Applied Intelligence 21, 195–224, 2004

Information Extraction from the Web: System and Techniques

LUO XIAO AND DIETER WISSMANN
CT SE 5, Siemens AG, Erlangen, Germany
Luo.Xiao@siemens.de
Dieter.Wissmann@siemens.de

MICHAEL BROWN
Global Transactions, Ltd., Berlin, Germany
Mike@GTCT.com

STEPHAN JABLONSKI
Department of Computer Sciences VI, University of Erlangen-Nuremberg, Germany
Stefan.Jablonski@informatik.uni-erlangen.de

Abstract. Information Extraction (IE) systems that can exploit the vast source of textual information that is the internet would provide a revolutionary step forward in terms of delivering large volumes of content cheaply and precisely, thus enabling a wide range of new knowledge driven applications and services. However, despite this enormous potential, few IE systems have successfully made the transition from laboratory to commercial application. The reason may be a purely practical one—to build useable, scaleable IE systems requires bringing together a range of different technologies as well as providing clear and reproducible guidelines as to how to collectively configure and deploy those technologies.

This paper is an attempt to address these issues. The paper focuses on two primary goals. Firstly, we show that an information extraction system which is used for real world applications and different domains can be built using some autonomous, corporate components (agents). Such a system has some advanced properties: clear separation to different extraction tasks and steps, portability to multiple application domain, trainability, extensibility, etc. Secondly, we show that machine learning and, in particular, learning in different ways and at different levels, can be used to build practical IE systems. We show that carefully selecting the right machine learning technique for the right task and selective sampling can be used to reduce the human effort required to annotate examples for building such systems.

Keywords: information extraction, machine learning, knowledge acquisition, internet applications, methodology and design

1. Introduction

There has been an explosive growth in the amount of information available on networked computers around the world, much of it in the form of natural language documents. An increasing variety of search engines exist for retrieving such documents based on matching keywords. However, precision of retrieval is often sub-optimal—it is all too often still a painful quest for a user to try several search queries and read a number of documents before a required piece of information is found.

The vision of an internet capable of directly producing specific answers to specific queries still seems a number of years away. A less ambitious goal may be to allow more content-driven retrieval of documents within a particular vertical domain. Hence a traveller may want to retrieve all holidays to a *Mediterranean island*, in a *hotel with water sports* and *less than 100 meters to the beach*. Similarly, someone searching for a new career move might want to see all openings for an *IT manager* in *a content management company* within *commuting distance of London*. Answering such questions about available information requires a deeper "understanding" of the original natural language text (such as those terms highlighted above in italics) than simply matching keyword variations. One way of providing more "understanding" is through Information Extraction (IE) technology. IE is the task of locating specific pieces of data within a natural language document. In recent years, IE has been the focus of such large-scale research initiatives as the DARPA's MUC (Message Understanding Conference) program [1]. Moreover, the advent of the internet has given IE a particular commercial relevance.

As will be described in this paper, there are a wide range of techniques that are applicable to the problem of information extraction. For example, recent research in computational linguistics indicates that empirical or corpus based methods are currently the most promising approach to developing robust, efficient natural language processing (NLP) systems [2]. These methods automate the acquisition of much of the complex knowledge required for NLP/IE by training on large volumes of annotated text, e.g. tree-banks of parsed sentences [3], or by Rule Induction and Generalisation [4]. Such approaches are commercially appealing because of the relatively low manual cost of maintenance, under the conditions that suitable training data can be cheaply provided.

Another approach is to use more carefully crafted syntactic rules that identify and extract fine pieces of information from structured or semi-structured documents. The precision of such approaches makes them interesting for many industrial and business domains [5, 6]. Supervised (training by example) machine learning techniques [7, 8] can also be used to augment or even replace human rule encoding, which can greatly reduce the construction task. Several different symbolic and statistical methods have been employed, but most of them are used to generate one part of a larger IE system.

A complete IE system has a framework, which consists of various IE components. Each component carries out one or more IE tasks. Such systems normally apply more than one IE techniques. Some systems are designed not only for information extraction but also for other language engineering tasks (e.g. GATE [9]). Nevertheless, a ubiquitous difficulty with IE systems is that they are difficult and time consuming to build, and they generally contain highly domain specific components, making porting to new domains time consuming. Furthermore, many IE systems are designed for limited research problems (such as MUC evaluation). However, often the technology and methodology behind such systems is not scaleable and therefore it is difficult to make the transition to successful commercial applications. Thus, more efficient means for developing generally-applicable IE systems are desirable, combing multiple techniques in a complementary way. A complete top down design has to be made from system architecture to complicated underlying extraction algorithms. Hence, two levels of designs for a practical oriented IE system are required: the whole system architecture design and the underlying methods to undertake the IE tasks stepwise.

Recently, information extraction from online-documents (e.g. HTML-web pages) and well-formed structured XML-documents has gained significantly in importance [10]. Researchers used various empirical methods and algorithms to apply information extraction to web pages [5, 11–13]. However, most approaches are specialist non-standard solutions. On the other hand, advanced techniques in the XML-family can provide powerful searching, querying and selection functionalities (such as XPath, XML-Query, XQL etc.) across a wide range of applications. A promising strategy therefore is to seek to apply these developed mature techniques for information extraction from web pages.

This paper has focused on two primary goals. First, we show how an IE system (CapturePlus), which is used for real world applications and different domains, can be built using a combination of autonomous, corporate components (agents). Such a system requires: clear separation of different extraction tasks and steps, portability to other application domains, trainability and an open interface. Second, we show that machine learning, and, in particular, learning in different ways and levels, can be used to build practical information extraction systems. Although the system architecture is designed both for normal and web (HTML) documents, this paper focuses primarily on techniques for information extraction from web pages.

The rest of this paper is organized as follows. Section 2 presents background knowledge on information extraction, and machine learning approach. Section 3 describes the system architecture and introduces each component of the system briefly. Sections 4–6 describe algorithms, methods and learning in each information extraction step (preprocessing, name extraction and template filling). Then, in Section 7 we describe and discuss evaluation results. Finally, Section 8 has a conclusion and suggests ideas for future research directions.

2. Background Knowledge

The aim of this section is to provide a more extensive background on the techniques used in information extraction systems and to further qualify some of the problems involved in building IE systems.

2.1. Information Extraction

Information Extraction (IE) is a process, which takes unseen texts as input and produce, fixed format, unambiguous data as output. This data may be used directly for display to users, or may be stored in a database or spreadsheet for direct integration with a back-office system, or may be used for indexing purposes in search engine/Information Retrieval (IR) applications.

It is instructive to compare IE and IR; whereas IR simply finds texts and presents them to the user (c.f. classic search engines), IE analyses texts and presents only the specific information extracted from the text that is of interest to a user. For example, a user might want to create an overview of share price companies in the content management sector. First they may have to enter a list of relevant terms (e.g. "*content management, knowledge management, information management*," . . .) and then read each returned 'hit' themselves to see if (I) it relates to a relevant company and (II) what the actual share price is. In contrast, in an idealised scenario, a user could set up an IE system that is able to (I) more accurately classify company websites by a deeper understanding of their content and (II) having found a correct web site, to extract the company name and share price automatically and add it to a database.

There are advantages and disadvantages of IE with respect to IR. IE systems are more difficult and knowledge intensive to build. Much manual effort goes into tailoring IE rules, whereas IR is often based just on statistical weightings of keyword lists. This makes IR very generalized whereas specific IE applications may not port easily to new applications or domains. Moreover, at run time, the matching techniques of IE are more computationally expensive. This means that it is difficult to build complex IE techniques into the kind of search engine needed to provide general querying across the whole internet where the volume of online documents is massive and processing time per document must be minimal—IE is at best suited to specific vertical domains. However, the cost benefit of IE with respect to IR comes primarily through reducing the amount of search and reading effort required by end users to find the relevant information from large document collections.

There are five basic tasks that can be associated with Information Extraction (as defined by the leading forum for this research, the Message Understanding Conferences [14])

- Named Entity recognition (NE): Finds and classifies items of information in text, e.g. names, places etc.
- Co-reference Resolution (CO): Identifies identity relations between items of information texts.
- Template Element construction (TE): Organizes and combines different NE results to create a single entity description.
- Template Relation construction (TR): Finds relations between TE entities.
- Scenario Template production (ST): Fits TE and TR results into specified event scenarios—i.e. creates a narrative structure.

For current commercial requirements, the tasks of NE and TE are the most interesting. For example, a marketing division might use IE to create a market analysis database by extracting product data from online catalogues. Similarly, a call-center might use IE to analyse the content of incoming emails in order to sort and deliver them to responsible people or to make automatic replies.

The general process to build an IE system could be described as follows. First, a knowledge engineer defines an empty *template*. A template is a structured object made up of a set of slots (attribute value pairs) which represents a checklist of the types of information item that need to be extracted from a particular type of document/website. Secondly, the knowledge engineer must create a set of (NE) agents capable of finding information items to fill each slot in the template. Thirdly, the knowledge engineer may need to add rules to adjudicate when different NE agents produce possibly ambiguous results—i.e. interpret the same piece of text in two different ways (note, this problem is inevitable in any real IE application!) Finally, the output format, which can be a database, XML file, text report, etc. must be defined. In particular for the second (NE agent definition) and third tasks (template filling) machine learning can be applied to greatly reduce the manual construction effort and hence cost of deploying an IE solution.

2.2. Machine Learning Approaches

Many well known Machine Learning (ML) approaches are used to help build high performance IE systems. ML usually refers to the changes in systems that perform recognition, planning, diagnosis or control tasks associated with Artificial Intelligence (AI).

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

- Some tasks cannot be easily defined except as examples of input-output pairs (either because no well defined function can be formulated, or because too much effort and expertise is required to manually define such a function). For example, a straightforward task is to mark a sequence of text as being an exemplar of a given type of information. Indeed, in some commercial scenarios, such as content factories, this may occur indirectly as a natural bi-product of human operators filling out standard content forms (e.g. via drag and drop). ML can be used to make useful generalisation from such training examples.
- ML can be used to find hidden relationships that are emergent from large volumes of data. For instance, template elements in semi-structured documents may have spatial relationships, which are not easily discovered by manual inspection. Machine learning methods can often be used to extract these relationships.
- Human designers produce IE systems in the laboratory that do not perform robustly in the real world because it is difficult to predict *a priori* all text variations for occurrences of a particular type of information. Machine learning methods can be used for on-the-fly improvement of deployed systems. Incremental machine learning algorithm (such as Case-Based Reasoning) are particularly well suited for this.
- The pure size and volume of required extraction rules may prohibit human encoding. For instance, a single XSL-Pattern used to extract information from HTML documents can be more than 1000 bytes long. Similarly, just to reliably extract dates from text may require hundreds of rules to cover all variations. Manual creating such patterns causes errors and is time consuming. ML may automate the process or "fill in the gaps" in manually encoded rules.
- Information is continually being created. Vocabulary changes. Even within a single vertical domain, there is a continual stream of new information and concepts. Continual manual maintenance of an IE system is usually prohibitive to commercial success. However, ML techniques, e.g. using statistical machine learning approaches to track new terminology on the internet, may be applied to drive a continual maintenance process.

Many well known machine learning algorithms are widely used in the area Information Extraction [7, 8]. Examples include; statistical approach [15], inductive rule learning [5, 11, 16–18], Bayes network [12], decision tree [19], Hidden Markov Model [5, 17, 20], Case-Based Reasoning [21, 22], Neural Network [23], etc.

In CapturePlus, supervised machine learning methods are primarily used. Manual examples of information to be extracted are created by manually filling out empty template slots with text values that are marked and then drag&dropped into empty slots. From these examples, the ML components generate rules and patterns required by NE agents. Experience indicates that sensible results from ML require at least 10 examples per information item and may require upwards of 100 examples for complex information items.

As will be discussed, no one ML technique is suited for all possible types of information, hence a hybrid architecture is required. Additionally, statistical approaches are advocated for generating a spatial model of the relative order in which information items occur within a particular class of documents. This can be a strong aid to disambiguating the results of individual NE Agents.

2.3. Document Types

A practical IE-system must be able to process a range of document types, from free-form text to the more structured documents that constitute most web sites.

2.3.1. Structured and Semi-Structured Documents. The complexity of the information extraction task is fundamentally governed by the characteristics of the documents from which the information is to be extracted. Extraction of information from formal, well-structured documents, where a strong syntax or mark-up exists for the description of salient information, can be achieved with 100% accuracy through the construction of a parser. Such documents are for example well-formed XML documents or automatically generated documents, which have strictly defined fields and field position ordering. Many such documents are to be found on the internet, e.g. online stock quotes, weather reports, travel information, etc.

Conversely, where the source document is completely informal, with no restriction as to where or in what format information is to be written, then the knowledge extraction task is akin to a full-scale natural language understanding application. Typical free-form documents in the industrial field might be online news reports, company white papers, etc.

The middle ground is where many interesting practical applications for IE currently lie. For these types of application, the document can be referred to as semi-formal. This term is taken loosely here to mean a number of things:

- There is some formality in the basic syntax of the document that can be exploited to more reliably identify information occurrences.
- The boundary of an individual entity is reflected in the structure of the document itself.
- The relative position/ordering of occurrences of items of information within the overall text area relevant to an entity (template instance) is predictable and can generally be exploited to aid their recognition.
- The cardinality of a given item of information within a template instance is predictable.

For semi-structured documents, the above types of constraint are not in themselves sufficient to unambiguously locate required information. It is assumed that enough syntactic formality exists to make feasible the creation of high quality NE-components, e.g. based on hand-coded rules. Nevertheless, it is assumed that enough syntactic variation persists to make the construction of perfect NE-components (i.e. 100% precision/recall) an expensive and difficult task. The structural constraints of semi-structured documents provide supporting evidence that should be combined with traditional IE approaches to locating information in free-text. Again this requires some form of hybrid IE architecture.

Note that the assumption that the scope of an occurrence of a template can be ascertained from the structure of a document is a strong one. This implies some syntactic convention for delimiting the start and/or end of an instance (it may even be that each instance is itself a separated document). Moreover, it is assumed that the document area for one template instance does not infringe on that of another. In other words, the problems of identifying how many template instances exist and of determining to which template instance(s) a given feature instance belongs are assumed trivial. This apparently strong assumption is often justified in many real-life applications, such as filtering email messages or newsgroup entries etc.

The class of semi-structured documents is particularly significant as most online documents of commercial interest for IE applications fit this class. Examples include; online product catalogue, monthly financial analyse and report, online CVs and job adverts, online hotel and holiday information, etc.

