technology operates by creating a number of intelligent agents that are able to mimic human shopping behaviour and can be trained to extract specific information from any merchant site on the Web.

The agents are manually trained by the use of a GUI environment to teach them how to shop at a merchant site. The people performing the manual training need not be programmers, and are called Simon Product Intelligence (SPI). The SPI's surf the Internet for the merchants' product sites. At the product site the GUI environment is used to copy information from the site to an online form. Based on the SPI's behaviour and the copied information, the code that makes the shopping agent work is generated automatically.

5.3 Summary

There are a number of different applications where information extraction from Web can be useful. However, the largest success has been seen in the field of comparison-shopping, where information on products are extracted from a collection of Web vendor sites. During the last two years commercial systems for comparison-shopping have appeared, and Jango, mySimon and Junglee are the most prominent ones.

Jango performs the extraction online and uses machine learning to learn the structure of Web sites. mySimon also performs the information extraction online, but uses a different approach for the learning, where non-programmers shop at product sites teaching intelligent agents how to extract the relevant information from the sites.

Junglee extracts data and stores them in a database, and uses the database as the source for the comparison-shopper. Specifically designed languages are used to manually describe the structure of Web sites, and to generate the code for the extraction.

Chapter 6

Summary and Discussion

Summary. Information extraction is a fairly new field that has developed during the last decade. Research in the field has been focused through the MUC conferences. These conferences have also focused on the evaluation of such systems, and evaluation metrics for the task has been defined.

Information extraction can be performed both on structured, semistructured and free text. For free text, techniques from natural language processing are often used. For structured and semistructured text, which seldom contains full grammatical sentences, delimiter-based techniques that exploit the structure of the documents are used.

Web pages have been the main target for the research on information extraction from structured and semistructured documents. Wrappers are developed to extract information from a specific Web site, and through a library of wrappers for different Web pages a uniform presentation of the data and their relations can be obtained.

The construction of wrappers is often tedious and requires expert knowledge, and as Web pages are very dynamic the costs of maintaining wrappers can be high. Hence, automatic construction of wrappers for Web sites has become a focused problem, and machine learning techniques based on inductive learning have been applied for this task.

Several (academic) systems for semi-automatic and automatic generation of wrappers have appeared. These systems use machine learning algorithms to generate extraction patterns for online information sources. ShopBot, WIEN, SoftMealy and STALKER generate wrappers for well-structured Web sites using delimiter-based extraction patterns. RAPIER, WHISK and SRV are able to handle sources that are less structured. These methods are closer related to the field of traditional IE and use relational learning.

There are a series of application areas where information extraction and wrapper generation for Web sites can be useful. Currently, only one of these areas has become the arena of commercial systems, namely the area of comparison-shopping. Here, the objective is to

extract product information from different online vendors, presenting a unified list to the user. The most prominent and successful systems in this area have been Jango, Junglee and mySimon.

Discussion. The search engines of today are not powerful enough for all tasks. They return a collection of documents, but they can not extract the relevant information from these documents. There is also an increase in the so-called hidden Web, which search engines are unable to access. Hence, there is a need for tools that can facilitate the task of extracting and gathering relevant information from Web pages.

The field of information extraction from the Web is still new, but will continue to develop, as Internet information integration is both important and difficult. The use of machine learning techniques for the problem will still be highly relevant, due to the need to automate the process as much as possible to be able to cope with the large amount of dynamic data found on the Web. In [52], a combination of different types of approaches, obtaining adaptive systems, is believed to be a promising direction for the development of the technology. Also, in [36], a combination of methods, using a hybrid of both linguistic and syntactic features, is suggested.

The systems presented in this report mainly focus on the extraction of information from HTML-coded documents. In the next few years the coming of XML is expected. While HTML is designed to describe what a document looks like, and is the formatting language of a document, XML tells you what a document means. Hence, XML defines the content rather than the presentation. This can make wrapper induction simpler, but will not eliminate the need for wrappers.

For the future, the challenges of the field will be to make systems for wrapper induction, which are flexible and scalable, and thereby able to adapt to the growth and the dynamics of the Web.

Bibliography

[1] S. Abiteboul.
Querying Semistructured Data.
Proceedings of the International Conference on Database Theory (ICDT), Greece, January 1997.

[2] B. Adelberg.
NoDoSE - A tool for Semi-Automatically Extracting Semistructured Data from Text Documents.
Proceedings ACM SIGMOD International Conference on Management of Data, Seattle, June 1998.

[3] D. E. Appelt, D. J. Israel.
Introduction to Information Extraction Technology.
Tutorial for IJCAI-99, Stockholm, August 1999.

