TU

Technische Universität Wien

DIPLOMARBEIT

Information Extraction – Utilizing Table Patterns

ausgeführt am Institut für
Softwaretechnik und Interaktive Systeme

der Technischen Universität Wien

unter der Anleitung von
Univ.-Prof. Dr. Silvia Miksch und Mag. Kathi Kaiser

durch

Burcu Yıldız
Matrikelnummer 9926103
Informatik (E881)
Weyringergasse 32/11, 1040 Wien

Wien, im August 2004

Table of Contents

List of Figures

Acknowledgements

I would like to take this chance to thank all those who have supported me during this work, directly and/or indirectly.

I would first like to express my sincere thanks to my advisor Univ.-Prof. Dr. Silvia Miksch for guiding me through my work, giving me constructive critics, and for encouraging me to write this master thesis in English.

Second, I would like to thank Kathi Kaiser for giving me ideas to make my application better and for correcting my written work.

Third, I would like to thank my family for supporting me through my life and helping me to finish my study and this thesis.

Last, but definitely not least, I would like to thank God, for guiding and helping me through almost 23 years on earth. Without His help I would not have reached where I have.

Introduction

With the wide spread use of computer and internet technologies we face a huge amount of information in digital and mostly unstructured form. The problem to make use of "relevant" information for a specified purpose from such documents is getting harder and harder. In this context there arise research areas about retrieval, filtering, extraction, and integration of information which comprise the field of Information Processing.

Approximately 80% of the information in our world is textual. To find a way to store this information in a widely usable form became more important every day. There exist many document structures and formats. One of them, the portable document format (PDF), is commonly used in almost all areas. The main benefit of this format is that it enables users to easily and reliably exchange and view electronic documents independent of the environment in which they were created. There is another development with the same aim - to create completely portable data - named XML. It stands for E**X**tensible **M**arkup **L**anguage and marks up a document so that a reader can identify each piece of the document, whereby a reader can be a human being or a program. That makes it easier to handle the stored data, for example to extract relevant information from the file, or to do further processing on it.

The task of this master thesis is to develop a tool for extracting table information from PDF files and store this information in a structured XML file. Thus, the initial state at the beginning of my work, my approach and it limitations are explained.

Outline

Chapter 1: Information Processing – contains a detailed introduction to the fields related to Information Processing. These fields are Information Retrieval, Information Filtering, Information Extraction, Information Integration, and Wrappers.

Chapter 2: Information Processing with PDF Files – looks at the main architecture and the strengths and weaknesses of the PDF file format. Further, the question of how well is PDF suitable for Information Processing in general and for Information Extraction in special is explored.

Chapter 3: Information Processing with XML Documents – contains the historical background, the architecture, and the application areas of XML. Further, some complementary specifications to XML that could be of interest for Information Processing purposes are listed. And last, the question of how well is XML suitable for Information Processing is explored.

Chapter 4: Comparison of Existing PDF Extraction Tools – compares some tools that extract information from PDF files and stores it in either HTML files or XML documents.

Chapter 5: Task Description and Implementation - contains an explanation of the task of this thesis. First, the task of table extraction is being introduced. Then, the implementation itself and the graphical user interface are being presented. Further, the limitations of the approach and future work that could be done are listed.

CHAPTER 1

Information Processing

Today, we are overwhelmed with data from different sources and of various formats as we were never before. Although not each data contains relevant information, the need of turning data into information is of enormous importance for almost anyone. This is the driving force for research in fields concerned with the acquisition, management and processing of data/information. The field of *Information Processing* is the theme of this chapter.

Like many other terms in the information theory, there is no exact definition of the term of *Information Processing* (IP). Sometimes the synonym *Data Processing* is used instead. A definition of IP on the website of the "Free Dictionary"[1], describes IP as: "the sciences concerned with gathering and manipulating and storing and retrieving and classifying recorded information".

I must note that thereby, information can mean anything, not only digital data in computer systems. I will only analyse the processing of digital data and more restricted, the processing of text documents.

I will take a look on some of the sciences that build a part of IP. Their common aims are to assist the user by her search for information. The first step of such a process is to search for documents that are relevant to the user's request. Thus, the first fields I will take a look on are the fields of *Information Retrieval* and *Information Filtering*. Information Retrieval is concerned with the selection of relevant documents in a document collection, whereas Information Filtering is concerned with the selection of relevant documents in an incoming stream of documents. Once a user gets the potentially relevant documents, she must search for the parts of these documents, which are relevant for her specific purpose, or she must integrate these documents into a given environment. That leads to the fields of *Information Extraction* and *Information Integration*. Information Extraction is concerned with the location and extraction of relevant data in a document, whereas Information Integration is concerned with the integration of the extracted data or the document in a different information resource. Then, for the sake of completeness, I will take a look at the field of *wrappers*, a restricted field of Information Extraction, which is concerned with the extraction of information from web documents.

[1] http://www.thefreedictionary.com

1.1 Information Retrieval

Information Retrieval (IR) is a field concerned with the structure, analysis, organisation, storage, searching, and retrieval of information. In this context, information can mean anything, for example, text files, sound files, images, and so on. In the current time, many *Information Retrieval Systems* (IRS) exist. The most of them handle only textual information, where the most common types of textual data are technical reports, newspaper articles, books, etc. The task of such an IRS is to search through a large set of documents and to return the documents that are predicted as potentially relevant to a user's information need. [Salton & McGill, 1983]

In the last few years, the importance of image or video retrieval also increases. Such IRSs are referred to as *Visual Information Retrieval Systems*. Del Bimbo [1998] describes the purpose of this field as: "to retrieve, from a database, images or image sequences that are relevant to a query. It is an extension of traditional information retrieval so to include visual media." Although, this field becomes more important, there are only a few commercial products to date. Some of them allow the user to search with a simple image for similar ones, and others allow the user to build complex visual queries, supporting the user in every query-building step.

Anyway, I will confine my explanation to the retrieval of text documents only. This restricted field is also known as *Document Retrieval* (DR) or *Text Retrieval* (TR), but I will use the more general term.

1.1.1 History

In the past, card catalogues in libraries were the most established IR "systems". In these catalogues, a user could find the books about a specific theme searching for related key words registered in the cards. The goal of these "systems" was to enable people a fast access to the available data. You will surely agree that today the available data is more then ever, especially data in digital form. Thus, effective and efficient retrieval systems are needed to assure that relevant data not get lost.

The Defense Advanced Research Projects Agency (DARPA) was interested in this field and originates the TIPSTER text program. The goal of this program was to advance the state of the art in text processing technologies and was formally ended in 1998. The Text Retrieval Conference (TREC) was started in 1992, as part of the TIPSTER text program and was co-sponsored by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defense (DoD). On the website of the TREC[2], the goals of this workshop series are listed as follows:

- To encourage research in information retrieval based on large test collections;
- To increase communication among industry, academia, and government by creating an open forum for the exchange of research ideas;
- To speed the transfer of technology from research laboratories into commercial products by demonstrating substantial improvements in retrieval methodologies on real-world problems;
- To increase the availability of appropriate evaluation techniques for use by industry and academia, including development of new evaluation techniques more applicable to current systems.

There were twelve conferences to date, and the next will be held in 2004. The first TREC should be seen as a first step in the development of IRSs using a large test collection. Because

[2] http://trec.nist.gov/

the test collection that was made available by the TRECs was about 100 times larger than any other test collection used by the developers in the past, the participants had to rebuild their systems to be able to handle such a vast amount of test documents. TREC-3 provided the first platform for more complex experimentation. [Voorhees, 2003]

The test collection used in the TRECs is large enough to be seen as an operational environment. Test collections consist of a set of documents, a set of information needs called *topics*, and of relevance judgements (i.e., an indication whether a document is relevant to a topic or not). The relevance judgements used in TRECs were binary – either a document is relevant or it is not. Relevance judgements can be made in three ways [Harman, 1993]:

1. Relevance judgements could have been made on all documents in the test collection. This way is clearly impossible, because of the large amount of documents in the collection. It would take hours, when not days, to complete the judgements.
2. Relevance judgements could be done on a random sample of documents.
3. Relevance judgements could have been made on a sample of documents selected by the participating systems. This way is called *pooling*.

Harman lists the steps to construct the 'pool' of documents as follows [Harman, 1993, p.39]:

1. Divide each set of results into results for a given topic.
2. For each topic within a set of rules, select the top 200 ranked documents for input to the pool
3. For each topic, merge results from all systems
4. For each topic, sort results based on document numbers
5. For each topic, remove duplicate documents

All of the retrieval conferences had two main tasks, the *ad hoc task* and the *routing task*.