2.3.2. Normal Documents and Web (HTML) Documents—The Addition of Tags. It goes without saying that the WWW has become the most common information and knowledge resource used for a wide range of business activities. However, limitations of the WWW, particularly in terms of the accuracy of information retrieval, are well known. This has recently prompted the vision of the “Semantic Web” (see www.semanticweb.org) in which the content of online documents is much more rigorously ‘marked-up’ and therefore suitable for machine interpretation. This in turn will enable a whole range of improvements in web-based applications, as well as new possibilities for automated information processing (see [10]). Nevertheless, the migration from today’s informal and chaotic WWW to the well organized Semantic Web will not be easy and is likely to occur in isolated pockets within specific vertical domains. Information Extraction may become of major commercial importance and, indeed, play a major role in the progression toward the Semantic Web, because it allows this migration from text to formal representation within a vertical domain to be carried out with high degrees of automation.

The widely used document format in WWW is Hyper Text Mark-up Language (HTML), which is a simple mark-up language used to create hypertext documents that are portable from one platform to another. Compared to normal (non marked up) documents, Web documents consist of rich tag information which provides additional information about each piece of text, e.g. determines the type of that text. HTML tags are typically not precise enough to uniquely identify information for IE purposes. However, tags provide helpful structure information, which can be exploited to generate more accurate extraction rules. Thus, as previously stated, although some automatically generated web documents may be considered as structured documents, most fall under the category of semi-structured documents.

The recent direction for representation of information on the Web is using XML and XSL (e.g. see www.w3c.org). While XML describe layout independent content, XSL is used to define the presentation of content in a browser. Beyond these wide spread formats, the W3C (World Wide Web Consortium) Organisation is also trying to promote more expressive formats, such as RDF, to more strongly support machine-understandable information in web pages. If online documents are written in XML (or derived, formal formats), standard extraction technique, such as XSL-Patterns, XQL etc., can be used to access desired information. However, most web pages are still written today in HTML, which lacks the formal rigor of XML and therefore makes the task of extracting information much more difficult.

2.3.3. Document Types—A Summary. In summary, documents range from unstructured free text, through semi-structured documents to (usually automatically generated) structured documents. Most documents on the internet are semi-structured. This is the type of document which is of most interest for practical IE applications. Examples of different document types are summarized in Table 1.

3. System Architecture

In this section we introduce the system architecture of our Information Extraction system, CapturePlus. The architecture is a hybrid combining a number of different IE agents, as shown in Fig. 1.

We partition the system into four components: Preprocessor, NE-Agents, TE-Agents and Output Generator, each of which will be briefly described below. A more detailed discussion of the main IE tasks will then follow in subsequent sections.

3.1. Pre-Processing

In CapturePlus, the first step is pre-processing. In general, pre-processing enables a consistent and formatted input that makes the task of finding relevant information and text zones more straightforward and efficient. In this phase, documents are segmented and indexed. In addition, for HTML documents, documents are converted to well-formed XML format.

A more in-depth discussion of the Pre-Processing problem is given in Section 4.

3.2. NE-Agent

In CapturePlus, Named Entity (NE) recognition is carried out by a number of different modules, NE-Agents, extract the content from pre-processed documents using structured information and classical IE methods. Feature extracting is the most important step in the overall IE process, and a number of different and complementary techniques are built into CapturePlus in order to be able to do this task for a wide range of document and content types. Each IE technique is represented as a different type of NE-Agent.

Table 1. Summary of document types.

	Structured	Semi-structured	Free
Normal	Software log files Questionnaire forms	Commercial email (e.g. confirmations of booking, product ordering)	Private emails Fiction (novels)
		Patent applications	
		Project reports	
Online	Stock quote	Product catalogues	Web news
	Online weather forecast	Company home pages	White papers
		Travel booking sites	
		Online job advertisements	

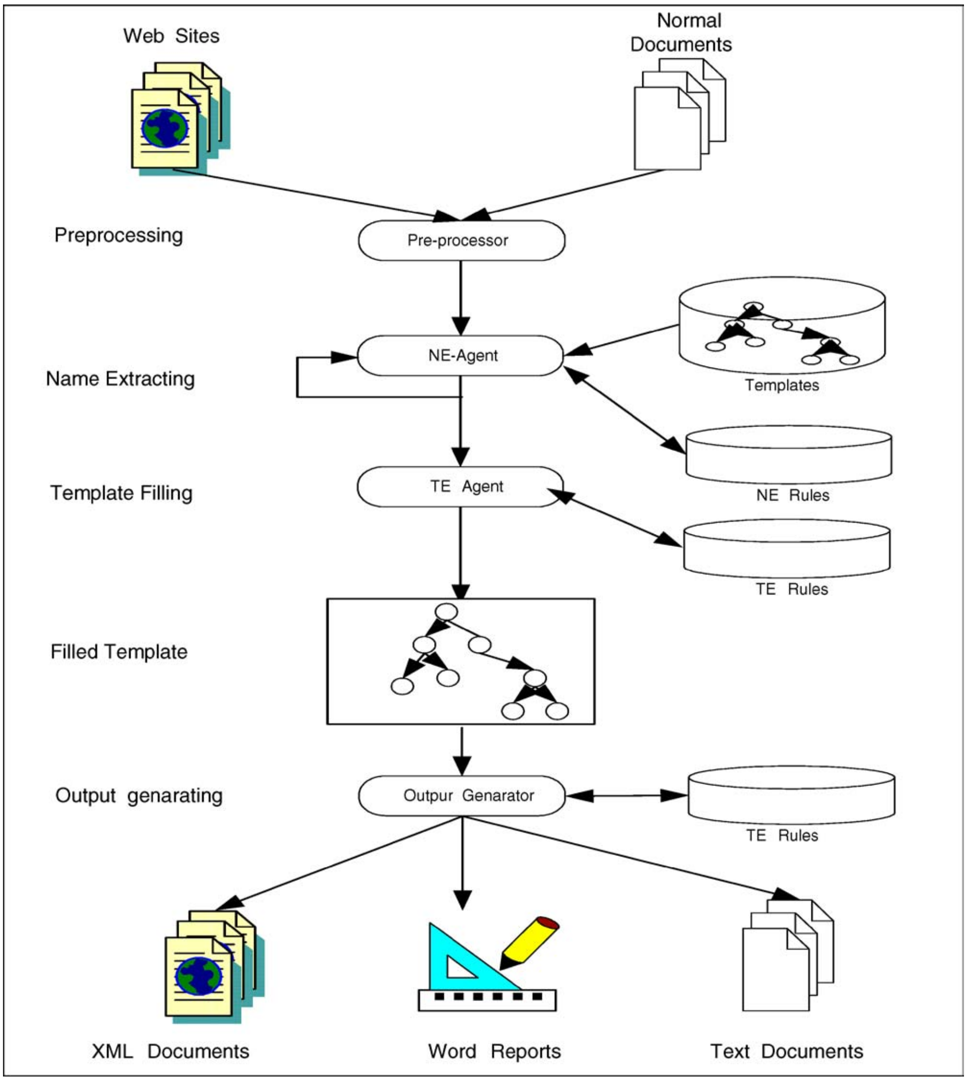


Figure 1. System architecture of CapturePlus.

An NE-Agent instance is created to apply a specific IE technique to a given type of textual information with particular syntactic or semantic properties. Techniques vary from simple keyword matchers through to complex hierarchical expressions and Natural Language grammars.

At run time, populations of NE-Agents can be applied independently. It is also possible to combine some NE-Agents collaboratively to get optimal results, as is illustrated in Fig. 2. NE recognition in CapturePlus actually involves three stages: text zone recognition, parallel contents extraction and result merging. NE Agents that perform text zone recognition are configured to segment an input document into regions which are distinct in terms of the types of content they contain. Note, methods for delimiting text zones can vary depending on document type—in structured or semi-structured documents tags may explicitly delimit relevant text areas, whereas for freeform documents more 'fuzzy' text zone boundaries (e.g. based on keyword distribution) may be used. Text zones provide a limited context in which other NE Agents can search for specific pieces of information.

In order to distribute extraction tasks to corresponding NE-Agents a NE-Scheduler is necessary. This can include dynamic scheduling if more than one NE-Agent is suitable for the same type of information (e.g. the NE-Scheduler may first distribute the task of finding

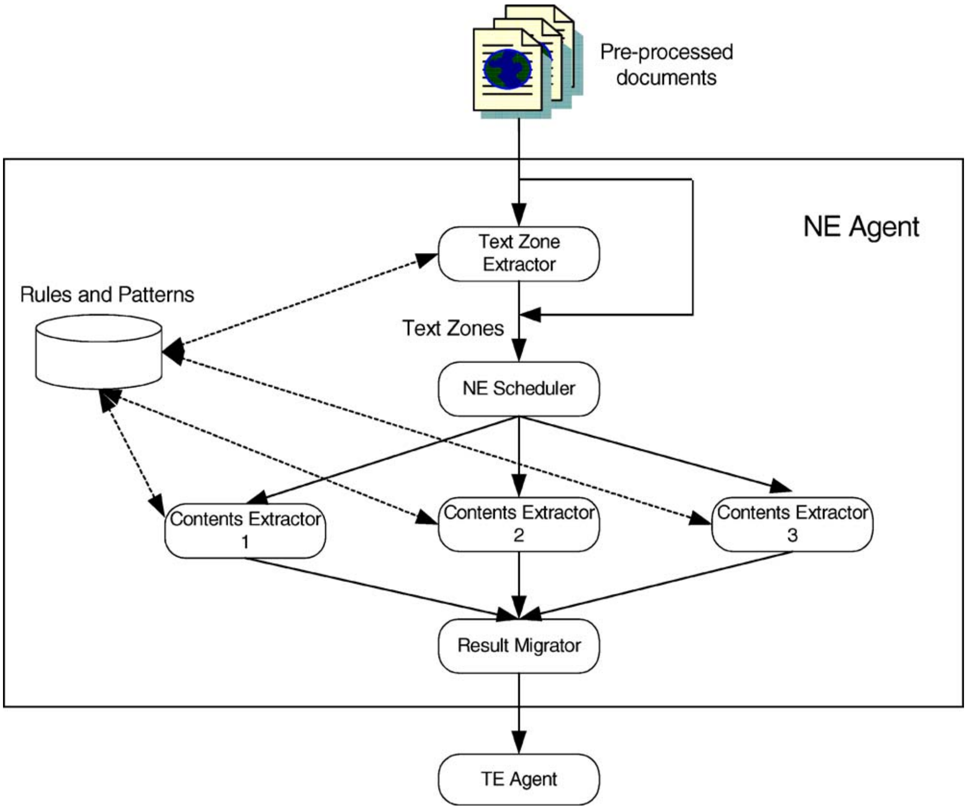


Figure 2. Construction of NE-Agent.

a persons name to an NE-Agent that has reliable syntactic rules, and if this NE-Agent fails, the task may be distributed to a weaker NE-Agent that relies on name lists for matching). Alternatively, multiple NE-Agents may search for the same information in parallel. In this case, the results of NE-Agents can contain overlapping, redundant, or even conflicting values. Therefore, in a final stage results from multiple NE Agents are merged with duplications removed, using a voting algorithm, to produce a final list of possible text values for each different type of information to be extracted. Results also include useful meta data, such as recognition confidence and text position, that enables disambiguation and organisation in the TE-Agent stage.

A more in-depth discussion of the NE recognition problem is given in Section 5.

3.3. *TE-Agent*

In CapturePlus, as for many IE systems, the goal is to fill out a *template* that represents the expected and required content of a particular type of input document. Most IE systems incorporate NE results directly into the template. In contrast, in CapturePlus, the incorporation of NE-Agent results into a template is regulated by a TE-Agent. Beyond simply connecting NE-Agent results to template slots, the TE-Agent plays two major roles in CapturePlus (as described below): controlling multiple instantiation of a given template, and, disambiguating between conflicting NE-Agent results.

Many documents in practical domains consist of more than one template instance. For these documents, template filling is complicated because different occurrences of the same template must be automatically recognised and created. For example, for an online catalogue, the TE-Agent must be able to delimit each separate product and fill out a product description template for each product separately. The TE-Agent can achieve this in two ways; (I) strongly, by using specific NE-Recognisers to find template instance boundaries in text, (II) weakly, by using an optimisation technique to cluster NE-Agent results into candidate templates based on order and proximity. The latter method is time consuming and often delivers only approximate results. However, for most real applications, syntactic delimiters between instances exist and can be exploited—i.e. case (I) is the norm.

Information Extraction in general suffers from one fundamental deficit: it is possible to create NE-Agents that are able to recognise particular types of information with very high accuracy (**Precision**)—for example specifically tailored syntactic rules; it is also possible to create NE-Agents that find a high proportion of all occurrences of a particular type of information (**Recall**)—for example keyword-based matchers; however, it is very hard to create an NE-Agent that simultaneously has very good Recall and Precision.

A solution to the above dilemma is to support populations of many NE-Agents and to employ some combination logic to aggregate their result to produce better overall performance. For example, if many (redundant) NE-Agents are created to recognise the same piece of information, a voting algorithm can be used to select the most likely result and boost overall precision. Conversely, by creating more and more disjoint, high-precision NE-Agents, overall coverage (recall) can be gradually increased. The main practical problem with the above two approaches is the high development costs implied by creating large populations of collaborative NE-Agents. Here, an alternative approach, exploiting the TE-Agent, is proposed based on the observation that spatial relationships between different slots of a template can be used to allow precise knowledge about the value of one slot to be used to disambiguate the value of one or more other slot.

For example, suppose that in a product description, the *product code* and *price* (both numbers) are known to occur in a particular order (e.g. *price* before *product code*). Initial, weak NE-Agents developed for both *price* and *product code* may just recognise numbers and confuse *prices* and *product codes* (high recall poor precision). Knowing that a spatial relationship exists between these slots means that a developer need only improve the precision of just one of these NE-Agents to get overall high precision for *both* slots. For example, if the *price* NE-Agent is strengthened (e.g. to recognise a number in context of a currency), then the *product code* will also be found unambiguously because the TE-Agent can eliminate the false recognition of a *price* as a *product code*. This approach is discussed in more detail in Section 6 and is important because reliance on a TE-Agent to perform disambiguation can lead to a development methodology whereby effort can focus on a subset of critical slots/information items within a template (see also [15]).