[4] N. Ashish, C. A. Knoblock.
Semi-automatic Wrapper Generation for Internet Information Sources.
Second IFCIS Conference on Cooperative Information Systems (CoopIS), South Carolina, June 1997.

[5] N. Ashish, C. A. Knoblock.
Wrapper Generation for semistructured Internet Sources.
SIGMOD Record, Vol. 26, No. 4, pp. 8–15, December 1997.

[6] P. Atzeni, G. Mecca.
Cut & Paste.
Proceedings of the 16'th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'97), Tucson, Arizona, May 1997.

[7] M. Bauer, D. Dengler.
TrIAs - An Architecture for Trainable Information Assistants.
Workshop on AI and Information Integration, in conjunction with the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[8] P. Berka.
Intelligent Systems on the Internet.

http://lisp.vse.cz/ berka/ai-inet.htm, Laboratory of Intelligent Systems, University of Economics, Prague.

[9] L. Bright, J. R. Gruser, L. Raschid, M. E. Vidal.
A Wrapper Generation Toolkit to Specify and Construct Wrappers for Web Accessible Data Sources (WebSources).
Computer Systems Special Issue on Semantics on the WWW, Vol. 14 No. 2, March 1999.

[10] S. Brin.
Extracting Patterns and Relations from the World Wide Web.
International Workshop on the Web and Databases (WebDB'98), Spain, March 1998.

[11] M. E. Califf, R. J. Mooney.
Relational Learning of Pattern-Match Rules for Information Extraction.
Proceedings of the ACL Workshop on Natural Language Learning, Spain, July 1997.

[12] M. E. Califf.
Relational Learning Techniques for Natural Language Information Extraction.
Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, August 1998. Technical Report AI98-276.

[13] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom.
The TSIMMIS Project: Integration of Heterogeneous Information Sources.
In Proceedings of IPSJ Conference, pp. 7–18, Tokyo, Japan, October 1994.

[14] B. Chidlovskii, U. M. Borghoff, P-Y. Chevalier.
Towards Sophisticated Wrapping of Web-based Information Repositories.
Proceedings of the 5'th International RIAO Conference, Montreal, Quebec, June 1997.

[15] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, S. Slattery.
Learning to Extract Symbolic Knowledge from the World Wide Web.
Proceedings of the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[16] M. Craven, S. Slattery, K. Nigam.
First-Order Learning for Web Mining.
Proceedings of the 10'th European Conference on Machine Learning, Germany, April 1998.

[17] R. B. Doorenbos, O. Etzioni, D. S. Weld.
A Scalable Comparison-Shopping Agent for the World Wide Web.
Technical report UW-CSE-96-01-03, University of Washington, 1996.

[18] R. B. Doorenbos, O. Etzioni, D. S. Weld.
A Scalable Comparison-Shopping Agent for the World-Wide-Web.

Proceedings of the first International Conference on Autonomous Agents, California, February 1997.

[19] O. Etzioni
Moving up the Information Food Chain: Deploying Softbots on the World Wide Web.
AI Magazine, 18(2):11-18, 1997.

[20] D. Florescu, A. Levy, A. Mendelzon.
Database Techniques for the World Wide Web: A Survey.
ACM SIGMOD Record, Vol. 27, No. 3, September 1998.

[21] D. Freitag.
Information Extraction from HTML: Application of a General Machine Learning Approach.
Proceedings of the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[22] D. Freitag.
Machine Learning for Information Extraction in Informal Domains.
Ph.D. dissertation, Carnegie Mellon University, November 1998.

[23] D. Freitag.
Multistrategy Learning for Information Extraction.
Proceedings of the 15'th International Conference on Machine Learning (ICML-98), Madison, Wisconsin, July 1998.

[24] R. Gaizauskas, Y. Wilks.
Information Extraction: Beyond Document Retrieval.
Computational Linguistics and Chinese Language Processing, vol. 3, no. 2, pp. 17–60, August 1998,

[25] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom.
Integrating and Accessing Heterogeneous Information Sources in TSIMMIS.
In Proceedings of the AAAI Symposium on Information Gathering, pp. 61–64, Stanford, California, March 1995.

[26] S. Grumbach and G. Mecca.
In Search of the Lost Schema.
Proceedings of the International Conference on Database Theory (ICDT'99), Jerusalem, January 1999.

[27] J-R. Gruser, L. Raschid, M. E. Vidal, L. Bright.
Wrapper Generation for Web Accessible Data Source.
Proceedings of the 3'rd IFCIS International Conference on Cooperative Information Systems (CoopIS-98), New York, August 1998.

[28] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, A. Crespo.
Extracting Semistructured Information from Web.
Proceedings of the Workshop on Management of Semistructured Data, Tucson, Arizona, May 1997.