- The *ad hoc* task can be explained as the task where a static set of documents is searched with new queries. This task is comparable to a search process of a user in a library environment. The books represent the static set of documents and it is known, but the search queries of the user are not known.
- The *routing task* can be seen as a filtering process where only the relevant documents pass through. Here, it is assumed that the questions (filter profile) are always the same, but the incoming documents are changed. The system must decide for each incoming document whether it is relevant or not. [Harman, 1993]

The TRECs consist of subtasks called *tracks*. This tracks focus on areas in which particular retrieval tasks are defined. On the website of TREC[3] the purposes that tracks serve are described as follows:

> First, tracks act as incubators for new research areas: the first running of a track often defines what the problem *really* is, and a track creates the necessary infrastructure (test collections, evaluation methodology, etc.) to support research on its task. The tracks also demonstrate the robustness of core retrieval technology in that the same techniques are frequently appropriate for a variety of tasks. Finally, the tracks make TREC attractive to a broader community by providing tasks that match the research interests of more groups.

These tracks are listed and described on the website of TREC[4] as follows:

[3] http://trec.nist.gov

1. **Cross-Language Track**

   A track that investigates the ability of retrieval systems to find documents that pertain to a topic regardless of the language in which the document is written. While the last cross-language track in TREC was run in TREC 2002, cross-language retrieval tasks are studied in both Cross-Language Evaluation Forum (CLEF), and the NTCIR workshops.

2. **Filtering Track**

   A task in which the user's information need is stable (and some relevant documents are known) but there is a stream of new documents. For each document, the system must make a binary decision as to whether the document should be retrieved (as opposed to forming a ranked list).

   The filtering track was last run in TREC 2002.

3. **Genomics Track**

   A track introduced in TREC 2003 that will run again in TREC 2004. The purpose of the track is to study retrieval tasks in a specific domain, where the domain of interest is genomics data (broadly construed to include not just gene sequences but also supporting documentation such as research papers, lab reports, etc.).

4. **HARD Track**

   Another track introduced in TREC 2003 that will also run in TREC 2004. The goal of HARD is to achieve High Accuracy Retrieval from Documents by leveraging additional information about the searcher and/or the search context, through techniques such as passage retrieval and using very targeted interaction with the searcher.

5. **Interactive Track**

   A track studying user interaction with text retrieval systems. Participating groups develop a consensus experimental protocol and carry out studies with real users using a common collection and set of user queries. The interactive track was run as an adjunct to the Web Track in TREC 2003, and is not slated to run in TREC 2004.

6. **Novelty Track**

   A track to investigate systems' abilities to locate new (i.e., non-redundant) information. This track will run in TREC 2004.

7. **Question Answering Track**

   A track designed to take a step closer to information retrieval rather than document retrieval. The QA track will run in 2004, with a focus on definition, list, and factoid questions.

8. **Robust Retrieval Track**

   A track that includes a traditional ad hoc retrieval task, but with the focus on individual topic effectiveness rather than average effectiveness. The robust retrieval track first ran in TREC 2003 and will run again in TREC 2004.

9. **Terabyte Track**

   A new track for TREC 2004. The purpose of the terabyte track is to investigate whether/how the IR community can scale traditional IR test-collection-based evaluation to significantly larger document collections than those currently used in TREC. The retrieval task will be an ad hoc task using a static collection of approximately 1 terabyte of spidered web pages (probably from the .GOV domain).

10. **Video Track**

---

[4] http://trec.nist.gov/tracks.html

TREC 2001 and 2002 contained a video track devoted to research in automatic segmentation, indexing, and content-based retrieval of digital video. Beginning in 2003, the track became an independent evaluation (TRECVID).

11. **Web Track**

    A track featuring search tasks on a document set that is a snapshot of the World Wide Web.

1.1.2 Steps of the IR-Process

Rijsbergen wrote [Rijsbergen, 1979, p.3]:

> In principle, information storage and retrieval is simple. Suppose there is a store of documents and a person (user of the store) formulates a question (request or query) to which the answer is a set of documents satisfying the information need expressed by this question. He can obtain the set by reading all the documents in the store, retaining the relevant documents and discarding all the others. In a sense, this constitutes 'perfect' retrieval. This solution is obviously impracticable. A user either does not have the time or does not wish to spend the time reading the entire document collection, apart from the fact that it may be physically impossible for him to do so.

As Rijsbergen [1979] mentioned in his book, a manual retrieval process is impracticable. In order to simplify the retrieval process it is split in two main steps: indexing and searching

Indexing is the process of assigning descriptive items, named *key words*, to documents. The choice of good key words is an essential problem that significantly affects the efficiency of the retrieval system. I will give an overview of different indexing methods and will point at their advantages and disadvantages.

Searching is the process where the potentially relevant documents are being located. Rijsbergen [1979] gives a good overview over all search strategies and additionally points at ways to implement these strategies. I will only give a brief introduction on different search strategies and will list their advantages and disadvantages. For more detailed information I refer to the book [Rijsbergen, 1979].

1.1.2.1 Indexing

Indexing is the process of assigning descriptive key words to documents. This process is difficult and can be done either by a human or automatically.

If the indexing process is done by a human indexer, the indexer goes through all the documents and assigns each document a set of key words. It is clear that the performance of the process is highly based on the skills of the indexer, and that the process is time consuming. These are the main disadvantages of the manual indexing process. The main advantage of the manual indexing process is that if the human indexer is familiar with the domain, he can find key words that represent the document quite good. [Nohr, 2003]

To avoid the disadvantages of the manual indexing, ways to automate this process are being searched. By the automatic indexing process, the system splits the text into strings delimited by white spaces. There are three methods to automatically select key words [Mresse, 1984]:

1. **Selection of key words from a thesaurus**: To avoid inconsistency and to reduce the variability of the vocabulary, it's became common that the key words are selected from an index vocabulary, called a thesaurus. Such a controlled vocabulary must, first of all, tackle **synonyms** (i.e. words with similar or identical meanings). If a user searches for documents about cars then it is desired that the system also retrieves documents indexed with the key word "automobile". The thesaurus must further assign **acronyms** to the terms pointed by these acronyms. This would allow the user to

use acronyms in her query, like “MIPS” (*million instructions per second*), instead of writing the whole notion. **Antonyms** must be also tackled, because such words became synonyms if one of them would be negated. For example “slow” means in some way “NOT fast”, thus they had to be handled as synonyms. Another class of terms, which are more difficult to handle, is the class of **homonyms** (i.e. words with the same form but different meaning). Such terms must be viewed in the context in which they appear. For example the term “virus” can mean different things in different contexts, e.g., “computer virus” or “flu virus”, etc. [Pollitt, 1989]

2. **Selection with stop-word lists:** All the words in a document that also exist in a stop-word list (so called anti-thesaurus) are removed and the rest are selected as key words. This method has two advantages. First, all non-significant words are removed from the document. Second, because many words are removed, the file size can be reduced by between 30 and 50 per cent. [Rijsbergen, 1979]
3. **Weighted Selection:** By this selection method, the descriptors are handled with respect to its context in the environment. The environment can be the document, the query or the whole document collection. The weighting of key words, have the aim to attach the key words importance.

There is another way of indexing, which combines the human skill and the computer aid, called *computer-based indexing*. By means of this indexing process, the system searches for key words and asks the human indexer whether a word should be a key word or not.

1.1.2.2 Searching

Searching represents the actual step in which the documents are being classified as relevant and being returned. I shall give you an overview of the mostly used search strategies in IR.

Boolean search: The earliest retrieval systems required requests in form of Boolean expressions. The user has to build a sequence of index terms (key words) combined by the logical connectives (AND, OR, and NOT). The system then uses this expression for retrieving the relevant documents for which the query expression is ‘true’. Some disadvantages of this model are stated here:

- Boolean expressions have either the value true or false. For that reason it is not possible to order the returned documents according to its relevance because a document is either relevant or not, no other states are between true and false. This property leads also to the result that documents with a high probability of relevance are disregarded too. For example, if a user searches for the key words “information”, “retrieval”, “filtering” connected with an AND operator and a document only consists the first two key words, the document would be disregarded because the hole expression has the value ‘false’.
- Furthermore, to build Boolean expressions is not always easy. Many queries need highly complex expressions, which “normal” users cannot build.
- This model does not allow the use of weighted indexing and the weighting of the search terms.
- The amount of the return set is not predictable.

Fuzzy Search: This model tries to avoid some of the disadvantages of the Boolean model. The user can weight the search terms in the expression and the model can make use of weighted indexing, but the difficulty of building complex expressions holds. Because now states between 0 and 1 can exist, an ordering according to its relevance is possible.

Vector-space Search: In this model documents and queries are represented by vectors in an n-dimensional vector-space and each dimension corresponds to an index term. If a user sends a query, the vectors similar to that query-vector are searched and the results are ordered using a metric. To build a query in that model is easier as the two models I mentioned above. Another advantage is that the usage of that model for other collections is relatively easy. [Ingwersen, 1992]

Clustering Model: This model tries to make use of the similarity of the documents in the collection. The idea is simple: when a document is relevant to a query, other documents similar to this one can also be relevant with a high probability. For that reason, all documents in the collection are checked for their similarity to one another. Clusters are built of similar documents and these documents are saved physically near to one another, which improves the access speed. For each cluster a centroid element with minimal distance to each document in the cluster is created. All centroids are saved together. If a request comes from the user, the centroid with the highest similarity is selected and the documents in the belonging cluster are listed.