So in summary, the experience embodied in the CapturePlus architecture is that a robust approach to information extraction is two-stage, with relatively crude NE-Agent achieving high recall rates followed by a TE-Agent that restores the overall precision. Individual NE-Agent only apply locally within a document and therefore, in many cases, do not have enough input information to be able to make a wholly accurate decision as to the meaning of a piece of text (at least without resorting to a more profound Natural Language Processing analysis of the raw text). Accuracy requires a combination of local syntactic clues (that can be captured easily in the NE-Agent) combined with the broader dependency that a piece of information has with other types of information within the document (as captured by the TE-Agent).

The TE-Agent of CapturePlus therefore relies on knowledge about expected spatial dependencies between different types of information within a document. Normally, dependence between different slots is complicated, so that a probabilistic, spatial model is required. However, such a model can be created automatically using statistical approaches, assuming a number of manually filled out templates can be provided. Extensive and diverse empirical studies have shown that this allows generally around a 10% boost to the Precision of the overall IE system, without adversely effecting the Recall, with respect to the results of NE-Agents alone.

3.4. Output Document Generation

CapturePlus can generate arbitrary ASCII formats from the intermediate template instances it creates by applying a scripting language to these results. As this approach is highly similar to standard approaches in other areas, e.g. XSL-T transformation for XML documents, this part of the overall IE process is not discussed in detail in this paper.

4. Pre-Processing

Having given a brief description of the overall CapturePlus architecture in the preceding sections, this and following sections will treat each of the main Information Extraction processing stages separately and in more depth.

The first major IE task is pre-processing. The generic role of pre-processing can be defined as: *reformatting and normalising input documents in order to make the task of identifying relevant occurrences of information items easier and more efficient.* Typical pre-processing tasks might include: converting all documents to a single format and file type (e.g. ASCII), structuring a document into segments (e.g. sections->paragraphs->sentences), building indexing structures, e.g. for rapid access of individual words, normalising and tagging individual words (e.g. part of speech and stemming), etc.

4.1. Pre-Processing HTML

Specifically for IE for internet applications, input documents are usually HTML documents. A major pre-processing task for such documents is to standardise the mark-up tag structure of the document, e.g. by conversion to standard XML format.

As mentioned in Section 3.2, internet documents with explicitly tagged structures allow for more accurate IE because specific NE agents can be localised to search in text zones delimited by specific tags. Although HTML documents have structure defined by HTML tags, these are not well suited for IE purposes because HTML documents are not well-formed (e.g. tags do not always have closing tags). The syntactic structure of HTML documents cannot always be formally parsed by a structured parser (such as SGML).

The eXtensible Mark-up Language (XML) is the universal format for structured documents and data on the Web. XML documents have a complete and well formed hierarchical structure specified by start tags and end tags. The validity of an XML document can be verified with reference to a DTD or XML-Schema that defines constrains on allowable instances of a given type of XML document (i.e. constraints on the cardinality and structure of possible hierarchical structures). Hence it is advantageous to convert HTML to well formed XML as part of the pre-processing task. If this is achieved, then the hierarchical structure of a HTML document can be used, e.g. to define different text zones or to identify the beginning and end of multiple template instances within a website.

As Kushmerick [18] points out, emerging standards such as XML will simplify the extraction of structured information from heterogeneous sources. Knoblock and Minton [24] observed also, that wrapper induction algorithms may be able to use XML as a source of supervised training data. The extension considered here is to apply suitable pre-processing to HTML documents so that the (implicit) structured information in HTML sites can be parsed with help of XML-Parser to more rigorously determine the structure of the content. Normally a HTML site can not be parsed by an XML-parser directly. However, the conversion from HTML to XML is relatively simple: e.g. eliminate all single tags and enclose all attribute values in quotes ("). This conversion can be made automatically. For CapturePlus, the standard open source software program TIDY (provided by W3C organization—http://www.w3.org/People/Raggett/tidy) was used to convert HTML to XML automatically. TIDY is able to fix a wide range of formatting problems and syntax errors.

Note TIDY does includes some limitations, such as not adequately converting special characters (such as '&') to allowable XML notation (i.e.\amp). Hence some additional clean up specific to the purposes of CapturePlus needed to be performed.

Note also that the documents generated from HTML pages are not real XML-documents because they do not have DTD or XML-schema. However, such converted documents are well-formed and therefore can be parsed by an XML-parser. Hence they are sufficient for the purposes of Information Extraction.

4.2. Related Work and Further Aspects of Pre-Processing

In general, pre-processing for HTML documents has two major tasks: tokenisation and clean-up. In the first task, a system decomposes HTML into different tokens for later processing. A token can be thought of as a syntactic unit of text. STALKER [10], for example, uses a token-based Wrapper and therefore needs to tokenise HTML texts into a sequence of tokens (e.g., words, numbers, HTML tags, etc.).

The second task concerns the conversion of HTML into a well-formed document, as already discussed above. For example, XWRAP [12] also cleans up HTML documents in a pre-processing phase. The bad tags are deleted and the uncompleted tags are recovered. The cleaned HTML document is fed to a source language compliant tree parser.

For CapturePlus, use of standard tools made available by W3C was considered important to allow for portability. However, the pre-processing of CapturePlus, and for similar systems, currently has some limitations. Possible improvements include:

- Normalising the text within a HTML document, e.g. by using linguistic methods such as stemming.
- Augmenting the original tag structure with additional tagged mark-up which represents some a priori domain knowledge.

The latter actually implies a two-staged approach to Information Extraction—in pre-processing use simple IE agents, such as predefined name lists etc. to do a simple mark up of the original text in order to improve the accuracy of more complex IE carried out by the NE Agents. For example, ontologies of places, products types etc. could be developed and applied to help pre-process websites in this way and create a more structured and normalised input to NE-Agents. Examples of systems that have tried this approach include SRV [12], which annotates HTML source documents with predefined feature tags. Similarly, WHISK [5] requires input documents to be additionally segmented to aid in retrieval of multiple bits of information. However, for both these systems, some manual effort is required at the pre-processing stage which may detract from their applicability in real-world applications.

5. NE-Agent

Following pre-processing, the next step is the extraction of individual items of information via NE-Agents. As introduced in Section 3.2, this stage of the IE process has three sub-steps carried out by different types of NE-Agent: Text Zone recognition, content extraction and result merging. Each of these stages is described separately below.

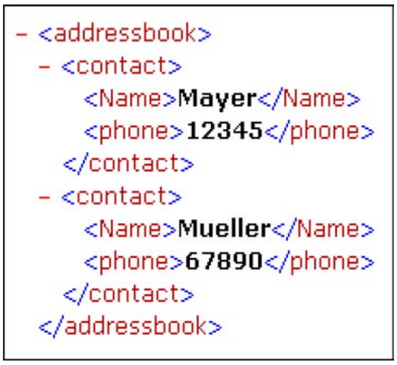
5.1. Text Zone Recognition

The input of Text Zone recognition is a pre-processed document. Text Zones are defined as part of a document, where useful contents can be extracted. Text Zone detection is usually dependent on actions carried out in pre-processing, e.g. to standardise HTML structure into XML structure. Text Zone detection depends on the syntactic properties of the text, such as significant start or end delimiters, which do not belong to the underlying content, but can help the system to locate the useful content. A Text zone may also be defined simply by exploiting a standard structure of a document (e.g. section->paragraph->sentence) as produced in pre-processing.

If the inputs to the IE system are internet pages, another opportunity for text zone recognition is possible. For HTML document XSL-patterns can be used, in order to get the text zones enclosed by tags. A requirement for text zone recognition in this manner is that the HTML files are already converted to well-formed XML documents in the pre-processing phase.

5.1.1. XSL Patterns. XSL-patterns are used to find element nodes in XML document (see e.g. http://www.w3.org/TR/1998/WD-xsl-19981216 for details). An XSL pattern describes a path to a type of branch of an XML tree, which in turn consists of a set of hierarchical nodes. To a first approximation, the syntax used for XSL patterns resembles that used to specify paths on a disk drive (though complications to include, e.g. attribute constraints, are also possible—thus providing more accurate selection criteria). For example, given the XML document showed in Fig. 3, the pattern "*addressbook/contact/phone*" selects "*phone*" elements that appear beneath a "*contact*" element, which itself appears beneath an "addressbook" element. In other words "*12345*" and "*67890*" would be selected. XSL-patterns also support wildcard such as "∗". For instance, the results after executing XSL-pattern "*addressbook/contact/*∗" could be all values in sub-elements "*Name*" and "*phone*".

Each item of information to be extracted by NE-Agents can have one or more XSL-pattern(s) associated with it to locate the relevant text zones. For some item of information, the text found by the XSL-pattern corresponds exactly to the to-be-extracted Value. Such item of information are often elements in table or some



```
- <addressbook>
  - <contact>
      <Name>Mayer</Name>
      <phone>12345</phone>
    </contact>
  - <contact>
      <Name>Mueller</Name>
      <phone>67890</phone>
    </contact>
  </addressbook>
```

Figure 3. An XML example.

other HTML structure generated automatically by a Robot in the web server. In this case, the task of NE-Extraction is already finished.

5.1.2. Automatic Learning XSL Patterns. Manually generating XSL Patterns is time consuming and error-prone. A XSL-pattern in a real application can be quite cumbersome. For instance, Fig. 4 shows a real XSL-pattern to detect the current temperature value in the CNN weather forecast web sites. Moreover, website structures can change overtime, requiring an update to XSL-pattern. Development of a machine learning algorithm to perform an automatic generation of XSL-patterns from given examples, is highly beneficial. In this section we will introduce a supervised bottom-up learning algorithm used in CapturePlus to generate XSL-patterns automatically. A simple worked examples is also given, so that the principle can be better understood.

Automated training of XSL patterns can be driven by a user manually marking text values of interest within an XML file. A standard XML parser can be used to determine the actual XML nodes specified by marked examples. Because the examples are typically defined as text positions, all XML nodes need to be annotated with position information. For each XML node returned by the Parser, an XSL pattern can be generated. The generated XSL pattern is processed bottom-up: from the actual XML text node, parent nodes are accessed recursively until the root of the XML document is reached. An XSL pattern is then generated as the path of node names with their attributes. Once all initial XSL patterns are generated in this way, a supervised learning algorithm is used to induce the patterns. The algorithm is shown in Fig. 5.

The system induces rules bottom-up. For each iteration, all overlapping XSL patterns are produced, based on new training examples and previously generated XSL patterns. A generalization of the overlap is produced by introducing wild cards and/or deleting attributes into XSL paths of the same length. The overlapping pattern covers all content extracted by both input patterns. For example, the overlapping pattern of *html/body/table/td/tr* and *html/body/table/td/font* is *html/body/table/td/**. The metric used to accept the overlapping pattern is the expected error rate of the new pattern, given by the following formula, where n is the number of extractions made on the training document set by the new pattern and e is the number of errors among those extractions (i.e. matches not previously given as examples).

$$\textit{Error rate} = (e + 1)/(n + 1)$$

```
html/body[@bgcolor = "#FFFFFF" ][@link = "#000099" ][@vlink = "#666666" ]/form/table/tr/td[@rowspan = "2" ][@width = "163"
][@valign = "TOP" ]/table[@border = "0" ][@cellspacing = "0" ][@cellpadding = "0" ][@width = "163" ]/tr/td[@width = "163"
][@bgcolor = "#FFFFFF" ]/table[@width = "163" ][@bgcolor = "#FFFFFF" ][@align = "top" ][@cellspacing = "0" ][@cellpadding =
"0" ][@border = "0" ]/tr/td[@width = "460" ][@valign = "TOP" ]/table[@width = "460" ][@cellspacing = "0" ][@cellpadding = "0"
][@border = "0" ]/tr[@valign = "TOP" ]/td[@width = "460" ]/table[@width = "460" ][@cellpadding = "0" ][@cellspacing = "1"
][@border = "0" ][@bgcolor = "#FFFFFF" ]/tr/td[@width = "190" ][@valign = "TOP" ][@bgcolor = "#DDDDDD" ]/table[@width =
"190" ][@cellpadding = "0" ][@cellspacing = "0" ][@border = "0" ]/tr[@bgcolor = "#FF9900" ]/td[@width = "180" ][@valign = "TOP"
]/table[@width = "180" ][@cellpadding = "0" ][@cellspacing = "0" ]/tr/td[@valign = "bottom" ]/span[@class = "redtemps" ]
```

Figure 4. A real XSL-pattern to extract a temperature value from CNN weather web site.

```
Initialise the result set Training to be null
WHILE Example Set is not empty
        Get an example from Example Set
        Generate an initial XSL pattern for this example and set it as input_pattern
        FOR each trained_pattern in Training
New_Pattern = GetOverlappingPattern (trained_pattern, input_pattern)
Calculate the error rate E_i for New_Pattern
        END FOR
        IF all E_i > Threshold THEN         Add the input_pattern to Training
        ELSE                                Add New_Pattern with the lowest E_i to Training
        END IF
        Reorganize the Training to content only disjunctive patterns
        Delete example from Example Set
END WHILE
```

Figure 5. Algorithm for learning of XSL-patterns.

If the error rate is lower than a defined threshold, the overlapping pattern will be added as a trained pattern, otherwise the specific XSL-pattern for the new training example is set as a new trained pattern. After adding a new XSL-pattern, the set of all trained patterns is reorganised to make sure that there are only disjunctive patterns in the trained pattern set. In Fig. 6 we give an example to illustrate how XSL patterns are inductive generated.