[29] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, V. Vassalos.
Template-Based Wrappers in the TSIMMIS System.
Proceedings of the 26'th SIGMOD International Conference on Management of Data, Tucson, Arizona, May 1997.

[30] C-H. Hsu.
Initial Results on Wrapping Semistructured Web Pages with Finite-State Transducers and Contextual Rules.
Workshop on AI and Information Integration, in conjunction with the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[31] C-H. Hsu and M-T Dung.
Generating Finite-Sate Transducers for semistructured Data Extraction From the Web.
Information systems, Vol 23. No. 8, pp. 521–538, 1998.

[32] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, P. J. Modi, I. Muslea, A. G. Philpot, S. Tejada.
Modeling Web Sources for Information Integration.
Proceedings of the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[33] N. Kushmerick, D. S. Weld, R. Doorenbos.
Wrapper Induction for Information Extraction.
15'th International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, August 1997.

[34] N. Kushmerick.
Wrapper Induction for Information Extraction.
Ph.D. Dissertation, University of Washington. Technical Report UW-CSE-97-11-04, 1997.

[35] N. Kushmerick.
Wrapper induction: Efficiency and expressiveness.
Workshop on AI and Information Integration, in conjunction with the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[36] Kushmerick, N.
Gleaning the Web.
IEEE Intelligent Systems, 14(2), March/April 1999.

[37] S. Lawrence, C.l. Giles.
Searching the World Wide Web.
Science magazine, v. 280, pp. 98–100, April 1998.

[38] A. Y. Levy, A. Rajaraman, J. J. Ordille.
Querying Hetereogeneous Information Sources Using Source Descriptions.
Proceedings 22'nd VLDB Conference, Bombay, September 1996.

[39] S. Muggleton, C. Feng.
Efficient Induction of Logic Programs.
Proceedings of the First Conference on Algorithmic Learning Theory, New York, 1990.

[40] I. Muslea.
Extraction Patterns: From Information Extraction to Wrapper Induction.
Information Sciences Institute, University of Southern California, 1998.

[41] I. Muslea.
Extraction Patterns for Information Extraction Tasks: A Survey.
Workshop on Machine Learning for Information Extraction, Orlando, July 1999.

[42] I. Muslea, S. Minton, C. Knoblock.
STALKER: Learning Extraction Rules for Semistructured, Web-based Information Sources.
Workshop on AI and Information Integration, in conjunction with the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[43] I. Muslea, S. Minton, C. Knoblock.
Wrapper Induction for Semistructured Web-based Information Sources.
Proceedings of the Conference on Automatic Learning and Discovery CONALD-98, Pittsburgh, June 1998.

[44] I. Muslea, S. Minton, C. Knoblock.
A Hierarchical Approach to Wrapper Induction.
Third International Conference on Autonomous Agents, (Agents'99), Seattle, May 1999.

[45] S. Nestorov, S. Aboteboul, R. Motwani.
Inferring Structure in Semistructured Data.
Proceedings of the 13'th International Conference on Data Engineering (ICDE'97), Birmingham, England, April 1997.

[46] STS Prasad, A. Rajaraman.
Virtual Database Technology, XML, and the Evolution of the Web.
Data Engineering, Vol. 21, No. 2, June 1998.

[47] J.R. Quinlan, R. M. Cameron-Jones.
FOIL: A Midterm Report.
European Conference on Machine Learning, Vienna, Austria, 1993.

[48] A. Rajaraman.
Transforming the Internet into a Database.
Workshop on Reuse of Web information, in conjunction with WWW7, Brisbane, April 1998.

[49] A. Sahuguet, F. Azavant.
WysiWyg Web Wrapper Factory (W4f).
http://cheops.cis.upenn.edu/ sahuguet/WAPI/wapi.ps.gz, University of Pennsylvania, August 1998.

[50] D. Smith, M. Lopez.
Information Extraction for Semistructured Documents.
Proceedings of the Workshop on Management of Semistructured Data, in conjunction with PODS/SIGMOD, Tucson, Arizona, May 1997.

[51] S. Soderland.
Learning to Extract Text-based Information from the World Wide Web.
Proceedings of the 3'rd International Conference on Knowledge Discovery and Data Mining (KDD), California, August 1997.

[52] S. Soderland.
Learning Information Extraction Rules for Semistructured and Free Text.
Machine Learning, 1999.

[53] K. Zechner.
A Literature Survey on Information Extraction and Text Summarization.
Term paper, Carnegie Mellon University, 1997.

[54] *About mySimon.*
http://www.mysimon.com/about_mysimon/company/backgrounder.anml