The main advantage is that the similarity of the documents is important. This leads among other things to faster systems because less *input-output* costs occur.

1.1.3 Evaluation

There is a common agreement between the people that are concerned with IRSs, that the systems have to be analysed in terms of performance, but there is still not an agreement about which quantities of an IRS has to be involved in the evaluation process and how the evaluation functions must look like, exactly. [Rijsbergen, 1979]

This disagreement is, in my opinion, a result of the fact that the evaluation process can be affected by a large number of variables and that the researchers attach the importance to different variables. Some of these variables are for example the type and coverage of the document collection (i.e. the extent of relevant documents in the document collection), the used indexing tools, the analysis and search methods incorporated into the system, the input-output equipment used, time lag and costs needed to produce answers, the operating efficiency, and so on. [Salton & McGill, 1983]

To put the evaluation process in a more standardized platform the Text Retrieval Conference (TREC) was established. There have been twelve conferences to date. These conferences try to encourage the research in IR based on large test collections, because only large test collections are able to model operational settings realistically.

All the IRSs, participated in these conferences, are evaluated, among others, with the standard two measures: *precision* and *recall*. **Precision** is the number of relevant documents retrieved to the total number of documents retrieved.

$$Precision = |\text{ relevant documents retrieved }| / |\text{ documents retrieved }|$$

Recall is the number of relevant documents retrieved to the total number of relevant documents in the collection, both retrieved and not retrieved.

$$Recall = |\text{ relevant documents retrieved }| / |\text{ relevant documents in the collection }|$$

There are other measures too, but I will only mention one more, namely the *fallout* measure. **Fallout** is the proportion of non-relevant documents retrieved. [Rijsbergen, 1979] Let L be the number of retrieved documents, R the number of relevant retrieved documents, N the number of documents in the collection, and C the number of relevant documents in the collection. Then, the formula of the fallout measure is as follows:

$$Fallout = (L - R) / (N - C)$$

The first measure, precision, requires no other knowledge about the other documents in the collection that are not retrieved, because a user can simply classify the retrieved documents as relevant or non-relevant and compute the measure value. But the other two measures recall and fallout require knowledge about all the documents in the collection, because these measures require the number of relevant respectively non-relevant documents in the entire collection. One way can be, to classify all the documents in the collection after every query, in relevant and non-relevant documents. This way is understandably time consuming and therefore only applicable on small document collections and impossible in operational systems. [Mresse, 1984]

In the next chapter, I will give a brief introduction in the field of Information Filtering, which is also a way to get potentially relevant documents in an Information Processing process.

1.2 Information Filtering

Belkin and Croft describe the task of **Information Filtering** as the task of removing irrelevant information from an incoming stream of unstructured textual information according to user preferences. [Belkin & Croft, 1992]

The filtering process can be either on the server side or on the client side. If the filtering is on the server side, the server predicts whether a document is relevant for the user or not, and sends the user only the relevant documents. An example for such server-side filtering systems can be news filters, which send to its users just the news the users are interested in. If the filtering is on the client side, the filtering system on the client side decides whether a document will be removed from the input stream or passed through to the user. An example for such client-side filtering systems can be simple e-mail filters, which may route spam-mails to the trash folder or group e-mails in different folders regarding to its contents.

In both filtering systems, the user's needs are in the centre of attention. Describing the user needs can be accomplished in two ways:

First, the user describes her needs by herself, easily by specifying key words that describe the field of interest. While it seems to be very easy to specify key words, this approach brings some disadvantages with itself. The user may choose key words that do not describe her interests in the best way; the selected key words have more than one meaning; or the selected key words describe also some irrelevant fields.

Second, the filtering system can build a user profile regarding to the user's previous activities. This system improves the ability known as *adaptive filtering*, which means that the filtering system is able to adapt itself if the user's interest has changed. Some work on making filtering systems adaptive is done, for example, by Klinkenberg and Renz [1998].

There are three ways to adaptive information filtering:

1. **Cognitive (Content-based) filtering**: As the name implies, this filtering technique makes use of the content of a document and compares it with the user profile to decide whether to remove it or to save it. The feedback of the user about whether a decision was right or wrong improves the efficiency of this filtering technique. The main advantage of this technique is that the system can discover new interesting information if it is similar to previously encountered documents. This technique has the problem of "serendipity", which means that the system removes interesting documents if they did not previously encounter. [Olsson, 1998]

2. **Social filtering:** This filtering technique makes use of other user's profiles that are similar to a specific user profile. For example, imagine that I will be informed about new books in the category of science fiction. The filtering system can then decide that I will probably also be interested in crime novels, if other users with the same interest read also crime novels. Because this technique does not rely on the content of documents, systems using this technique can discover new documents of a sort previously not encountered. The main disadvantage of this technique is that it requires a significant amount of users in the system before it can make decisions based on their opinions. Another disadvantage is that it can happen that I am really interested in crime novels too, but because there is no user who reads science fiction books and crime novels I get no information about crime novels. [Olsson, 1998] It might require quite a few users in the system before it starts to work well, but when there is a sufficient amount of users, new users can benefit from the previous users' ratings. This means that the problem of cold-start is reduced when the system does not have to start from scratch every time a new user begins to use it.
3. **Hybrid filtering**: This technique tries to combine the two filters described above to take advantage of the pros and to avoid the contras.

Now you can think that IR and Information Filtering just do the same thing. In some issues you may be right, but Belkin and Croft [1992] list three main differences among them:

1. The user profiles in Information Filtering systems represent long-term interests, while in IR the queries represent short-term interest.
2. In Information Filtering systems the input is a stream of documents, while in IR systems the input is a collection of documents.
3. The filtering process tries to decide, which documents have to be removed, while the retrieval process tries to find relevant documents.

1.3 Information Extraction

"Information Extraction (IE) is a form of natural language processing in which certain types of information must be recognized and extracted from text." [Riloff, 1999]

An **Information Extraction System (IES)** analyse the input text in order to extract relevant portions. IESs do not attempt to understand the meaning of the text, but they do only analyse portions of input text that contains relevant information. [Bagga, 1998]

In the past, this work of extracting information of interest from text was done by hand. A person has read all documents and created a database with the information. This task was very time consuming and difficult. Furthermore, the evaluation of the results was difficult, too. Thus, the interest of a computer system for this task was increasing. This interest leads in the development of IESs, whose historical background is explained in the next subsection.

1.3.1 History

In the late 1980's the U.S. government funded different research groups to work on message understanding systems. A number of these different groups decided to come together to compare their message understanding projects and to understand the approaches of the other groups better. This leads to the first of an ongoing series of message understanding conferences. To get a basis of comparison these groups decided to work on a set of common messages.

There have been seven Message Understanding Conferences (MUC). The term "message understanding" disappeared and the more accurate term of "Information Extraction" has taken its place.

A MUC builds a platform to evaluate the performance of different IE projects developed by different sites, both from academic and industrial research areas. You can detect a significant improvement of the IE systems over the years at the MUCs. This is one of the most interesting aspects of these conferences.

Participants of a MUC evaluation work on a given description of the current scenario with a set of documents and a training corpus (i.e., templates to be extracted from these documents). Then they get some time (1 to 6 months) to adopt their system to the new scenario. After this time, the participants get a new set of documents (test corpus) to test their systems with. After this testing procedure the participants return their extracted templates to the conference organizer, who manually builds the answer key (i.e. set of templates for the test corpus). [Grishman, 1997]

I will give you an overview of these conferences to show how the field of IE grew over the years. [Gaizauskas & Wilks, 1998]

MUC-1 (1987): This conference builds just a platform to compare the participated systems. The texts were naval operation reports. There was no task definition and there were no evaluation criteria.

MUC-2 (1989): The task at this conference was to fill a template, which had ten slots. The domain was again tactical naval operation reports. Inadequate evaluation criteria were defined and the scoring was done by the participating sites.

MUC-3 (1991): The domain consists of articles reporting terrorist activities in Latin America. The defined template was more complex as the template used at MUC-2; it consisted of 18 slots. Formal evaluation criteria were introduced. A semi-automated scoring program was developed and made available for use by participants during development. Official scoring was done by the organizers.
The two earlier conferences were initiated, designed, and carried out by Beth Sundheim under the auspices of the Navy and focused on extraction from military messages. Since the third conference, the conferences had been carried out under the auspices of the TIPSTER Text-Program and focused on extraction from newswire articles. [Chinchor, 1998]

MUC-4 (1992): The domain remained essentially unchanged. Just the template complexity increased to 24 slots.

MUC-5 (1993): The systems were tested in two different domains: financial newswire stories about joint ventures among two or more entities (companies, governments, and/or people) and product announcements in the electronic circuit fabrication area in two languages: English and Japanese. For the first time, nested templates were used. Scoring was modified to include new evaluation metrics and the scoring program enhanced.

Over the course of the five MUCs, only the tasks and templates had become increasingly complex. A meeting in December 1993, following the fifth MUC, defined a set of objectives for the forthcoming MUCs: to push IE systems towards greater portability to new domains, and to encourage more basic work on natural language analysis by providing evaluations of some basic language analysis technologies.