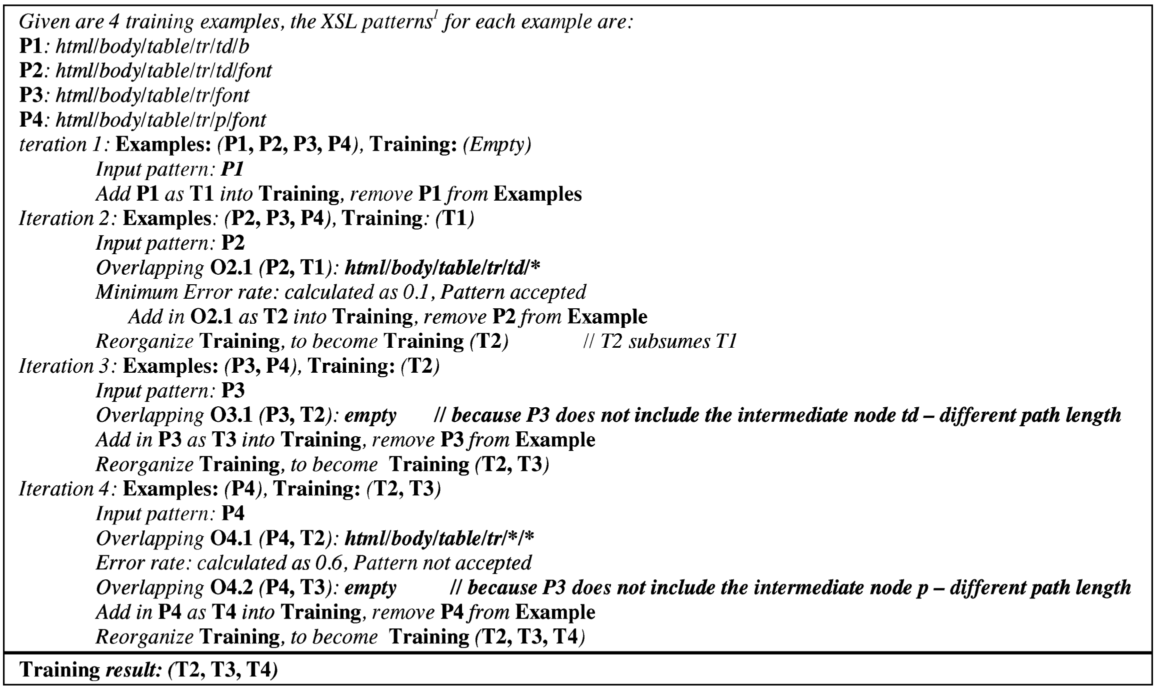
5.1.3. Related Work. Information Extraction from Web Pages is a topical IE task. Techniques, which try to extract useful information from HTML/XML, are commonly referred to as wrappers. There are two basic types of wrappers: Tag-based and structure-based wrapper.

The Tag-based wrappers treat a HTML-document as a flat text string with mark-up tags. Such wrappers try to identify tags-properties around underlying content and use such properties to perform extraction. For structure-based wrappers, HTML-documents are tree-structures and extraction rules are learnt and applied in a similar way to navigating a file directory path.

STALKER [11] is a well known example of a Tag-based wrapper using the principles of finite automata. For STALKER a document is a sequence of tokens S (e.g. words, numbers, HTML tags, etc.) A key idea underlying this work is that the extraction rules can be based on "landmarks" (i.e., groups of consecutive tokens) that enable a wrapper to locate the content of x within the content of p. For instance, a STALKER wrapper rule can be defined as following:

$$R1 = \mathit{SkipTo}(\mathit{Name})\mathit{SkipTo}(\langle b\rangle) \text{ or}$$
$$R2 = \mathit{SkipTo}(\mathit{NamePunctuationHtmlTag})$$

R1 has the meaning "ignore everything until you find a *Name* landmark, and then, again, ignore everything until you find ⟨b⟩", while *R2* is interpreted as "ignore all tokens until you find a 3-token landmark that consists of the token *Name*, immediately followed by a *punctuation* symbol and an *HTML* tag." Such rules can be learned and generated automatically with an inductive learning algorithm. The learning algorithm of STALKER iterates until there are no more uncovered positive examples. When all the positive examples are covered, STALKER returns the solution, which consists of an ordered list of all learned disjuncts. According to the author, the function for learning perfect disjuncts is a greedy algorithm: it generates an initial set



```
Given are 4 training examples, the XSL patterns[1] for each example are:
P1: html/body/table/tr/td/b
P2: html/body/table/tr/td/font
P3: html/body/table/tr/font
P4: html/body/table/tr/p/font
teration 1: Examples: (P1, P2, P3, P4), Training: (Empty)
        Input pattern: P1
        Add P1 as T1 into Training, remove P1 from Examples
Iteration 2: Examples: (P2, P3, P4), Training: (T1)
        Input pattern: P2
        Overlapping O2.1 (P2, T1): html/body/table/tr/td/*
        Minimum Error rate: calculated as 0.1, Pattern accepted
            Add in O2.1 as T2 into Training, remove P2 from Example
        Reorganize Training, to become Training (T2)          // T2 subsumes T1
Iteration 3: Examples: (P3, P4), Training: (T2)
        Input pattern: P3
        Overlapping O3.1 (P3, T2): empty     // because P3 does not include the intermediate node td – different path length
        Add in P3 as T3 into Training, remove P3 from Example
        Reorganize Training, to become  Training (T2, T3)
Iteration 4: Examples: (P4), Training: (T2, T3)
        Input pattern: P4
        Overlapping O4.1 (P4, T2): html/body/table/tr/*/*
        Error rate: calculated as 0.6, Pattern not accepted
        Overlapping O4.2 (P4, T3): empty       // because P3 does not include the intermediate node p – different path length
        Add in P4 as T4 into Training, remove P4 from Example
        Reorganize Training, to become  Training (T2, T3, T4)
Training result: (T2, T3, T4)
```

Figure 6. A worked example for learning XSL-pattern from given examples.

of candidates and repeatedly selects and refines the best refining candidate until it either finds a perfect disjunct, or runs out of candidates.

STALKER considers HTML as only as flat set of strings with some useful tokens (tags). This feature of STALKER allows a very high extraction speed, because the SkipTo operation can be implemented as a straightforward pattern matcher. Conversely, in CapturePlus, and similar structure-based wrappers, a document has to be at first parsed by an XML parser before the XSL-patterns can be applied. The XSL pattern matching is more complicated and therefore more computational intensive than the algorithm used in STALKER. However, the STALKER approach breaks down when the underlying tree structure of the document is highly complicated. Todays, professional web pages contain high degrees of layout and formatting information. Simple flat SkipTo operation will tend to extract too many false instances within such documents. Moreover, CapturePlus uses not only hierarchical tags but also attribute information, which can help to locate the underlying content more precisely. Similar structure-based wrapper systems include e.g. [5, 13, 17,18].

An excellent example of a structure-based wrapper is XWRAP [13]. As for CapturePlus, XWRAP relies on pre-processing to clean up HTML documents and a tree parser to generate the document structure. The next step is region (text zone) extraction, to identify regions of interest, as specified through interaction with a user and defined in terms of the parse tree. For example, a region extraction rule in XWRAP is: "*HTML.BODY.TABLE[0].TR[0].TD[4].TABLE[2]*"—note that this rule format is similar but not conformant to standard XSL syntax. Finally, Semantic Token extraction is performed to create a set of semantic token extraction rules that can be used to locate and extract information of interest in similar Web documents from the same web site. The output is a comma-delimited file, containing all the element type and element value pairs of interest.

XWRAP is an interactive system which, unlike CapturePlus, does not provide machine leaning to help create general pattern rules. The user has to interact with the IE-system to define text zones and accept extracted contents. Furthermore, XWRAP the use of its own rule format makes XWRAP less easy to combine with standard XML tools. In short, the usage of XSL in CapturePlus is seen as a major advantage for the general applicability of the tool and approach.

5.2. Content Extraction

After detecting text zones, content extraction can be performed within relevant text zones only, which significantly boosts overall precision in most practical applications.

There is no fundamental restriction on the range or type of content that can be handled by an IE system. Typical example include; specifically formatted numbers (price, telephone, ...) dates, names (of people, places, products, ...), narrative description, object properties etc. Because of the variety of types of content, there is no general technique, which can be used to locate and extract all relevant information. Hence, in more successful IE systems, such as CapturePlus, content extraction uses various techniques; from simple full text extraction, extraction based on fuzzy matching, extraction with regular expressions, through to complicated trainable token based rules.

In the following subsections, a number of different IE techniques are described and compared with respect to their relative strengths and weaknesses.

5.2.1. Basic IE Techniques—Full Text Extraction, Extracting with Similarity and Regular Expressions. The simplest form of extraction is full text searching, whereby exact matches in text can be located. This technique is rapid and simple and may be well suited to types of information that can be represented as lists of predefined strings, such as lists of names (e.g. places, products in a product family,...) or constants (e.g. currencies).

The downside of full text searching is that (I) it often leads to false matches and, conversely, (II) the success of matching on desired information is often sensitive to the exact wording in the original text. To solve these problems, a common extension to this technique is to use a thesaurus which connects words via relationships (typically "*synonyms*", though other relationships such as acronym, abbreviation, etc. are common). The role of the thesaurus is to allow *query expansion*, i.e. when a given term is used for full text search, its related terms are also used. Hence, in this way, a search for "USA" might match on "United States of America", and indeed any other term synonymous with "USA". Alternatively, a synonym set itself may be used as a search term, hence ⟨*Currency*⟩ might return a match on "*dollar*", "*Euro*", "*GPD*", etc.

Provision of a thesaurus is equivalent to providing a type of "background knowledge" or even "ontology" in other systems [5]. Such background knowledge can be a powerful aid for high quality information extraction—it provides much better coverage (Recall) when searching but can also be used to disambiguate words with multiple meanings (Precision). Word sense disambiguation of this type is achieved by selecting the most plausible meaning for a given word in a text by consideration of the words that have occurred in the same text region and their (thesaurus) relationships to possible meanings of the word in focus [25].

One of the main draw back with using a thesaurus is the cost of creating the "background knowledge". Synonyms can be created manually. However, a manual creation of all words and their relationships to one another is highly laborious. One solution is to apply statistical techniques, such as co-location analysis, to generate candidate synonyms based on large text volumes. Alternatively, synonyms can be generated by observing manual tasks, such as template filling, and creating synonyms from examples. The effort can be avoided by using an existing thesaurus, at least to provide general word relations (domain specific terminology may still need to be created for the IE application). Alternatively, terms can be mined from existing databases, where available.

In CapturePlus, two thesauri are used: an automatically created domain specific thesaurus and a standard thesaurus. The former is created and maintained by an example-driven approach, monitoring human correction to the results of the IE system. The latter is a standard NLP component (WordNet, http://www.cogsci.princeton.edu/∼wn/). Additionally, a thesaurus editor provides an interactive GUI to allow users to add, delete, and change words and relations.

One problem with keyword driven matching is that there are often many different variations of the same word (morphology). One general solution to this is to match keyword strings using a similarity measure that is tolerant to variants, rather than using an exact match. For example, extracting the word "*product*" with similarity will find content such as "*product*", "*products*", "*productivity*", etc. Extraction based on similarity is particularly suitable for documents which may contain noise. For example, if electronic documents are generated by scanning and by OCR (Optical Character Recognition), errors can occur during the conversion process. Original characters within a word may be replaced by similarly written characters, e.g. '0'⟨-⟩'D', '3'⟨-⟩'8', 'm'⟨-⟩'rn', etc. Extraction based on similarity can overcome such problems. Similarity of words can be calculated with a general purpose "fuzzy match" algorithm. Alternatively, linguistic models of morphology can be used in conjunction with a thesaurus to more formally represent word variants—such linguistic techniques are less generic and also more noise sensitive. On the other hand, a general purpose similarity algorithm will inevitably produce some superficial similarity matches between strings that have no underlying semantic relationship. Therefore, in conclusion, fuzzy keyword matching gives improved robustness (recall), particularly in applications including noise, but may have a detrimental effect on accuracy/precision.

Keyword spotting, with or without a similarity matching algorithm, has fundamental limitations. There persists the problem that the same word may have different meanings depending on how it is used. Moreover, much useful information, particularly numeric information, cannot be captured by keywords alone. The next level up in sophistication therefore is to apply regular expressions.

A regular expression is a general pattern that describes a set of string instances. Regular expressions are suitable for matching content with significant syntactic properties (such as number, date, time, price, etc.). Regular expressions are constructed by using various operators to combine smaller expressions (alternatives, options, repetitive sequences, ...). For example, the rule "$[a, p]m[0-9]+ : [0-9]+$" could be used to extract times represented in a format such as "AM 12:45". Regular expressions are widely used in the Unix world as searching/replacing tool and can be applied for information extraction in a similar way. In general, the processing of regular expressions is very fast, because the regular expression can be compiled to a finite state transition network and no background knowledge or lexicon is needed. However, highly nested expressions and expressions containing large numbers of options can still be computationally expensive.

Because regular expressions are based on FSA (Finite State Automata), learning and automatic generating regular expressions are theoretical possible, though typically only very simple regular expressions can be learnt due to the complexity of the syntax involved. Moreover, regular expressions have some limitations in terms of the expressive power, including;

- Searching in context is only partially handled
- Explicit representation of hierarchical grammars is hard (i.e. using the same basic pattern to construct many more complex expressions)

Table 2. Summarization of simple extraction techniques.

	Typically suitable for	Advantages	Disadvantages
Full Text (incl. Thesauri)	Domain specific names, constants etc.	Performs medium precision and recall. Lookup speed is very fast.	Significant effort required to generate domain-specific thesauri
Similarity	Contents with noise, morphological variants	Domain independence, no background knowledge, very high recall	Often poor precision, may be slow lookup
Regular Expressions	Contents with significant syntactic properties, numeric information	Good precision, can be generated through training examples	Expressions are often hard to read. Some expressive limitations (context and hierarchy)

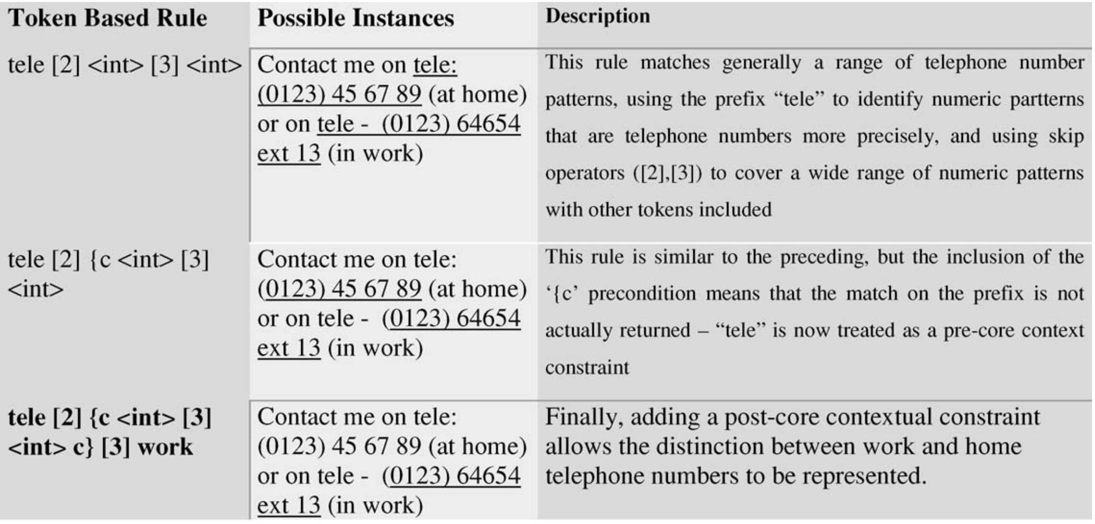
- Regular expressions cannot elegantly be combined with other search techniques, such as keyword lookup and thesauri

Table 2 summarizes the basic extraction methods discussed in this section with respect to their applicability, advantages and disadvantages.

5.2.2. Extraction Using Token Based Rules. In CapturePlus, a rule based approach to information extraction was developed in order to (I) make expressions more easy to learn from example (simplified syntax) and (II) allow more explicit handling of contextual constraints. The Token-Based Rule (TBR) approach to information extraction was initially developed by DFKI (The German Research Centre for Artificial Intelligence GmbH - http://www.dfki.de/).

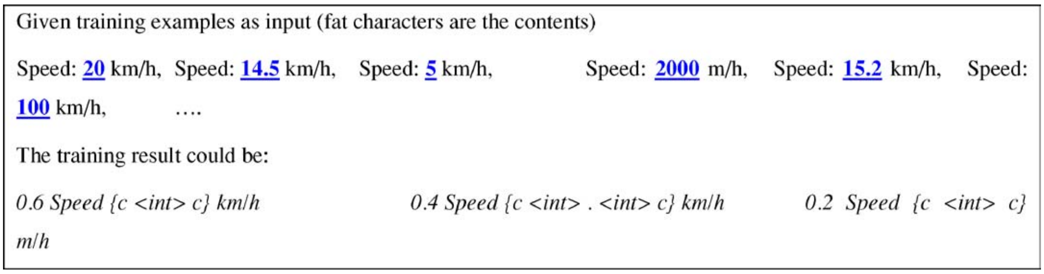
Token-based rules consist of three parts: *pre-core*, *core* and *post-core*. While the core describes the underlying content to be extracted, the pre- and post-core define expected information before and/or after this content (respectively), thus allowing for the definition of more precise rules. The expression of all three parts of a TBR are constructed as sequences of tokens, whereby tokens are grouped into; digits, alphabetic characters, or 'special characters (i.e. all other characters). White space characters are treated as separators between other tokens. Additionally, skip operators can be included that match up to a given maximum of to-be-ignored tokens. Finally, the special tokens '{c' and 'c}' in a TBR rule denote the separation of pre-core and core, and core and post-core, respectively. Figure 7 shows some simplified, artificial examples of the usage of TBR.

Extraction using TBRs proceeds as follows: At first, documents are tokenised into the previously stated string types (digit sequences, alphabetic sequences, sequences of other non-white-space characters; After tokenising, the system interprets token-based rules as



Token Based Rule	Possible Instances	Description
tele [2] <int> [3] <int>	Contact me on tele: (0123) 45 67 89 (at home) or on tele - (0123) 64654 ext 13 (in work)	This rule matches generally a range of telephone number patterns, using the prefix "tele" to identify numeric partterns that are telephone numbers more precisely, and using skip operators ([2],[3]) to cover a wide range of numeric patterns with other tokens included
tele [2] {c <int> [3] <int>	Contact me on tele: (0123) 45 67 89 (at home) or on tele - (0123) 64654 ext 13 (in work)	This rule is similar to the preceding, but the inclusion of the '{c' precondition means that the match on the prefix is not actually returned – "tele" is now treated as a pre-core context constraint
tele [2] {c <int> [3] <int> c} [3] work	Contact me on tele: (0123) 45 67 89 (at home) or on tele - (0123) 64654 ext 13 (in work)	Finally, adding a post-core contextual constraint allows the distinction between work and home telephone numbers to be represented.

Figure 7. Token-based rules (TBR).



Given training examples as input (fat characters are the contents)

Speed: **20** km/h, Speed: **14.5** km/h, Speed: **5** km/h, Speed: **2000** m/h, Speed: **15.2** km/h, Speed: **100** km/h, ...

The training result could be:

0.6 Speed {c <int> c} km/h *0.4 Speed {c <int> . <int> c} km/h* *0.2 Speed {c <int> c} m/h*

Figure 8. Example of learning token-based rules.

finite state automata and matches pattern within a defined text zone. As soon as the matching is successful, the Extractor outputs result with position information and relevance values (relevance values are established during training and specified as part of the TBR itself, as will be described below).

Token-based rules can be learned and generated automatically from examples. The simplified view of document content that the tokenisation provides makes automated learning more straightforward. The Trainer uses an inductive supervised training algorithm. The trainer generates at first a rule only with core pattern. The trainer then tries to specialize the rule by adding pre- or post-core constraints and to generalize the rule by adding skip operators. Modified rules are then evaluated over all given examples. The output of training is a set of token-based rules that cover all examples and a set of relevancies for the rules. Figure 8 shows a simple example of learning token-based rules. The number at the begin of rule depicts the estimated accuracy of the rule w.r.t. training example coverage, the following string defines then pre-core, core and post-core.

Token-based rules are typical syntactic rules. They work similarly to as regular expression but combine the search for a core match with contexts (pre-core and post-core). Such rules can be widely used to extract a wide range of information from websites. The TBR are particularly well suited to extraction of:

- Numeric information followed by units
- Values of formally or informally tagged fields
- Information inside commonly occurring sentence structures.

5.2.3. Hierarchical Grammars. The relative simplicity of TBR expressions occurs because tokenisation of the original document is performed prior to applying or learning TBR. Tokenisation of this type is common within natural language processing. A logical extension is to allow rules themselves to act as tokenisers. In this way, hierarchical grammars are constructed whereby rules for complex or specialized types of information are constructed from more basic elements. For example, the following rule;

$$\{c < \text{TELEPHONE\_NUMBER} > c\}[3]\text{work}$$

expresses the specialized concept <WORK_TELEPHONE_NUMBER> in terms of the more basic concept <TELEPHONE_NUMBER>. In other word, <WORK_TELEPHONE_NUMBER> can only be recognized if instances of <TELEPHONE_NUMBER> are first found and, for each of these instances, the corresponding text is normalized to be replaced by just a symbolic name for the concept <TELEPHONE_NUMBER>. This type of hierarchical relationship allows arbitrarily structured grammars to be constructed and hence very complex types of information to be extracted. However, disadvantages include; complexity of learning and construction, as well as relatively slow runtime application. A more detailed discussion of hierarchical information extraction is given in [26].

5.2.4. Text Zone Classification for Information Extraction. The difference between Information Extraction (IE) and Information Retrieval (IR) has already been discussed in Section 2. However, some IR techniques can be used for some IE tasks. In particular, classification of text zones within a document, e.g. a particular tagged field of a website, can be performed by adapting standard IR techniques. For example, IR could be used to extract a short description of a product from a catalogue, a plain text description about travel destinations, job adverts within a certain field, etc. A precondition is that the segmentation of the website into logical text zones has been already carried out (see Section 5.1).

In Information Retrieval (IR) there is no lack of classification models: for example, Boolean, vector space, probabilistic, and fuzzy models are all well known and widely used [27]. Most, including the approach adopted in CapturePlus, base text classification on a match against a weighted set of keywords. In the training phase of CapturePlus, the Keywords Trainer first eliminates stop words from the examples and then analyses statistical information about the remaining words. As is typical, keywords are generated if the frequency of keyword occurrence with the given positive examples is greater than the expected frequency (e.g. as measured against a control corpus). The output of training is a set of keywords with relevance. In the extraction phase, the Keywords Extractor classifies a text zone based on the number and weighting of relevant keywords that are to be found within that text zone. For example, a keyword-based query for recognizing holiday advertisements might be {{0.40, "*travel*"}, {0.37, "*accommodation*"}, {...}}. Relevance value R can be calculated using the following formula, where r_i is relevance of a keyword found in the text zone, N is total number of keywords.

$$R = \sum r_i / N$$

If the relevance is greater than a defined threshold, the text zone is output as a result with position and relevance information. Note, while such classification is generally based purely on identifying relevant keywords, other types of NE-Agent can also be used to provide the 'symptoms' for a classification. For example, a regular expression for finding distances might be part of the classification for holiday advertisements, as a typical description of a hotel includes descriptions of how far the hotel is from the beach, airport, town centre, etc. Using more complex NE Agents to tokenise text descriptions prior to classification is a promising approach to providing much more accurate classification (e.g. see [22]). The trade-off is typically one of runtime performance—the more sophisticated the tokenisation, the more computationally expensive is the overall classification task.

5.2.5. Related Work. The range of techniques that have been applied for content extraction are too numerous to adequately review here. The basic approaches have ranged from keyword matching, syntactic rules and Natural Language Processing (NLP), through to advanced techniques which combine basic techniques with well-known artificial intelligence or machine learning methods. The best that can be done here is to point out some noteworthy examples. Issues of particular practical interest will include; manual effort required to construct the IE system, type of information that can be extracted, and general performance quality and speed.

Syntactic rules are widely used for extracting information which has significant syntactic properties. This technique is popular because of the clear definition of pattern matchers (ease of maintenance), language independence and good time performance. No dictionaries or lexicons are generally required (though in some cases such resources are useful—see [6]). Syntactic rules are suitable for such content as time, speed, currency, name, number etc. However, syntactic rules work best only in relatively local contexts and can have performance limitations for more complex information; typically a syntactic rule results in either high precision but poor recall (too specialized rules) or poor precision but high recall (too generalized rules). Various constraints can enhance extraction performance considerably. Much like the Token Based Rules discussed in Section 5.2.2, such constraints can include pre-patterns and post-pattern [12, 16, 23], or semantic constraints [6].

Natural language processing has been an active area of related research for many decades. Many advanced IE technologies rely on NLP techniques. NLP techniques are used not only for NE tasks, but also for TE and TR construction. The MUC researchers developed IE systems primarily based on NLP techniques [14]. NLP techniques are especially suitable for free text or semi-structured texts written in a natural (free form) way. However, the goal of simulating human understanding of natural text is still a long way off and many fundamental unsolved problems in NLP still persist. Thus, care must be taken in applying NLP techniques in practical IE applications—often the application does not demand the full power of theoretical NLP in order to produce acceptable results and at the same time limitations in NLP may impact on the overall system. As an example, an open problem is so called Word Sense Disambiguation (WSD) [25], whose aim is to identify the correct sense of a word in a particular context, among all of its senses defined in a dictionary or a thesaurus. WSD could be used to pretokenise documents and thus simplify the overall IE task. However, errors caused by WSD problem will be propagated to later processing. To work around this problem almost all advanced algorithms use a combination of various syntactic and semantic constraints to identify the contents of interest.

Applying NLP to information extraction tasks also implies significant construction overheads as a number of knowledge resources (full grammars, dictionaries, morphology rules, etc.) may need to be provided. Machine learning may help reduce the amount of manual cost involved. A promising approach of this type is to apply Case-Based Reasoning (CBR) to NLP/IE problems. In CBR, reasoning and classification is performed purely by reference to *similar*, previously encountered examples. Learning is simplified to the task of adding missing examples to memory.

Riloff [22] pioneered a practical system (CIRCUS) using CBR for information extraction. A case is represented as a structure with five slots: signatures, perpetrators, victims, targets, and instruments. During training, each document in the training corpus is manually converted into a set of cases. Each document is represented as a set of cases, one for each sentence that produced at least one useful piece of information. The resulting case base contains thousands of natural language contexts from hundreds of texts. The heart of the algorithm is its ability to accurately judge the relevancy of new cases at a sentence level. The application in [22] is for document retrieval, hence content extraction at a sentence level using CBR was shown to give improved performance for document retrieval tasks.

Actually, CBR techniques can be used in several IE tasks. At the lowest level, CBR can be used to extract context with more flexibility than syntactic rules—syntactic rules are good for capturing the most commonly occurring patterns whereas CBR can use the principle of precedent to capture rarely occurring patterns as well as the general cases. At the template filling level, filled templates may provide a case base which can be referred to in order to disambiguate NE-Agent results or to supply missing (default) slot values for similar future template instances. However, in practice, some problems in using CBR have to be faced: first, an adequate, and often application specific, case base must be constructed—this may be large and require significant manual effort even if cases are created by just marking examples. Secondly, defining an adequate and balanced metric for similarity comparison may be application specific. Thirdly, unless significant effort into designing adequate indexing strategies is carried out, the runtime cost of comparing new cases to a large case base may mean that the processing speed in a CBR system is normally not satisfactory for large text volumes. Nevertheless, the simplicity and flexibility of CBR make it an interesting candidate for many IE tasks.

Alternative machine learning approaches try to create a more compact model of text content to provide high runtime performance. Interesting in this respect are regression trees (model trees), one of the family of divide-and-conquer algorithms that includes categorical decision trees (such as C4.5, [28]).

Recently, a system called RoboTag has been developed which uses decision trees to learn the location of slot-fillers in a document [19]. RoboTag uses C4.5 to learn decision trees which predict whether a given place in the text is the start or end of a slot-filler value—in this way information to be extracted is located. Prior to learning and applying decision trees, texts are preprocessed, to segment them into text zones and to tokenise text based on morphological analysis and lexical lookup. The trees consider the token for which the decision is being made along with a fixed number of tokens around it. In the training phase some parameters have to be set in order to improve tagging performance. In the extraction phase RoboTag applies the learnt decision tree classifiers to a new text to produce a list of potential begin and end tags for each piece of information to be extracted. Each potential begin and end tag produced by the decision tree also has a confidence rating. A scoring function is used to evaluate the relative merits of different sets of begin-end parings to provide the overall optimal content extraction results.

The critical factor in C4.5 is to select relevant training features. The extraction based on of C4.5 alone is very quick. However, the total processing speed is depended on the selection of features and the preprocessing of features. Robotag has e.g. four features: Token Type, POS, Location and Semantic Type. Such features are based on other basic IE technologies (such as token rules, POS tagging etc.). However, such features are selected only for defined problems and defined document class, making the selection of features individual to a specific solution and the solution itself less portable. Additionally, to be able to learn reliable decision trees, relatively large numbers of examples may be required.