Some sites have needed more than six months to adapt their systems for the fifth MUC, whereas the most effort was invested in the adaptation of the system into the new domain. This leaded to the question, whether this effort was justifiable or not.

MUC-6 (1995): The domain consists of articles regarding changes in corporate executive management personnel. A set of four evaluation tasks was specified, which I will explain later on:

1. Named entity recognition
2. Co-reference
3. Template elements
4. Scenario templates (traditional IE)

MUC-7 (1998): One more task was added to the previous tasks: the template relation task. For the first time, the multilingual named entity evaluation was applied using training and test articles from comparable domains for all languages. The domain for all languages for training was airline crashes and for testing was launch events.

Over the years, several tasks introduced at the MUCs. I will give you an overview of these tasks, referring to the work of Marsh and Perzanowski [1998].

The **named entity** task (NE) represents the lowest level of IE tasks, defined in the MUC. It comprises of identifying and categorization of proper names appearing in a text. The entities defined in the MUC guidelines were entities (organization, person, location), times (date, time) and numbers (money, percent)

The problem of *syntactical* or *semantic variability* in texts is a challenge of the NE task. The different formats in which a date can appear in the text (e.g., 12/02/2001, 12.02.2001, 12-02-2001) is an example for the *syntactical variability*. The usage of identical words that refer to different things like person names and company names is an example for the *semantic variability*. Another source of problem can be the appearance of abbreviations in the text (e.g., “J.K. Rowling”, “Joanne K. Rowling”, “Joanne Kathleen Rowling” stand for the same person).

Another task was introduced, which is the same as the NE task, but for Chinese and Japanese, called the **multi-lingual entity task** (MET).

The **template element task** (TE) is about extracting basic information related to organization, person, and artefact entities, drawing evidence from anywhere in the text. It separates domain-independent from domain-dependent aspects of extraction. Answer keys contain objects for all organizations, persons, and vehicle artefacts mentioned in the texts, whether relevant to scenario or not.

The task of extracting relational information among entities is called the **template relation task** (TR). It is first introduced for the seventh MUC.

The top-level IE task is the **scenario template task** (ST). It is required to extract pre-specified event information and it related the event information to particular organization, person, or artefact entities involved in the event.

The **co-reference task** (CO) captures information on co-referring expressions (i.e., all mentions of a given entity). In a text, the same entity can appear in several ways. For example, a text about a person, let’s call him Prof. X, can consist several references such “he is …”, “Mr. X was …”, “the Professor has …”, etc. The co-reference task claims that the system can capture all these references using content und context of all these appearances.

1.3.2 Evaluation

The evaluation of an IES is a non-trivial issue, because of the fact that the performance of an IES depends on different factors. Some of those factors that may affect the performance of an IES are the complexity of the extraction task, the quality of the knowledge bases available to the system, the syntactic and semantic complexity of the documents to be processed, the regularity of the language in the documents, and so forth. The MUC's scoring program represents an important first step by showing that it is possible to rigorously evaluate some aspects of an IES. [Cardie, 1997]

From the third MUC on, organizers developed an official scoring program to score the results of the participated IE systems. The most used evaluation measures are the following ones:

Recall = | correct slot-fillers in output template | / | slot-fillers in answer key |

This value is a measure of completeness.

Precision = | correct slot-fillers in output template | / | slot-fillers in output template |

This value is a measure of correctness.

F-score = 2 x Precision x Recall / (Precision + Recall).

The F-score is a function that combines recall and precision values in a single measure.

1.3.3 Architecture of an IE System

Cardie [1997] points out that the architecture of IESs varies from system to system, but they have some main functions in common.

The first phase of an IE process is the *tokenization* and *tagging* phase. In this phase the text is first divided into sentences and words. This can be trivial for languages such as English, where the words are separated by white spaces. But for many languages this is not so easy, for example for Chinese or Japanese, where words are not delimited with white spaces. For such languages an additional segmentation module has to be included in the system.

Once, the text is divided into sentences and words the sentence analysis phase begins. In this phase, one or more stages of parsing are applied that together identify noun groups, verb groups, prepositional phrases, and other simple constructs. IE systems generally use *partial parsing* instead of *full-parsing,* where for each sentence a whole parse tree is created. The full-parsing method was tested also in some earlier IES, but it come out that this method brings not a significant improvement of performance. It is a Natural Languages Processing (NLP) method, more exactly using in the meaning understanding (text understanding) task, whose aim was to "understand" what a text was talking about. IE is just about extracting pieces of relevant information. Generally, text documents contain just a few sentences of interest. If the IE system would use a full-parser, all the other sentences that can be ignored in fact, would be parsed for nothing. Thus, there is no need for the full-parsing method.

After the sentence analysis phase, comes the first entirely domain-specific component of an IES, called the *extraction* phase. During this phase, the system identifies relations among entities of interest. The *merging* phase tries then to combine entities that refer to each other. For that purpose, all entities, which were identified in the extraction phase, were checked whether an entity is new or refers to an existing entity.

The last phase in an IES is called the *template generation* phase. In this phase, the system determines the total number of events in the text and maps the pieces of information onto each event to generate finally the output templates.

1.3.4 Approaches

There are two main approaches by building IE systems: a rule-based approach and an active learning approach. Both have significant advantages and disadvantages. You must choose between them recording to your own requirements.

1.3.4.1 The Rule-Based Approach

This approach requires expertise about the IE system itself and about how to build rules to extract the desired information for a specific domain. Thus, one or more knowledge engineers are needed to build the grammar and the rules for the IE system, mostly by hand. Therefore, the performance of the entire system depends on the skill of the knowledge engineers.

The rule-building process is an iterative process, where first some initial rules are created. The system first runs with this initial corpus of rules on some training data. According to the results a decision is made where the rules are under-generated or over-generated. Afterwards, the knowledge engineers make modifications to the rules and iterate the process.

The main advantage of this approach is that the performance of such handmade systems is very good. The main disadvantage of this approach is that it is difficult to adapt the system to new requirements in the specification. For example, if a system is built for only upper or lower case texts and then it is decided that the system must also handle multi-case texts all the rules must be rewritten. But this disadvantage takes not always place. For example, if the system was originally build to extract city names and then it is decided that additionally mountain or river names must be extracted, the adaptation of the system is relatively easy, because this change requires only some more rules that can be written without much effort. But no one can guarantee that a specification is not changed in the future or that it is not changed in a manner that the adaptation process takes much time. [Kauchak et al, 2002]

1.3.4.2 The Active Learning IE-Approach

This approach differs from the rule-based approach in the sense of the expertise that is needed to build the system. No knowledge engineers are required, but someone who knows about the specific domain. This person has to only annotate the parts of a text that are relevant to build a corpus of training data with annotated texts. The learning algorithm is applied on this corpus, which automatically builds rules for a specific domain. The process of building rules can also be more interactive. The user can be involved in the process to decide whether the hypothesis of the system for a text is right or wrong. Thus, the learning algorithm can change its rules by using the new information. [Kushmerick & Thomas, 2002]

The main advantage of this approach is that no expertise is required to build the IES. This would allow people without knowledge about the process of building an IES or about the process of creating rules to adopt an IES for their specific purpose.

Another significant advantage is the easy adaptation to new domains. If we take a look at the example mentioned in 1.3.4.1 we see that this approach acts different depending on the situation. The active learning approach acts similar. In the example about multi-case texts, an IES based on the active learning approach can easily adapt its rules just by applying the learning algorithm again. But for the example about the change in the specification to extract additionally mountain or river names the adaptation is not so easy. Someone must go through all the training data and annotate the additional parts of interest in the texts.

The main disadvantage of this approach is that training data may not exist or cannot be found. Such a case would wipe out the whole benefit. It can also be possible that the domain is too complex and is not so easy to find and annotate the parts of relevance. Another disadvantage

comes from the example mentioned above, namely by some changes in the specification all the training data must be annotated again.

Thus, the performance of this approach highly depends on the training data and if the training data cannot fit some conditions, the whole approach makes no sense anymore.

1.3.4.3 Comparison of the Approaches

Besides the description of the advantages and disadvantages of the approaches we can say that the choice of the right approach depends on the requirements and on the sources that are available.

If a high performance system is needed and there does not exists a lot of training data, or the domain is too difficult, the right choice would be the rule-based approach, because a knowledge engineer can create rules for patterns, even if these are not available in the training corpus. Furthermore, the knowledge engineer can write rules just in the right generality for the specific domain and requirements.

On the other hand, if a system is needed for a task and enough training data is available and the domain is not too complex, for example for a named entity task, the right choice would be the active learning approach, because it saves much time in the rule building process.

1.3.5 Challenges

The IE development has to face several challenges that build barriers to make it a practical technology. Some of them are described in this subsection.

1.3.5.1 Higher Precision and Recall

The precision and recall value of the evaluated IE systems, for example, at the seventh MUC, differs according to the specific tasks. The according F-values (measure with recall and precision weighted equally) are observed in the following table.

Tasks	Named Entity [NE]	Co-reference [CO]	Template Element [TE]	Template Relation [TR]	Scenario Template [ST]
MUC-7	F < 94%	F < 62%	F < 87%	F < 76%	F < 51%

Figure 1: Fallout values for the several tasks at the MUC-7

It is clear that the question of the acceptability of these values differs also according to the specific tasks. Cowie and Lehnert [1996] suggest that 90% precision will be necessary for IE systems to satisfy information analysts. Thus, just one task, namely the NE task, achieved this value. There is much work to do to reach this value for all of the other tasks.

Improvements in precision and recall are high priority challenges for IE systems. No one can expect a wonder that takes the development many steps forward, but there exists a hope also to getting better precision and recall values step by step.

1.3.5.2 Portability

IE systems in general were developed for a specific scenario and for a specific language. To adapt such systems to a new scenario requires mostly months of effort, as we see at the developed/adapted systems for the fifth MUC. This fact builds one of the major barriers to make IE systems a practical technology. Systems and tools are needed that can be adapted by the user for new scenarios in days or weeks, not months.

Adaptation can be required in the following terms: [Ciravegna, 2001]

1. **Adapting an IES to a new domain**

   It is clear that IESs can only become a common-use technology if they provide the extraction of text from different domains. No one would spend his money to get a system that works only for a single domain, thus in many cases for a certain time, because the interest in a domain can disappear. Extracting information about a new domain requires new rules, and so on. This problem affects systems with different approaches in different ways. Adapting rule based systems, for example, is often harder than adapting an active learning system.

2. **Adapting an IES to a new languages**

   Most of the existing IE systems are designed for textual data in a single language, in general English (in special cases in Chinese or Japanese). The task to make an IE system able to handle textual data in other languages is in many cases a very difficult one. Some of the Asian languages, like Chinese, are good examples to illustrate this difficulty. In Chinese, words are not delimited by a white-space. That makes an additional step in the process of IE necessary, namely the word segmentation step. Word segmentation is in many cases not a trivial task, because it requires an original lexicon. It is clear that it is not easy and feasible to build an original lexicon for a language each time. Additionally the grammar has to be changed, too. Some works on automating these steps are ongoing.

3. **Adapting an IES to different text genres**

   It is a common practice that IESs are trained on a corpus of documents with a specific genre. But a portable IES has also to be able to handle documents with different genres, because specific text genres (e.g. medical abstracts, scientific papers, police reports) may have their own lexis, grammar, discourse structure, etc.

4. **Adapting an IES to different types of data**

   One can broadly say that an IE system has to extract relevant information from text. The term of “text”, in fact, is not restricted and can have any behaviour. Thus, text can be in form of emails, newswire stories, military reports, scientific texts, and so on, which have very different formats. An email, for example, does not have a pre-defined or predictable format. It is only a free natural language text. Newswire stories, in contrast to emails, have a specific format. They have a title and mostly abstract the principle topic in the first paragraph.

   Additionally, there is the fact that the widespread use of the internet makes texts in different structural forms available, such as (semi-)structured HTML or XML files. To adapt an IES, which was initially developed for a specific type of text, is a non-trivial task even if the various types of text are about the same domain.

1.3.5.3 Scalability

The problem of scalability of an IES has two relevant dimensions. First, an IE system must be able to process large document collections. This dimension causes often no problems because IESs use in general simple shallow extraction rules, rather than sophisticated slow techniques. Second, an IE system must be able to handle different data sources. For example, weather information from different forecast services can have different formats; therefore, the extraction rules of the system must contain customized rules for such different sources. An IE system that is able to master both dimensions would use, with a high probability, the active learning approach. [Kushmerick & Thomas, 2002]

1.4 Information Integration

In a time in which we are overwhelmed with information from various data sources (e.g. databases, documents, e-mails, etc.) in very different formats, it is a big deal to make use of all this data in an efficient way. Thus research in the field of *Information Integration* (II) becomes more important today. **Information Integration**: "is the process of extracting and merging data from multiple heterogeneous sources to be loaded into an integrated information resource." [Angeles & MacKinnon, 2004]

To point at the differences of II today compared to the past, I quote the following [Halevy & Li, 2003, p.3]:

> First, we noted that the emergence of the WWW and related technologies completely changed the landscape: the WWW provides access to many valuable structured data sources at a scale not seen before, and the standards underlying web services greatly facilitate sharing of data among corporations. Instead of becoming an option, data sharing has become a necessity. Second, business practices are changing to rely on information integration – in order to stay competitive, corporations must employ tools for business intelligence and those, in turn, must glean data from multiple sources. Third, recent events have underscored the need for data sharing among government agencies, and life sciences have reached the point where data sharing is crucial in order to make sustained progress. Fourth, personal information management (PIM) is starting to receive significant attention from both the research community and the commercial world. A significant key to effective PIM is the ability to integrate data from multiple sources.

We can distinguish between several kinds of II [Brujn, 2003][Breu & Ding, 2004]:

1. **Technical Information Integration:** This kind of integration can be split into two levels: the hardware (platform) level and the software (platform) level. The hardware level encompasses differences in the computer hardware, the network architecture, the used protocols, etc. The software level encompasses differences in the used operating system, the database platform, etc.
2. **Structural Information Integration:** The structure of the data may be based on different principles, as for example relational database tables, hierarchical trees, etc.
3. **Syntactical Information Integration:** This encompasses differences of the data formats, as for example databases, plain text, etc. The different naming of the same entity in different databases can be also an example for this kind of integration problem (personal_id in one database and p_id in another database to name the same entity, namely the identification number of a person).
4. **Semantic Information Integration:** This kind of integration is the most difficult one. It encompasses different intended meanings of similar concepts in a schema. It becomes a standard to give concepts self-describing names. But the meanings that different users understand by looking only at the names are often not unique. It is possible, that two concepts with the same name are assigned really to different meanings (homonyms) or, that the same concepts are named differently in two schemas (synonyms).

We can distinguish between two fundamental aspects of II: *Data Integration*, and *Function Integration*. The definition I used above for II can also be used for *Data Integration*, because *Data Integration* deals with the problem of making heterogeneous data sources accessible by using a common interface and an integrated schema. A "common interface" should pretend the user that the collection of data is from a single data source. Function Integration tries to make local functions from disparate applications available in a uniform manner. Such an

integration solution has to pretend the user that the collection of functions is homogeneous. [Leymann & Roller, 2002]

An important derivative of Function Integration is used in enterprises and is named *Enterprise Application Integration* (EAI). In the centre of an EAI system is an integration broker, which acts as a hub between connected applications and routes the messages between them. Because of the behaviour of EAI systems, their capabilities for data integration are limited. EAI solutions, often, only provide access to one source at a time. But because of the fact that business transactions become more complex and often require information distributed across multiple data sources, data integration platforms has to be developed to complement EAI solutions. [Nimble]

A system that provides its users a uniform interface to a multitude of heterogeneous, independently developed data sources is called an **Information Integration System (IIS)**. A user of such a system has not to locate the data sources, to interact with each one in isolation and to manually combine data from multiple sources. [Halevy & Li, 2004]

There are two paradigms for II: *ad-hoc integration* and *global integration*.

For better understanding, I will give you first an example for an ad-hoc integration process from everyday life. Assume that you have developed an optimization algorithm for a graph drawing problem and have implemented it. In the next step, you will test your implementation with simple test graphs to get an idea of whether your implementation works or not. Therefore, you have created some test graphs in a format that your implementation can handle. After the first testing, you will test your implementation thoroughly with real world test graphs of higher complexity. But, the test graphs you can find are all in a different format than your program is prepared for. At this point you have two options: you can either rewrite your implementation to be also able to handle this input format, or you can write a script to transform the unsuitable test graphs into a suitable format. Both options are equally hard, but if you are only a user of an implementation and have no access to the source code or you have not the knowledge to make the required changes, you have just the latter option. For each graph format that the implementation cannot handle you must write a new script. This kind of integration is known as **ad-hoc information integration**.

Ad-hoc information integration solutions are not scalable and portable, because they are established for a single case with certain requirements and they are not applicable in different cases with different requirements. It is even harder to maintain such a solution, because if the requirements change the solution has also to be changed. This might not be requiring much effort for a personal user with simple demands but the situation is quite different in a business environment. First, the requirements for an II solution are complex and underlie continuous changes. Second, the amount of different applications that must operate together is large. With every new application that has to be integrated, new integration solutions must be established.

Global integration tries to overcome the disadvantages of the ad-hoc information integration and works quite different. An IIS, designed with this approach consists of a global schema, source schemas, and a mapping between the global schema and the source schemas that acts like a mediator.

We can distinguish between four kinds of mapping approaches [Lenzerini, 2002]:

1. **Local-as-View (LAV) or source centric:** The sources are defined in terms of the global schema.
2. **Global-as-View (GAV) or global-schema-centric:** The global schema is defined in terms of the sources.

3. **GLAV:** A mixed approach.
4. **Point-to-Point (P2P):** The sources are mapped with one another without a global schema.

To choose one of these approaches we have to consider the advantages and disadvantages of each one. In the following, I will give an overview of the pro and contras of the LAV approach, the GAV approach, and the P2P approach.