Other approaches to information extraction make use of statistical machine learning techniques such as Hidden Markov Models [29] or artificial neural networks [23]. For example, Merkl describes in [23] the use of hierarchical self-organising maps to extract key information from documents. A self-organising map (SOM) is an unsupervised artificial neural network, consisting of layers of neurons, starting with an input layer, proceeding through intermediate layers to an output layer. As SOMs are unsupervised, they possess self-learning qualities that use a genetic algorithm style approach to perpetuate the best performing layers. In [23], it is pointed out that this approach may be particularly well suited to text classification problems (see Section 5.2.4). However, neural networks have rarely been applied to extraction of the fine-grained information typical of IE tasks [12]. A reason may be the need for relatively large numbers of training examples to get accurate identification/classification with a neural network.

So in summary, from this brief overview of related work it can again be emphasized that no one technique has yet to emerge as the silver bullet for all IE tasks. NLP techniques may provide the power for most complex scenarios, at high construction costs. Symbolic machine learning techniques may provide the key to building practical, application specific content extractors, whereas statistical machine learning approaches may be complementary in terms of providing text classification capabilities. Only in a hybrid architecture, such as CapturePlus, can the merits of these disparate approaches be combined to provide an overall solution to content extraction problems.

5.3. Results Merging

In a multi-NE-Agent system, each underlying NE-Agent will operate in independence and create a list of possible document positions for occurrences of the corresponding type of information. These disparate results need to be merged. In some text zone classification cases, this requires combining results from different NE-Agents via a logical operator ("AND" and "OR"). For "AND" operation, intersection of all instances are selected (a hotel description is a text zone which contains information on the number of stars of the hotel AND its name), while for "OR" operation, union of instances are calculated (a room description includes facilities such as has-TV OR has-mini-bar OR has-double-bed OR . . .)

Having resolved logical dependencies, the next stage is to remove redundancy. Many different NE-Agents may be attached to the *same* slot in a template and because extraction algorithms work autonomously, results can contain redundant information. Within CapturePlus, possible redundant results can be:

1. *Duplication*: Two results have the same position and length. In this case, one of instance has to be dropped.
2. *Overlapping*: Extracted areas are overlapped. In this case, the overlapped area may be merged. The new result is added into the final list and the original two are then deleted.
3. *Containment*: One extracted area contains the other. Here only the bigger one is kept and the other is dropped (special case of 1)

Following these rules, a single results list is produced from all result lists of each NE-Agent attached to a slot. This final results list represents all possible instances of the template slot. This is passed on to the next processing step (the TE-Agent), where from the basic information extraction results, one or more unambiguous instantiations of the overall template will be produced.

6. TE-Agent—Template Filling

After NE-Agent for content extraction, extracted information is fed into the TE-agent. In the phase of template filling, the TE-agent tries to assign the right information instances to the correct template slots. There at least three tasks of a TE-agent:

1. Extracted instances have to be assigned to the slots in template. If the relationship between instance and corresponding slots is one-to-one, the assignment is trivial, otherwise some intelligent sorting and configuration is needed.
2. Normally NE-Agents can not extract information perfectly, that is, with 100% precision and recall. TE-agent can be used as a post-processor to enhance the overall results. In particular, precision is enhanced here by intelligent removal of false hits during a conflict-resolution and disambiguation procedure.
3. If the document has multi-instance, the TE-agent has to duplicate the template and correctly assign the content to each template instance.

The following sections concentrate primarily on the second task above. A new approach to disambiguation is introduced based on spatial reasoning about the structure of incoming documents. This approach will be compared to related work.

6.1. The Spatial Model

The task of slot value disambiguation requires that a TE-Agent drop the false instances and keep the right one. To solve this problem, interrelationships between different slots of a template can generally be exploited.

In particular, for many types of document and particularly many websites, a predictable layout to content within the document exists which means that the relative spatial relationships between slots can be used to resolve ambiguities—i.e. a piece of information found in the wrong place w.r.t. other pieces of information may be removed. Nevertheless, even if all individual items of information are correctly found, if the boundary between template instances cannot be accurately ascribed and information instances accordingly grouped then the overall quality of information extraction will be severely degraded. For example, if there are multiple products in an online product catalogue, then it is critical that each price extracted is paired to the appropriate product name and description. Finding boundaries is not always a trivial task, but again, spatial relationship can strongly assist in this task, e.g. by identifying which items of information occur early or late in typical descriptions helps define a possible transition from one template instance to the next. This section described the contribution of the spatial model deployed in CapturePlus to the overall information extraction task.

For the class of document that is of interest here (structured and semi-structured—such as websites), significant spatial relationships are expected between the individual slots of a given template—in other words, the order in which the information instances occur in the original document is not completely random. Furthermore, the capture of these spatial relationships is a task that can be automated and achieved in a non-intrusive manner—e.g. based on observing manual Drag&Drop operations to fill out templates based on example documents.

The model of the spatial relationships between the instances of template slot values corresponds to the profile of the TEAgent of CapturePlus. Note that the user who, in effect, trains the TE-Agent need not be an expert user with respect to the information extraction technology; they merely must be capable of reliably carrying out the template filling task manually.

Several types of possible spatial relationship can be captured between two slots A and B. Examples include:

- DirectBefore(A, B)—i.e. an instance of content B comes directly before content A
- DirectAfter(A, B)—i.e. an instance of content B comes directly before content A
- Before(A, B)—i.e. an instance of content B comes somewhere before content A
- After(A, B)—i.e. an instance of content B comes somewhere before content A
- Contain (A, B)—i.e. an instance of content B is only a part of content A

These relationships are created based on the statistical analysis of multiple manually filled templates (for the experiments described in later sections, the TE profile is generated from 100 manually filled templates—though often considerably less examples would suffice). The relationships are further sub-divided into:

- Constraints—i.e. spatial relationships that should not be invalidated (they were observed always to hold)
- (Weighted) Preferences—relationships that are not always valid but have a statistically significant frequency. These relationships are typically represented as a numeric weight <1.0.

An example of part of the TE-profile fore the Job Ad experiment (see Section 7) is displayed below. This example is derived from online Job Advertisement. Note that the After/Before/Containment relationships generally result in constraints where as the more specific Direct Before/Direct After relationships generally result in weighted preferences: this reflects a persistent overall structure for the description of each template instance although the detailed ordering of individual content instances is less well defined. This observation justifies the classification of the Online Job Advertisement document type as being semi-structured.

```
Constraints:
After(JobTitle, Salary)                              // Salary always comes somewhere after JobTitle
After(City, State)                                   // State always comes somewhere after City
...
Preferences:
Direct After (JobTitle, Salary, 0.333)               // Salary often comes directly after JobTitle
Direct After (JobTitle, Required_Experience, 0.259)  // Required_Experience often comes directly after JobTitle
Direct After (City, State, 0.815)                    // State usually comes directly after City
...
```

6.2. Template Filling with Greedy Voting Algorithm

The preceding section briefly outlined the spatial model for template slot dependencies that can be derived in CapturePlus. This section introduces a local optimisation algorithm for controlling template filling based on this spatial model.

The TE-Profile, described above can be used in order to filter the results communicated from a collection of NE-Agents—i.e. it is to be used to rule-out falsely identified template slot value instances while preserving the correctly identified slot values. There are a number of possible ways in which the statistically derived information could be exploited. The algorithm tested here resolves conflicting slot values based on a greedy voting algorithm.

The first stage is to identify conflicts. For example, three types of commonly occurring conflict are:

- ***Ambiguity***—*Multiple instances for a given slot have been found, although it is known in the template that the slot has cardinality 1 (more generally, more instances are found than allowed by cardinality constraints)*
- ***Unexpected Ordering***—*The instances of two different slots occur in an order that is contradictory with respect to known constraints of the TE Profile.*
- ***Multiple Interpretation***—*A given part of the text is used as the basis for the value of more than one template slot.*

As an example, assume the following tupple reflects the order (with respect to document position) in which slot values for a given template instance occur within the document (as communicated by the NE agents):

(Salary_1, JobTitle_1, Required_Experience_1, City_1, Salary_2)

The conflicts that are generated (with respect to the previously given example TE-Profile) are:

Conflict1(Salary_1, JobTitle_1) -> Unexpected Ordering—Salary instance must come after Title instance
Conflict2(Salary_1, Salary_2) -> Ambiguity—Multiple instances of same slot (Salary)

Conflicts can be resolved by eliminating one of the conflicting slot values. This is achieved by allowing all slot values outside of the given conflict to vote for each of the conflicting instances. Namely;

Votes($F_{conf,i}$) $= \sum$ *Vote*($F_{conf,i}$, F_j), *where* F_j *is a feature instance not in the current conflict.*

The values for Vote($F_{conf,i}$, F_j) are as follows:

- -1 *if the spatial relationship between* $F_{conf,i}$ *and* F_j *violates a known constraint*
- $+1$ *if the spatial relationship between* $F_{conf,i}$ *and* F_j *conforms to a known constraint*
- $-W$ *if the spatial relationship between* $F_{conf,i}$ *and* F_j *violates a known Preference of weight* W
- $+W$ *if the spatial relationship between* $F_{conf,i}$ *and* F_j *conforms to a known Preference of weight* W
- *else* 0.

For example, for Conflict1

Votes(Salary_1) = Vote (Salary_1, Reguired_Experience_Year_1) + Vote (Salary_1, City_1) + Vote (Salary_1, Salary_2)
= (0.0) + (0.0) + (0.0) = 0.0
Votes(Title_1) = Vote (Title_1, Reg_Exp_Year_1) + Vote (Title_1, City_1) + Vote (Title_1, Salary_2)
= (0.26 + 0.35) + (0.0) + (1.0–0.33) = 1.28

The feature instance that receives least votes is eliminated from the selected conflict, and all other conflicts. In other words, Salary_1 would be eliminated on the strength of Conflict1. As this also resolves Conflict2, no further action need be taken, i.e. the resultant slot value set after the filtering by the TE-agent would be:

(Title_1, Required_Experience_1, City_1, Salary_2)

As shown in the above example, the elimination of a slot value as a result of resolving one conflict is propagated to other conflicts. This means that the performance of the TE-Agent is sensitive to the order in which conflicts are resolved, even if the next conflict to be solved is selected randomly The slot value eliminated by resolving the first conflict is not necessarily the best slot value to be eliminated from other untried conflicts. In this sense, the optimisation algorithm is "greedy". This provides good runtime performance but does not guarantee an optimal global conflict resolution . Nevertheless, as will be discussed below, this simple approach has been empirically demonstrated to give significant improvements in the overall accuracy of the information extraction system.

6.3. Related Work

Several well known statistical techniques for classification and recognition are based on Bayesian statistics. Bayesian statistics provides a way of combining multiple pieces of evidence to reach an overall decision. For example, Freitag used this technique to extract information based on prediction of position and length [12]; specifically, to identify the name of the speaker in a seminar announcement. This problem can be modelled as a collection of competing hypotheses, where each hypothesis ($H_{\mathrm{p;k}}$) represents that a particular fragment of text (i.e. a sequence of k tokens) at a given text position (p) gives the speaker's name.

The evaluation of Bayes [12] on seminar announcements is relatively simplified, in that the size of documents is small, appropriate information to be extracted does not vary significantly in length, and each document contains only one description of a seminar announcement. Under such restrictions the Bayes model is able to predict the position and length of tokens to a high degree of accuracy. However, it is not proven that the approach is scaleable to less well structured documents and for more varied pieces of information to be extracted.

Hidden Markov Models (HMMs) are a type of probabilistic finite state machine, and a well-developed probabilistic tool for modelling sequences of observations. As such, HMMs have strong potential for carrying out the type of spatial reasoning advocated here to improve the template filling task. There are several systems using HMMs for information extraction tasks [22, 27]. Such systems are often simplistic in that they do not tend to automatically derive complex spatial models; either they rely on using one state per classification and/or they use manually assembled FST models. They tend to focus also only on localised text structure rather than relative position in the overall document.

In general, in order to build a HMM for IE, it must first be decided how many states the model should contain, and what transitions between states should be allowed. Each relevant word in the training data is assigned its own state. Each state is associated with the class label of its word tokens. To bound each instance in text, a transition is placed from the start state to the first state of each training instance, as well as between the last state of each training instance and the end state. Model structure can then be learned and refined automatically from data, starting with either a maximally-specific model, using techniques such as Bayesian model merging.

The HMM system deployed by, Seymore [20] is a good illustration of this general scheme and demonstrates some complexity in that it focuses on learning the structure of one HMM to extract *all* the relevant fields, taking into account field sequence. The HMM is used for information extraction from research paper headers. Each state of the HMM is associated with a class to be extracted, such as title, author or affiliation. Each state matches words from a "class-specific unigram distribution". Such state specific word lists and the state transition probabilities can be automatically learned from training data. In order to label a new header with classes, the words from the header are treated as observations of different classes and the most globally optimal interpretation is retained. However, this system must first do some preprocessing of documents to perform tokenisation e.g. <ABSTRACT>, <EMAIL> etc.). In fact. The role of the HMM is purely that of the template filling agent—i.e. to model ordering relationships between tokens rather than to extract the tokens themselves. As such, this approach strongly mirrors the approach used in CapturePlus.

The major advantage of HMM for modelling order relation of TE-slots is that a more global optimisation algorithm for slot filling is produced. That is, the selection of the best sub-sequence or combination of slot values is calculated with respect of all relevant slot values. Consequently, this causes a potential problem of time performance, and, more significantly, a potential problem in terms of the number of training examples required. Evaluations show satisfactory results with very simple templates, which contains less than

5 slots. However, for over 50 slots per template, the number of training examples and runtime performance may become prohibitive for practical applications.

By contrast, the greedy algorithm used in Capture-Plus presents no significant run-time problems and can operate on a less precise spatial model, which can be derived from relatively few manually filled out templates. This approach has allowed the technique to be applied to internet IE applications where a single detail template may even include several hundred slots.