In the *LAV approach*, the quality depends on how well the sources are characterized. This approach promises high modularity and extensibility, because if one source is changed, only the definition of the source has to be changed. [Lenzerini, 2002] This approach is best suited when many and relatively unknown data sources exist and there is a possibility for adding or deleting data sources. [Borgida, 2003]

In the GAV approach quality depends on how well the sources are compiled into the global schema through the mapping. Because the global schema is defined in terms of the sources, the global schema has to be reconsidered every time a source changes. [Lenzerini, 2002] This approach is best suited when few and stable data sources exist. [Borgida, 2003]

P2P integration addresses only an isolated integration need and is therefore not suitable for reuse. For each pair of sources, a different mapping must be developed, which requires many hours of effort. Even then the mapping may fail to deliver the full range of the desired results. Each time a source is changed or a new source has to be integrated new mappings must be developed which means additional effort. With every new source, the number of links that must be added, tested, deployed, and maintained grows geometrically. Integrating 2 data sources require 1 link, 3 sources require 3 links, 4 sources require 6 links, and 5 sources require 10 links, and so on. [Nimble]

1.4.1 Application Areas of II

Among other information technology (IT) areas the field of database systems was one of the first fields that show interest in II research. The reason may be that database systems become the most commonly used structured data storage systems. The increase of the amount of such systems takes with it the interest in integrating databases from different systems.

II has become a recent issue also in business areas. I have quoted the following list to give an overview of IT and business areas that often create requirements for II [Alexiev, 2004, p.5, 6]:

> The following IT and business areas often create requirements for data integration:
>
> - Legacy application conversion and migration: convert data to a new application before retiring the old application.
> - Enterprise Application Integration and Application-to-application (A2A) integration, where applications existing within the enterprise should be made to inter-operate.
> - Business/executive reporting, OLAP, multidimensional analysis: regular loading of operational data to a data warehouse for easier analysis and summarization.
> - Business-to-business (B2B) integration between business partners.
> - Business Process Integration: coordination (“orchestration”) of separate business processes within and across enterprises in order to obtain a more synergistic and optimized overall process. Includes needs for Business Process Modelling, enacting, workflow, data modelling and integration.

1.4.2 Challenges

According to [Jhingran et al., 2002], II has three dimensions that make the task of managing the data more complex: *heterogeneity*, *federation*, and *intelligence*. I will shortly describe these three dimensions.

1. **The *heterogeneity* of data:** Currently, IISs have to deal with structured (e.g. databases, etc.), unstructured (e.g. text, audio, video, etc.), and semi-structured content (e.g. XML documents, etc.).
2. **The *federation* of data:** Today, data sources needed to be integrated are mostly distributed over multiple machines in different organizations. The federation problem encompasses the question of who owns and controls the data and the access on the data. Privacy and security issues become also important, because every organization has different security and privacy policies.
3. ***Intelligence*:** Another important issue is that of analyzing the data to turn the data into information, and more precisely into intelligence (e.g. detecting trends in a business, etc.).

Because of the mentioned three dimensions many challenges arise. Some of them are long term-challenges and others are challenges that currently occupy the attention of II researchers.

The long-term goal of II and the capabilities of systems reached this goal are explained in [Halevy & Li, 2004, p.3, 4]:

> The long-term goal of information integration research is to build systems that are able to provide seamless access to a multitude of independently developed, heterogeneous data sources. These systems should have the following capabilities:
>
> - integrate sources at scale (hundreds of thousands of sources),
> - support automated discovery of new data sources,
> - protect data privacy,
> - incorporate structured, semi-structured, text, multimedia, and data streams, and possibly inconsistent data,
> - provide flexible querying and exploration of the sources and the data,
> - adapt in the presence of unreliable sources, and
> - support secure data access.

In [Halevy & Li, 2004] there is also a list of specific challenges. I will explain these challenges here, to give you an idea about what work has to be done in the future. Hence, architectures are needed that enable large-scale sharing of data without no central control.

Reconciling heterogeneous schemas/ontologies: The fundamental problem of II is that the sources are heterogeneous, which means that the sources have different schemas and underlie different structuring methodologies. To integrate such heterogeneous sources, a semantic mapping is needed (often referred to as schema matching or ontology alignment). Today, these mappings are generated by humans. Because this task is time consuming and error prone, tools have to be developed to aid the human designer by its work.

Data-sharing with no central control: In many cases data cannot be shared freely between connected parts. Central control is therefore not possible in many environments. For such cases architectures are needed that enable large-scale sharing of data with no central control.

On-the-fly-integration: Currently, IIS's cannot be easily scaled up with new data sources. Thus, a challenge is to reduce the time and skill needed to integrate new data sources. This would make it possible to integrate any data source immediately after discovering it.

Source discovery and deep-web integration: Over the past few years the information (mostly stored in databases) behind the websites, which are queried on the fly when a user makes a request, deepened the web dramatically. To integrate these information sources is a very big deal and have potential. To discover these sources automatically, to integrate them appropriately, to support efficient query processing of user queries, etc. are some of the challenges that arise in that context.

Management of changes in data integration: IISs need to be able to handle updates of the data sources.

Combining structured and unstructured data: Currently, IISs are often not able to handle structured data sources (e.g. databases, XML documents, etc.) and unstructured text (e.g. web pages, etc.). This problem arises because the querying methods for both kinds of data sources are quite different. Whereas, structured data is queried by predefined query languages (e.g. SQL, XQL, etc.) unstructured text is queried by keyword searching. Thus, languages that are appropriate for such queries and efficient methods for processing them are needed.

Managing inconsistency and uncertainty: In an IIS, it often occurs that the data sources are inconsistent or uncertain. Methods must be developed to locate inconsistencies or uncertainties and to reconciling them.

The use of domain knowledge: The usability of IISs can be increased by using domain knowledge. Such knowledge can be used to guide the user by his work.

Interface integration and lineage: Often the data sources that are needed to be integrated in the system have its own user interfaces that give users easy access to the data. Hence, integrating such sources requires also combining the different visualisations.

Security and privacy: Data sources have often different security and privacy policies. By integrating multiple data sources this differences in the policies must be considered to ensure security and privacy.

1.4 Wrapper

In order to understand the usefulness of so-called *wrappers* better, the notion of "semi-structured documents" must be explain first.

As the term "semi-structured" implies, this kind of structure is a form between free text and fully structured text. The best-known examples for semi-structured documents are HTML documents. HTML is a mark-up language that contains text and predefined tags to bring more structure into the document. The text in such documents is often grammatically incorrect, and does not always contain full sentences. This makes it difficult to apply standard IE techniques on such documents, because IESs use linguistic extraction patterns. Thus, to extract information from such documents specialised IE systems are needed. Programs with the aim to locate relevant information in semi-structured data and to put it into a self-described representation for further processing are referred to as **wrappers**. It seems, as if IESs and wrappers do just the same, but the application areas are different.

The most widespread application area of wrappers is the World Wide Web with its unlimited amount of web sites that are mostly semi-structured. The differences between the structure of each document in the web and the fact that even sites of, for example, the same company are changed periodically, makes it obvious that building such programs by hand is not a feasible

task. This lead to two main problems in this field: *wrapper generation* and *wrapper maintenance*. [Chidlovskii, 2001]

1.4.1 Wrapper Generation

Manual *wrapper generation* means that the wrappers are written by hand using some sample pages. This procedure is time-consuming, error prone and labour-intensive. It is not scalable that means, even a little change in the page structure demands a re-writing of the extraction rules. The automatic or semi-automatic generation approach is a result of this search for ways to overcome these limitations.

The approaches of *automatic* or *semi-automatic wrapper generation* use some machine learning techniques based on inductive learning. These can base on heuristics or domain-knowledge. Although, the heuristic approach is relatively simple, it can only extract a limited number of features. The knowledge-based approach, on the other hand, tries to make use of the domain knowledge and so to build powerful wrappers. [Yang et al, 2001]

1.4.2 Wrapper Maintenance

Wrapper maintenance is the challenge of keeping a wrapper valid. Because, a wrapper cannot control the sources from which it receive data. A little change in the structure of a web site can make the wrapper useless (non-valid). The fact that some web sites change its structure periodically makes the task only harder. There are two key challenges to wrapper maintenance: *wrapper verification* (i.e., determining whether the wrapper is still operating correctly), and *wrapper re-induction*. The second challenge is more difficult, because it requires the change of the rules used by the wrapper. Even the *wrapper verification* task is not a trivial one, because the sources may have changed either the content or the formatting, or both, and the verification algorithm must distinguish this two. [Kushmerick & Thomas, 2002]

CHAPTER 2

Information Processing with PDF Files

In this chapter, I will give a short description of the *Portable Document Format* (PDF), the native file format of the Adobe™ Acrobat™ family. My aim is just to introduce this file format and not to give a full description of it. This chapter is based highly on the "PDF Reference Manual" written by Bienz and Cohn [1996]. For more detailed information I refer to this manual.

2.1 The Portable Document Format (PDF)

Bienz and Cohn defined PDF as follows: "PDF is a file format used to represent a document in a manner independent of the application software, hardware, and operating system used to create it." [Bienz & Cohn, 1996, p.5]

Further, Merz describe the PDF file format as: "PDF is a file format for saving documents which are graphically and typographically complex. PDF ensures that layout will be preserved both on screen and in print. All layout-related properties are fixed component of a PDF file and do not allow any variation in its interpretation – the appearance of a PDF document is completely fixed." [Merz, 1998, p.4]

The PDF file format is the successor of the PostScript page description language which was also initiated by Adobe™, and came out in the early 1990's. The PostScript page description language is a programming language like BASIC or C++, but is designed only to describe extremely accurately what a page have to look like. During the development of PDF, Adobe tries to avoid the weaknesses of the PostScript format. For example, PDF uses the same imaging model as PostScript but does not use the programming constructs to keep the simplicity of the format.

PDF is a layout-oriented representation format. Layout-oriented means, that the human readability is in foreground, rather than the machine readability. As you will see later on, layout-oriented representation formats are not that suitable for further processing. It could be hard to extract even text from PDF files, because the text can be, in the worst case, saved as a graphic, which would require the use of more complex algorithms, such as text recognition algorithms. [Gartner, 2003] I will give you a more detailed overview of the challenges of extracting information from PDF files later in 2.4.

2.2 Main Architecture of a PDF File

PDF files consist of four sections: a one-line header, a body, a cross-reference table, and a trailer.

1. **Header:** The header line is the first line of each PDF file. This line specifies the version number of the PDF specification to which the file adheres.
2. **Body:** The body of a PDF file consists of a sequence of indirect objects representing the document. In the body, there can also be comments. Comment can appear anywhere in the body section.
3. **Cross-reference table:** This table contains information about where the objects in the PDF file can be found. Every PDF file has only one cross-reference table, which consists of one or more sections.
4. **Trailer:** The trailer enables an application reading a PDF file to find the cross-reference table and thus the objects in the file.

No line in a PDF file can be longer than 255 characters long. The last line of a PDF file contains the string "%%EOF" to indicates the end of the file.

There are three kinds of PDF files [Weisz]:

1. **PFD Normal:** If you produce your text in a word processing or publishing system, with a PDF output capability, the PDF file you get is a PDF Normal file. That means that your file contains all the text of the page. Such files are relatively small.
2. **PDF Image Only:** This type is easy to produce. It is only an image of the page and contains no searchable text. Such files are fairly large and there is no possibility to search for text. The image quality depends on the quality of the source document and on the scanning operation.
3. **PDF Searchable Image:** Such files contain the image of the page and the text portions of the image. This enables a user to search for text. These files are usually larger than PDF Normal files.

2.3 Strengths and Weaknesses of PDF

If we surf the web we can find PDF files in heaps. Once technical details of an amazing five mega pixel digital camera, once a statistic about the last two years incomes of an enterprise, and once a brilliant crime novel of Sir Arthur Conan Doyle is saved in a PDF file. The widespread use of this file format must have its reasons. Thus, I will try to explain you the strengths of this file format shortly.

I mentioned that the PDF file format was created for a special purpose – to make it possible to build a document in an environment and view it in a, maybe, completely different environment without any difference. This specificity implies that PDF is not the file format par excellence for all purposes. Thus, I will try to explain the weaknesses of this file format, too.

The PDF format has a lot of properties, which makes it a demanded format to use in mostly all hardware environments and operating systems.

- **Portability:** This is the most exciting property of PDF. PDF files use only the printable subset of the ASCII character set which leads to the fact that PDF files are extremely portable across diverse hardware and operating systems environments.

- **Compression:** PDF supports a number of compression methods to keep the file size within limits, even if the document contains images or other storage impressive data.
- **Font independence:** The fact that people like to use different fonts in their documents, sometimes very extraordinary ones, is a challenge in document exchange. It could happen that the receiver of a document has not the fonts to re-create the document in her environment. PDF brings a solution to this problem, by saving a *font descriptor* for each font used in the file. Such a descriptor includes the font name, character metrics, and style information. This additional information does not affect the file size, because one font descriptor takes only 1-2K of storage and contains all the information needed to simulate missing fonts on the receiver side. This does not apply for so called *symbolic fonts* (i.e., a font that does not use the standard ISOLatin1 character set).

These three properties are more important in the reader's point of view. There are several other properties which are important for PDF developers (i.e., for people who want to build or change PDF files):

- **Random access:** Every PDF file has one cross-reference table which contains information about the locations of the objects in the file. This permits random access to each object in the file, thus the whole file needs not be read to locate any particular object. Because of the fact that the cross-reference table has a fixed location, namely the end of the file, a developer can easily get the information she wants about the location of a specific page or another object and must not go through the whole document to find what she wants.
- **Incremental update:** This property allows a user to easily change a file. Every time a change is made, the cross-reference table will be updated and the changed objects will be added to the file. The original data will remain unchanged. This allows a user to simply undo changes by deleting one or more added objects and cross-reference entries.

All of the mentioned properties form the strengths of the PDF format. On the other side there are some weaknesses, too. Merz explains the main disadvantage of PDF as follows [Merz, 1998, p.6]:

> PDF's biggest advantage is also its biggest drawback: it is purely layout-oriented and is not concerned with the text and overall structure of a document. The basic unit of a PDF document is the page. It can contain text, graphic and hypertext elements. The text is not saved as structural objects like paragraphs or headings, but, as in PostScript, in layout-related elements: as characters, words, or lines. This makes it much more difficult to modify or edit the text later on, as the structural information can not easily be recreated.

This disadvantage implies that it is hard to use a PDF file for further processing. It is understandable that many people are interested in further processing of PDF files, because today a lot of information is saved in PDF files and must be extracted in some way or another. Several tools are available, both commercial and free, to extract information from PDF files and save it into a more structured file format such HMTL or XML. I will present some of these conversion tools in CHAPTER 4.

2.4 Extracting Information from PDF

I mentioned that the PDF file format fits not the needs of all people, because it is hard to extract information from these files. The term “information” can be mean several things in the context of a PDF file, because PDF files can contain text, graphic and hypertext elements.
PDF was designed to be a print-layout format, and not intended to be editable. Thus, an extraction tool would have a lot of problems to face.

There are several elements of PDF that may produce errors during the conversion process [Gross, 2003]:

1. Word Spaces,
2. Hyphens,
3. Emphasis, Super and Subscripting, and
4. Special Characters and sub-fonting.

After looking at this possible error sources, we can say that we might consider that the converted information is correct, but we can never be sure of it. Therefore, it would be meaningful to use some correction algorithms to overcome such situations.

Extraction structure from a PDF file is another non-trivial issue, because PDF specifies only the position of elements in a page, and tell us nothing about the structure of the documents. A conversion tool that wants to store the extracted documents in a structured file format must use additional algorithms to rebuild the structure of the original document.

Some structural elements cause usually problems [Gross, 2003]:

1. **Multiple Columns:** Many documents have structured their page layout in form of two or more text columns. Such a form is usually seen in newspaper articles, or technical reports. This problem is not so hard to solve, but in some cases (e.g., too short columns) it would be harder to detect multiple columns, which would lead in a completely wrong extraction.
2. **Paragraph Delineation:** In PDF there is nothing that indicates the end of a paragraph. The extraction tool must guess where a paragraph ends.
3. **Page Header and Footer:** In PDF there is nothing that marks a text as footer or header. The extraction tool can only make use of the fact that such elements appear on every page to state that an element is a page header or footer.
4. **Tables:** These elements are one of the hardest elements to extract and even to recognize. There are no rules for creating tables, which means that a table have no predictable structure. The extraction tool must use heuristics to recognize and decompose the table structure. In Section 5.1.2, I give a detailed explanation about the problems of table extraction.
5. **Graphics:** Graphics are often used elements in PDF files. Sometimes the whole PDF file consists of only graphic elements, for example, if you scan some paper and save it as a PDF file. Extracting text from such files requires special recognition algorithms. In other cases, if the graphic is really a graphic, it may be still hard to guess which part of the file belongs to the graphic or not.
6. **Mathematical Equations:** These elements are highly complex elements that are not so easy to extract. The simplest way is to leave these elements as images.

CHAPTER 3

Information Processing with XML Documents

In this chapter I will give an introduction to the famous *extensible mark-up language* (XML). The information in this chapter highly depends on the book "Learning XML". [Ray, 2001] More detailed information about XML can be found in this book.

Furthermore, I will point out the properties of XML that makes it desirable for IP.

3.1 History and Description of XML

At the beginning, there was the *standard generalized mark-up language* (SGML), a descendent of the *generalized mark-up language* (GML), developed by Charles Goldfarb, Edward Mosher, and Raymond Lorie. SGML is a powerful meta-language (i.e., it can be used to build a mark-up language for documents) but it is also complex and was used generally by organisations that can afford both the software and the cost of maintaining complicated SGML and had complex publishing requirements. Thus, we cannot define SGML as a language for everyday use, for example, for web publishing.