7. Evaluations and Discussion

The preceding sections have gone into some technical detail concerning the separate stages of a combined information extraction systems and the alternative techniques available to realise each stage. It is useful to reiterate at this stage the two main thrusts of this paper, namely;

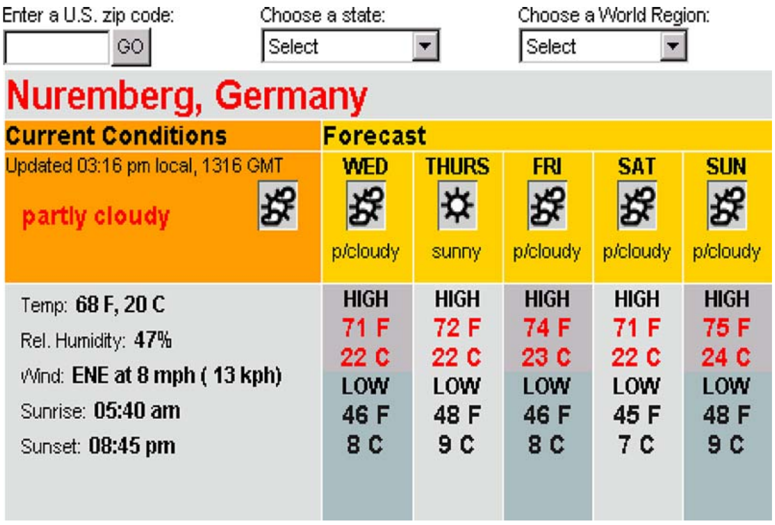
1. The *general* problem of IE from websites provides a comprehensive challenge to state-of-the-art IE systems
2. A state-of-the-art IE system requires necessarily a multi-levelled, hybrid architecture.

This section attempts to illustrate these claims further through empirical studies on IE-from-internet scenarios.

7.1. Experiments Description

Most Web sites can be classified as being one of two types of document: structured and semi-structured text. Structured texts are formatted so uniformly that a few examples are enough to learn perfect text extraction rules. This class of text will be represented in this experimental evaluation by pages originally taken from the CNN weather forecast web pages (http://www.cnn.com/WEATHER/). The CNN Weather Forecast domain consists of Web sites for various domestic and international cities with a four-day weather forecast. A sample page is showed in Fig. 9. 10 examples of these web pages were used for training and 100 examples for testing.

Semi-structured texts have fairly stereotyped information but are not rigidly formatted. Experiments for this type of website are reported for two example online scenarios: (Accommodation) Rental Ads and Last Minute Holidays. The Rental Ads domain was collected from the Seattle Times online-classified ads, downloaded from: http://www.isi.edu/∼muslea/RISE/repository.html. For the empirical study, the template defined for the WHISK IE system was used [5]. The template consists of three main slots: *neighbourhood, price* and (number) *bedrooms*. As for WHISK [5], here 400 examples of rental ads were used for training and a further 400 examples used for testing. The Last Minute Holidays Services was collected from the Web site of lastminute.com (http://www.lastminute.com). Templates



Nuremberg, Germany

Current Conditions

Updated 03:16 pm local, 1316 GMT

partly cloudy

Temp: 68 F, 20 C

Rel. Humidity: 47%

Wind: ENE at 8 mph (13 kph)

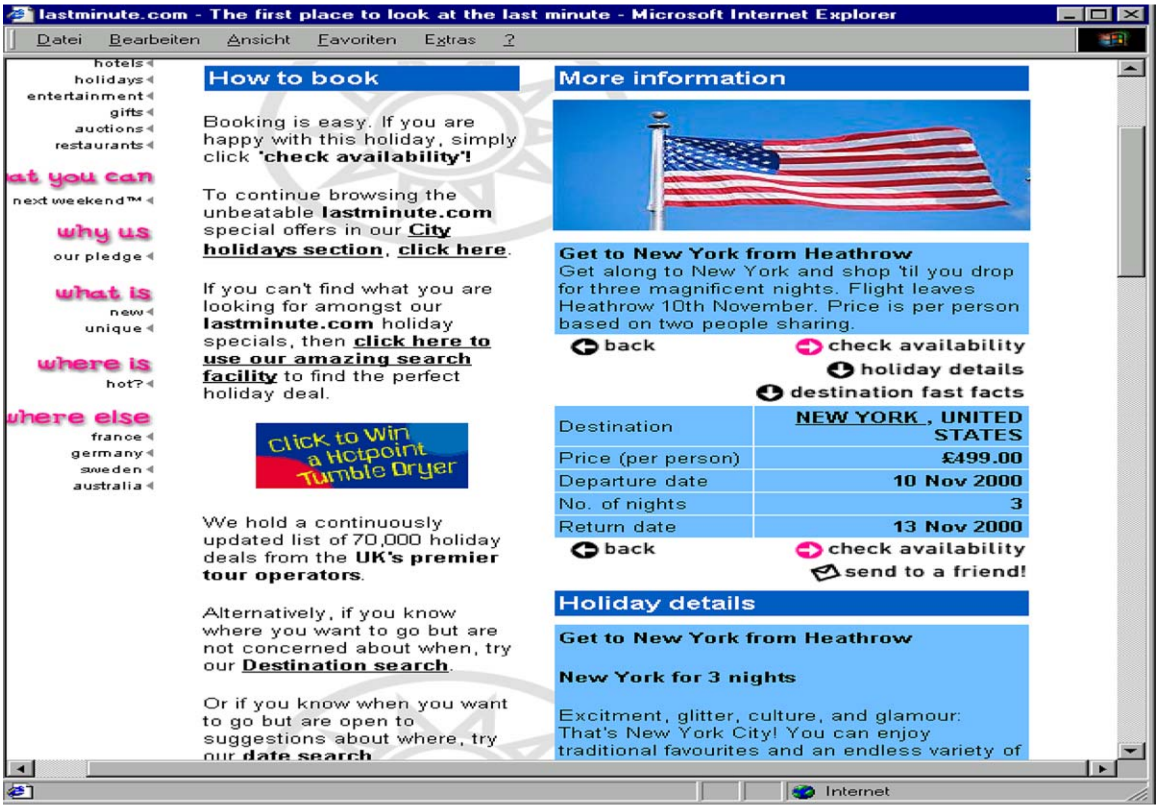
Sunrise: 05:40 am

Sunset: 08:45 pm

Forecast

WED	THURS	FRI	SAT	SUN
p/cloudy	sunny	p/cloudy	p/cloudy	p/cloudy
HIGH	HIGH	HIGH	HIGH	HIGH
71 F	72 F	74 F	71 F	75 F
22 C	22 C	23 C	22 C	24 C
LOW	LOW	LOW	LOW	LOW
46 F	48 F	46 F	45 F	48 F
8 C	9 C	8 C	7 C	9 C

Figure 9. Sample page of CNN weather report.


lastminute.com - The first place to look at the last minute - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
hotels
holidays
entertainment
gifts
auctions
restaurants
at you can
next weekend™
why us
our pledge
what is
new
unique
where is
hot?
where else
france
germany
sweden
australia
Click to Win a Hotpoint Tumble Dryer
Internet

How to book

Booking is easy. If you are happy with this holiday, simply click **'check availability'!**

To continue browsing the unbeatable **lastminute.com** special offers in our **City holidays section**, **click here**.

If you can't find what you are looking for amongst our **lastminute.com** holiday specials, then **click here to use our amazing search facility** to find the perfect holiday deal.

We hold a continuously updated list of 70,000 holiday deals from the **UK's premier tour operators**.

Alternatively, if you know where you want to go but are not concerned about when, try our **Destination search**.

Or if you know when you want to go but are open to suggestions about where, try our **date search**

More information

Get to New York from Heathrow
Get along to New York and shop 'til you drop for three magnificent nights. Flight leaves Heathrow 10th November. Price is per person based on two people sharing.

back | check availability
holiday details
destination fast facts

Destination	**NEW YORK , UNITED STATES**
Price (per person)	**£499.00**
Departure date	**10 Nov 2000**
No. of nights	**3**
Return date	**13 Nov 2000**

back | check availability
send to a friend!

Holiday details

Get to New York from Heathrow

New York for 3 nights

Excitment, glitter, culture, and glamour: That's New York City! You can enjoy traditional favourites and an endless variety of

Figure 10. Example page of Lastminute.com

were more expansive (>20 slots) and slots include: *names of offers, price range*, a *short description, travel details* (departure and arrival dates,...), additional information to *children and infants*, etc. . A sample page is illustrated in Fig. 10. For Last Minute Holidays the training set has 40 examples, while the testing set has 100 samples.

Table 3 shows some representative results of the experimentation using CapturePlus on these datasets. For the CNN Weather Forecast domain, after just 2 training examples, the system gets 100% recall and 100% precision. In Rental Ads the recall and precision are continually improved with more examples, reaching 95 and 98% respectively. For Last Minute Holidays the recall is 91% and precision is 98% with 20 examples. With 40 examples the recall is improved to 97%, but the precision goes down to 96% (evidence of slight over generalisation in the learning algorithm in this case).

Table 3. Experiments result.

Example name	Examples	Precision (%)	Recall (%)
CNN Weather	2	100	100
Rental Ads	50	87	88
	400	98	95
Last Minute Holiday	20	98	91
	40	96	97

The above results are taken after the whole IE process is carried out (i.e. NE Agents and TE-Agent). Deeper analysis identified the situations where particular NE-agents are especially important.

In the CNN Weather Forecast, training with one example can produce perfect results, as the web pages are generated automatically by a robot (fixed structure) and the values are simple and explicitly tagged. Because the required values are exactly and unambiguously enclosed by tags (that is, the whole texts of an XML node), no information extraction *within* XML fields is required for this application. All that is required is: pre-processing (to produce normalised XML from the HTML) and the induction of appropriate XSL-patterns for identifying the right tags. Some ambiguity in the

XSL-patterns exist, e.g. the four temperatures for the coming four days, have the same XSL pattern. However, they are ordered in a strict sequence. In this case, the TE-Agent plays the final crucial disambiguation role. Coupled in this way, these three standard components of CapturePlus allow a 100% effective parser for these web sites to be derived with minimal manual configuration (i.e. with just 2 manually filled out templates).

In contrast, in the Rental Ads application, learning XSL patterns has no relevance because all information is in one XML node. NE-Agents based on keyword recognition and Token Based Rules (TBRs) play the lead role here. In this case the extraction is equivalent to Information Extraction from normal (not HTML-coded) documents. In particular, continued addition of examples leads to continued improvement of the TBRs and keyword recognisers. For the slot "neighboured", the keyword recogniser creates a simple list of all neighbourhood names. For slots "price and "bedroom", the TBR extractor is required. The TBR for the slot "price" is created almost perfectly after 5 examples as "$ <int>". For the slot "bedroom", however, the TBR is at first learnt very crudely (i.e. over generalised—see below). Hence, the recall reaches already 99% with 50 examples, while the precision is only 33%. With 300 examples, the precision climbs to 88% and then stays more or less constant at this level.

More interesting is the Last Minute Holidays domain. Some slots are extracted purely using XSL patterns (e.g. price), because they have unique XSL patterns. Other slots require a combination of both XSL patterns *and* TBR or keyword-based NE Agents (e.g. short description, destination->city, destination ->country) as neither in isolation is sufficient. In this way, the travel-offer websites are an ideal example of semi-structured document whereby explicit (HTML or XML) tags help select relevant regions but additional focussed searching is required to identify exact content. Indeed, for some of the holiday add slots, unambiguously extraction can only be achieved after TE-Agent processing (such as additional information about children OR infants), because they have both identical XSL patterns and keywords but different spatial location (for example, additional information about children is always before infants). Hence, to tackle this problem a genuine mix of techniques is required.

The final evaluation covered here was aimed at measuring the effectiveness of the TE-Agent and was based on the rental-Ad data set. For this experiment, both NE-Agent and TE-Agent are trained automatically. The TBR extractor was used to train and extract the NE values. Rules are generated with an inductive supervised learning algorithm, as described in Section 5.2.2. As for WHISK [5] we used a training set of 400 examples and a test set of a further 400 examples. In this experiment the recall and precision are improved with more examples. Recall begins at 88.4% from 50 examples and climbs to 95.0% with 400 examples. Precision starts at 87.0% from 50 examples and reaches to 97.7% when 400 examples (see Table 3). These overall results are marginally better than those achieved by the original WHISK system.

To understand the overall results in more detail, precision and recall measures were taken after the NE-Agent stage as well as after the TE-Agent stage. The results are presented in Fig. 11. What is demonstrated is that the NE-Agents typically give good recall but poor precision at low numbers of training examples. This loss of precision is largely resolved by the TE-Agent meaning that CapturePlus attains relative good performance overall, even for relatively low numbers of training examples. Deep analysis reveals that the spatial model used by the TE-Agent becomes unchanged after 100 examples. Therefore continued improvement can be wholly attributed to improvement in the NE-Agents, particularly w.r.t. precision. Indeed, further analysis of the overall NE-Agent performance reveals that improvement can largely be attributed to a single agent: after 100 training examples, the precision of the NE-Agents for *Neighbourhood* and *Price* is already 90%, while the NE-Agent for (number) *bedrooms* reaches only 34% precision. In the document, there is normally an acronym of bedrooms (such as "bd", "bdrm" etc.) after the to be extracted number. Most of the improvement is therefore based on (I) learning the list of all possible bedroom acronyms and (II) using this list in the (post-core) context constraints of TBR rules to avoid overly general rules.

Note, the original WHISK system [5] avoided the above problems by providing bedroom acronyms as (manually) predefined background knowledge. This indicates two lessons learnt from this application for more effective deployment of CapturePlus, and IE systems in general—i.e. to speed up the rate of automated learning:

- Where available, use application specific terminology in a pre-processing stage to tokenise/normalise the documents

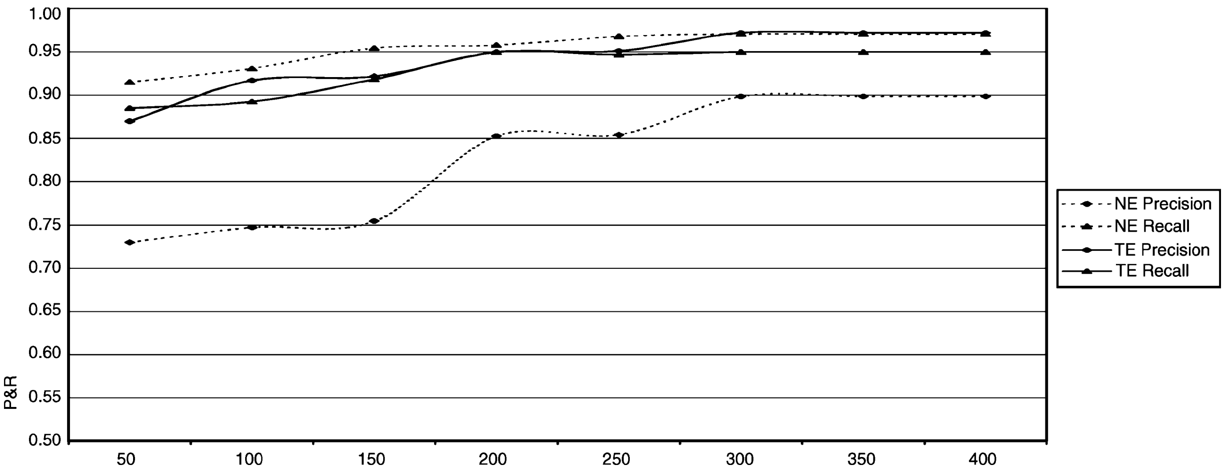


Figure 11. Learning Curve for the Rental ad domain.

- And/Or, carefully select NE-Agents to hierarchically decompose the learning task (e.g. have a keyword-based agent to learn acronyms and a TBR agent that depends on this agent to learn more complex patterns)

7.2. *Discussion*

To reiterate, because of the wide variety of content type and level of formality in websites, a hybrid IE system is required to provide a comprehensive solution to information extraction from the web. This view has been backed up by the experimental results briefly described in the preceding section. This implies that a knowledge engineer responsible for setting up such an IE system must carry out many diverse tasks, including:

- Defining the template structure to be filled out from each web site
- Determine required pre-processing tasks (e.g. format conversion, segmentation, tokenisation, ...)
- Select the right (combination) of NE-Agent types for each relevant type of information
- Manually Create or train the search profile of each NE-Agent—i.e. the lists of keywords, regular expressions, context rules, etc. that the NE-Agent will use to identify relevant information in text
- Establish an NE-scheduler that will distribute at run time the extraction task to underlying NE-agents according to the search profile definitions.
- Create a TE-Agent to adjudicate over potentially ambiguous template slot values assigned by NE-Agents.

Machine learning plays an important role in this overall process by allowing a knowledge engineer to create complex matching rules, not through direct manual encoding, but through provision of training examples. However, care is needed to match the appropriate IE technique to the appropriate type of information, otherwise prohibitively large numbers of training examples are required in order to attain acceptable performance. Hence, there is a very strong need to provide a methodology for configuring web-based IE systems and to provide metrics for each type of Agent in the IE system to determine when and in what configuration they should be applied. Indeed, a new type of NE-Scheduler could be envisaged that not only distributes tasks at runtime but also during the training phases: Such an NE-Scheduler could at first analysis the feature of given training examples and then decide to choose which training algorithm to apply. For example, the relative diversity and frequencies of keywords with training examples for a given type of information might determine whether rule-based or keyword-based patterns, or some mixture thereof, are appropriate to recognise occurrences of the information type. Additionally, over time, this new NE-Scheduler would use test results to direct further development to those parts of the overall system that are currently the "weakest link". As a step in this direction, a brief summary of the strengths, weaknesses and applicability of the main techniques used in CapturePlus and introduced in this paper are given below.

- *XSL-Patterns*: The training algorithm for XSL-pattern uses a supervised bottom-up machine leaning approach. The number of required examples depends on their quality—for strongly structured web sites, highly accurate XSL patterns can be learnt from very few examples. XSL-Patterns must be combined with other NE-Agents if content *within* a tagged field must be further analysed. A TE-Agent may be required to finally resolve ambiguous XSL-Patterns.
- *Keyword matching*: Learning synonyms simply requires an aggregation over all examples. The resulting recogniser may be well suited to any type of document and types of information which are typically represented by unique names (at least within the scope of the application). Under these circumstances, fast and accurate NE-Agents can be easily produced and maintained. The main burden is in providing the exhaustive list of all synonyms. Augmenting the keyword matcher by allowing similarity-based matches may help in this respect by improving recall, though with a possible degradation of precision.
- *Text zone classification*: Following statistical classification techniques borrowed from Information Retrieval, text zones can be categorised/extracted based on weighted lists of keywords. The learning algorithm is here a statistical word occurrence analysis which requires relatively high numbers of training examples. Empirical studies have shown that in most cases the number of examples should be at least 30. Runtime performance speed can be fast. Accuracy is highly dependent on the content of the document—for websites with several regions with similar content, text zone classification based on weighted keywords may be imprecise. Moreover, this technique requires good document segmentation during pre-processing.
- *Token based rules*: Generally, the more training examples that are given to the inductive learning algorithm to produce token-based rules, the better the performance of the TBR both in terms of precision and recall. In practice, about 5 training examples are needed to reinforce a specific pattern and ensure it is learnt as a TBR. Hence, the success of the learning of TBRs is dependent on the nature of the document content; if the content has significant syntactic nature or very strong pre- and/or post-core strings, and there are few pattern variations, very accurate rules can be learnt rapidly. Otherwise, token-based rule-learning tends to generate TBRs that cover the more prevalent/strong patterns and ignores weaker patterns (which may best be covered by adding new rules manually). That is, automatically generated TBRs guarantee high precision, but sometimes with poor recall. TBRs are usually best used for formatted numeric information and other fairly localised patterns. TBRs may be used in parallel with less precise techniques, e.g. keyword matching, to complementary effect—i.e. the weaker approaches are used to improve recall when (and only when) precise TBR fail.
- *Spatial model*: The spatial model based TE-Agent is applicable to semi-structured and structured documents. Its primary role is to enhance the precision of the overall system by removing ambiguous and false matches produced by the NE-Agents—this allows an overall better performance for the IE System than trying to encode high precision into the NE Agents themselves [15]. However, training the spatial model is relatively intensive in terms of manual effort to produce sufficient training examples, therefore it should first be established that a given application will benefit from the TE-Agent.

7.3. Propose Methodology

The above observations leads to a proposed methodology for the construction of a knowledge extraction system, at least for the sub-class of problems that involve template filling from semi-structured documents such as internet pages:

1. A knowledge engineer analyses the page contents and designs a “template” to summarise the required extracted information for a given application
   - All possibly relevant NE-Agent types for each required template slot are pre-selected
2. Non-experts manually perform the template filling task on a number of example documents
   - An automatic learning technique is used to construct a TE-Profile
   - A set of primitive NE-Agents are trained based on the known values of template slots.
3. The initial knowledge extraction system is applied to a further set of test documents

- The performance with respect to each template slot/NE-Agent is evaluated
- Critically weak NE-Agents (i.e. those agents with poor performance not compensated for by other NE-Agents or the TE-Agent) are identified

4. The knowledge engineer refines the system
    - Manual extension or correction to rules, keyword lists etc. may be carried out.
    - Focussed retraining of specific NE-Agents may be required.

Two final comments on this methodology are worth making. Firstly, for many industrial content management scenarios it is well suited. A typical scenario in a so called "content factory" is that teams of manual labourers carry out a repetitive task of reading some input information source (e.g. a product catalogue) and re-enter the information into some form-like GUI which is the front end to the content management system. Thus Step 2 of the above methodology is in many cases already being carried out, but the potential benefit is not exploited, in terms of training IE Agents that can then semi-automate the overall information entry task .

A second comment is that the evaluation metrics of "precision" and "recall" do not always make sense in a commercial environment. Deeper cost-benefit analysis should be included within the methodology to determine how much real benefit (e.g. in terms of man power savings) the effort needed to, say, increase precision or recall of the overall system by 10% would actually bring. For example, if information extraction is being used to provide slightly more sophisticated search engine facilities for a web portal, then some imprecision is surely tolerable by the end user and the emphasis may be on higher recall rates so that all information remains accessible. Alternatively, an IE system that mines online financial news sites in order to produce summarised financial data relevant to an invest portfolio may have much more stringent performance measures. Management of this kind of trade-off may benefit from ROC analysis (e.g. see [30]).

8. Conclusion

The ability to extract desired pieces of information from semi-structured natural language texts is an important task with a growing number of potential commercial applications, particularly relating to content and knowledge management. Tasks involving the locating of specific data in web pages and other online documents (email, newsgroups, . . .) are particularly promising applications. As the amount of textual information available on-line grows, information extraction systems will become more and more important as a method of making this information available and manageable—indeed IE may be an instrumental technology in enabling the migration from todays *ad hoc* internet to tomorrows Semantic Web (www.semanticweb.org). Manually constructing such information extraction systems is a laborious task; however, learning methods have the potential to help automate the development process and make commercial applications cost effective.

The contribution of this paper has been both from a technical and from a system viewpoint. At a technical level, as well as introducing an overall architecture, a number of innovative components have been described with an emphasis on machine learning, including: a revised bottom-up algorithm to construct XSL-Patterns, a statistical word occurrence analysis to generate keywords for extracting parts of text, inductive to generate complex grammar rules and, finally, and, statistical analysis approach to build a spatial model for template filling. The techniques have been evaluated and shown to be effective with different document types and various web-based application scenarios.

From a system viewpoint, this paper has presented a complete information extraction system not only for research projects but also for practical applications. In this paper, the focus has been on extracting information from web pages. However, the complete system is designed both for normal documents and web pages. The system is designed as a hybrid architecture embodying multiple agents that can not only be applied in parallel but also in collaboration in order to provide optimal overall performance. While this approach adds complexity, in terms of runtime coordination of multiple agents and, from a methodology perspective, in terms of having to marry the appropriate technique(s) to each relevant type of information, the hybrid approach is viewed as a fundamental necessity. The diversity of types and formats of textual information precludes a "silver bullet" solution to information extraction—i.e. no one technique can address all problems. In particular, the system clearly distinguishes between NE-tasks (which find isolated occurrences of information items of interest) and TE-tasks (which compile complete content descriptions form the information items).

Moreover, it is supported by a general methodology that allows construction of the IE system to be carried out in a focussed and cost-effective way.

Note

1. To simplify the example, all attributes in nodes are not showed in example patterns. In the real implementation the attributes are used also to calculate overlapping pattern. Sometimes the attributes play an important role to distinguish the XML nodes.

References

1. ARPA, "Defense advanced research projects agency," in *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Morgan Kaufmann, California, 1995.
2. E. Brill, "Some advances in rule-based part of speech tagging," in *Proceedings of the 12th Annual Conference on Artificial Intelligence (AAAI-94)*, 1994, pp. 722–727.
3. M. Marcus, B. Santorini, and M. Marcinkiewicz, "Building a large annotated corpus of English: The PennTreebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
4. F. Ciravegna, "Adaptive information extraction from text by rule induction and generalisation," *Joint Conference on Artificial Intelligence (IJCAI01)*, Seattle, Washington, USA, 2001.
5. S. Soderland, "Learning information extraction rules for semi-structured and free text," *to appear in the Journal of Machine Learning*, 1998.
6. D.W. Embley, N. Fuhr, C.-P. Klas, and T. Roelleke, "Ontology suitability for uncertain extraction of information from multi-record web documents," *ADI'99 Proceeding*, 1999.
7. S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert, "Crystal: Inducing a conceptual dictionary," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995, pp. 1314–1319.
8. S. Huffman, "Learning information extraction patterns from examples," in *IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*, 1995, pp. 127–142.
9. H. Cunningham, K. Humphreys, Y. Wilks, and R. Gaizauskas, "Software infrastructure for natural language processing," in *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, 1997.
10. D. Fensel, B. Omelayenko, Y. Ding, M. Klien, A. Flett, E. Schulten, G. Botquin, M. Brown, and G. Dabiri, *Intelligent Information Integration in B2B Electronic Commerce*, Kluwer Academic Publishers, 2002, ISBN 1-4020-7190-6.
11. I. Muslea, S. Minton, and C. Knoblock, "A hierarchical approach to wrapper induction," in *Proceedings of the Third International Conference on Autonomous Agents (AA-99)*, 1999.
12. D. Freitag, "Information extraction from html: Application of a general learning approach," in *Proceedings of the 15th Conference on Artificial Intelligence (AAAI-98)*, 1998, pp. 517–523.
13. L. Liu, et al. "XWRAP: An XML-enabled wrapper construction system for web information sources," *International Conference on Data Engineering*, 2000.
14. H. Cunningham, "A definition and short history of language engineering," *Journal of Natural Language Engineering*, vol. 5, pp. 1–16, 1999.
15. L. Xiao, D. Wissmann, M. Brown, and S. Jablonski, "Where to position the precision in knowledge extraction from text," to be published, IEA/AIE 2001; *The 14th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*, Budapest, 2001 pp. 187–196.
16. L. Xiao, D. Wissmann, M. Brown, and S. Jablonski, "Information extraction from HTML: Combining XML and standard techniques for IE from the web," to be published, IEA/AIE 2001; *The 14th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert*, Budapest, 2001, pp. 164–174.
17. 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.
18. N. Kushmerick, D. Weld, and R. Doorenbos, "Wrapper induction for information extraction," in *Proceedings of the 15th International Conference on Artificial Intelligence (IJCAI-97)*, 1997, pp. 729–735.
19. S. Bennett, C. Aone, and C. Lovell, "Learning to tag multilingual texts through observation," in *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, 1997, pp. 109–116.
20. K. Seymore, A. McCallum, and R. Rosenfeld, "Learning hidden Markov model structure for information extraction," in *AAAI 99 Workshop on Machine Learning for Information Extraction*.
21. M. Califf and R. Mooney, "Relational learning of pattern-match rules for information extraction," *Working Papers of the ACL-97 Workshop in Natural Language Learning*, 1997, pp. 9–15.
22. E. Riloff, "Information extraction as a basic for high-precision text classification," *ACM Transaction on Information Systems*, vol. 12, no. 3, pp. 296–333, 1994.
23. D. Merkl, "Exploration of document collections with self-organizing maps—A novel approach to similarity representation," in *Proceedings of the European Symposium on Principles of Data Mining and Knowledge Discovery*, Trondheim, Norway, 1997, pp. 101–111.
24. A. Knoblock and Minton, *Accurately and Reliably Extracting Data from the Web: A Machine Learning Approach*, IEEE 1999.
25. N. Ide and J. Véronis, "Word sense disambiguation: The state of the art," *Computational Linguistics*, vol. 24, no. 1, 1998.
26. L. Xiao, D. Wissmann, M. Brown, and S. Jablonski, "Hierarchical concept description and learning for information extraction," in *6th Natural Language Processing Pacific Rim Symposium (NLPRS)*, Tokyo, Japan, 2001.
27. F. Crestani, "The troubles with using a logical model of IR on a large collection of documents," *TREC*, 1994.
28. J. Quinlan, *C4.5, Programs for Machine Learning*. Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann, 1993.
29. T.R. Leek, "Information extraction using hidden Markov models," Master's thesis, UC San Diego, 1997.
30. D. Provos and T. Fawcett, "Analysis and visualisation of classifier performance: Comparison under imprecise class and cost distributions," in *Third Intl. Conf. of Knowledge Discovery and Data Mining*, 1997, pp. 43–48.