XML was developed by an XML Working Group formed under the auspices of the World Wide Web Consortium (W3C) in 1996 as a subset of SGML, to simplify SGML and to make it applicable for general purpose. This consortium was created in October 1994. Their aim is described on their website as follows: "to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability." W3C publishes *Recommendations,* which are specifications or sets of guidelines that are reviewed by the members or other interested parties and received their endorsements of the Director. The Recommendation for XML contains, among other things, the design goals for XML. If you are interested in the complete recommendation, you are referred to the website of the W3C.[5] The design goals for XML are:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

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

6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

XML itself is, despite its name, not a mark-up language. XML is, like SGML, a meta-language, which means that the user can use it to build her own mark-up language. A mark-up language is a set of symbols (tags) that can be placed in the text indicating the role of a portion of text.

Whereas, mark-up languages, like HTML, focus on how to display data, XML focus on what the data actually is. HTML contains an amount of predefined tags that a web designer can use to give the web site the look she wants. The web designer is limited by those predefined tags and cannot use other ones. XML is different. In XML there are no predefined tags, you must create your own tags.

3.2 Architecture of an XML Document

The basic unit of XML information is the *XML document*. An XML document is composed of elements and other mark-up. Those elements nest inside each other and shape the content of the document. Each document has a *document element* or *root element* at the top level which contains the other elements. Thus, an XML document has a hierarchical tree-like structure.

XML elements can have additional information in form of *attributes*, which are name-value pairs and are positioned in the start tag of an element.

Assume that I want to make an XML document to store the information about the books I have read over the years. And assume that I want to require for each book a title. Such an XML document may look as follows:

```
<?xml version="1.0" ?>
<mybooks>
        <book year='1998'>
                <title> The Kreutzer Sonata </title>
                <author>
                        <name> Leo </name>
                        <surname> Tolstoy </surname>
                </author>
        </book>
        <book year='2003'>
                <title> The Adventures of Sherlock Holmes </title>
                <author>
                        <name> Sir Arthur Conan </name>
                        <surname> Doyle </surname>
                </author>
        </book>
</mybooks>
```

Figure 2: Example for a XML document

There are two ways to create a mark-up language with XML:

1. **Freeform XML**

   In this form, there are just a few rules about how to form and use tags. Such a rule can be, for example, that each start tag must have an end tag, and so forth. But there is no restriction about the ordering or the naming of the tags. An XML file that satisfies these rules is called a well-formed document. This way is highly error-prone. Let's look at the previous example with the books. It is possible that I forget somewhere to write the title of a book. I would not recognize it until maybe a software would fail, because of the absence of the title tag.

2. **Document Modelling**

   The process of defining a language in XML formally is called document modelling. There are two document modelling standards used with XML, the *Document Type Definition* (DTD) and *XML Schema*. Both of them are used to create specifications that lay out the rules for how a document has to look like.

A DTD consists of a set of rules or declarations that specify which tags can be used in a document, and what they can contain. If you use such a DTD, you must add a reference to this DTD in your XML document. This reference declares your desire to have the document validated. The DTD can be included in the XML document. In such a case the DTD is called an *internal DTD*. The DTD can also be specified in a separate file. In this case the DTD is called an *external DTD*.

Another document handling standard is known as XML Schema and provides an interesting alternative to DTDs. XML Schema allows you to design fields in more detail than DTDs can do. For example, you can specify the exact number of nested elements of an element. Another difference is that XML Schema use XML as its encoding syntax.

There is no reason to have only one standard for document modelling. The fact, that there are two of them increases the flexibility of the user. The user can choose what she wants in respect of her purpose.

A DTD for the previous example may look as follows:

```
<?xml version=”1.0” ?>
<!ELEMENT mybooks (book*) >
<!ELEMENT book (title, author) >
<!ATTLIST book
        year CDATA #REQUIRED
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name,surname)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
```

Figure 3: A possible DTD for the XML document in Figure 2

The language that I describe by this DTD contains several rules:

- The root element is *mybooks* and can contain zero or more *book* elements.
- Each *book* element must have two elements, one *title* and one *author* element.

- Each *book* element must also have an *attribute* named year.
- Each *author* element must have two elements, one name element and one surname element.

An XML document is called *valid*, if the document conforms to the rules of the DTD, referenced in this document. If a document has no reference to an external or internal DTD but conforms to the XML syntax, the document is called a *well-formed*. Thus, we can say that each valid document is also well-formed, but the opposite statement, that each well-formed document is also valid, is not always true.

3.3 Complementary Specifications

There are some specifications that complement XML like XSLT, XPath, XLink, and so forth. I mention here only those that are related with the presentation and transformation of XML documents.

The presentation of an XML document is another important issue that would interest all the people who think about preparing a document for viewing by human beings. One of the most important properties of XML is that it separates content and style. The content is the essence of the information and it is saved in an XML document. The style information is saved in a stylesheet file and describes the form of output the XML document can take. Thus, the XML documents and their stylesheets are complementary. The separation of content and style yields in a number of noticeable properties:

1. You can use one stylesheet for an unlimited amount of XML documents. That ensuring a consistent look over the documents and reducing the costs in case of an update. Assuming you want to change the look of your documents. In such a case you do not have to change all the documents one by one, you must only change the stylesheet.
2. You can support more than one versions of the same content in different designs. This could be useful, for example, if you want to support several output devices with different display sizes like a mobile device (e.g., handheld), or a printer.
3. The work of the author who builds the document and of the designer who builds the stylesheet is also separated. Thus, the author can concentrate on his writing while the designer can build the stylesheet without worrying about each others work.
4. The documents are more readable and manageable, because they have no additional styling elements in it.

The transformation (i.e., to convert an XML document from one form to another) of an XML document is one of the most exciting issues. Documents can be transformed by using the *Extensible Style Language for Transformations* (XSLT). The ability to transform XML documents increases your flexibility:

1. It allows you to store the data in one format and display it in another. For example, you can transform your content in a HTML file and display it in a web browser.
2. It allows you to extract the information you needed. For example, you get a document with detailed information about the books of a bookstore but you only need the titles and the authors. By means of XSLT you can extract only those parts you need and store it in a different document. (add)

3.4 Application Areas of XML

Because of the benefits of XML, it becomes a widely used technology in a large variety of areas like document publishing, data exchange, etc. Zisman, further, points at the usefulness of XML for the creation of vocabularies, depending on its flexibility. For a list of such XML applications, I quote the following [Zisman, 2000]:

> Because of its flexibility, XML allows the creation of many vocabularies, as standards for various domains. XML can be seen as the grammar and a particular application as the vocabulary. Examples of some XML application can be found in the following domains:
>
> - Finance/commerce: Open Financial Exchange (OFX), Open Trading Protocol (OPT), Financial Information eXchange Markup Language (FIXML).
> - Push publishing technology: Channel Definition Format (CDF).
> - Web automation: Web Interface Definition Language (WIDL), Web-based Distributed Authoring and Versioning (WebDAV).
> - Multimedia: Synchronized Multimedia Integration Language (SMIL), Precision Graphics Markup Language (PGML).
> - Software distribution: Open Software Description (OSD).
> - Scientific data: Chemical Markup Language (CML), Mathematical Markup Language (MathML).
> - Software Engineering: UML eXchange Format (UXF), XML Metadata Interchange (XMI).
> - Language oriented: Ontology Markup Language (OML), Conceptual Knowledge Markup Language (CKML).
> - Metadata: Research Description Framework (RDF), XML-Data, XML Metadata Interchange (XMI).

The benefits of using XML in the business area are also of great importance. This leads from the ability of XML to cover most of the business related tasks which have generally high costs when used old methods. One of these tasks is, for example, the exchange of data between organisations or even between departments of the same organisation. Organisations have generally different data formats in which they store their business relevant information. If two organisations have to interchange messages (or more generally data) an exchange standard must be established to guarantee correct and effective data exchange.

Another important area is the area of Web Services (i.e. tools that enable software to interoperate automatically). Some of the standards which are based on XML are, for example, the *Simple Object Access Protocol* (SOAP), *Web Services Description Language* (WSDL), and *Universal Description, Discovery and Integration* (UDDI).

Web designers also see the potential in XML and try to make use of this technology. The main reason is that XML separates content and layout. Thus, a web designer could build several layouts with XSLT for the same XML document with not much effort. This makes it easier for the web designer to create layouts for different devices (like Personal Digital Assistants (PDAs), a computer monitor, etc.) with the same XML document. [Roy & Ramanujan, 2000]

The application area of XML that I'm interested in is the area of IP. Thus, the next subsection will contain the benefits of XML usage for IP purposes.

3.5 XML in Information Processing

As I mentioned earlier, XML is a structure based language. This makes it attractive for further processing, either manual or automatic.

Manual processing with XML documents is much easier than with unstructured documents, because a human being can recognize the logical structure and make use of it. One can also make use of several complementary specifications of XML.

The complementary specifications of XML are also useful for IP purposes. XSLT is a good example for this because it can be used for each of the following types of tasks [Alexiev, 2004, p.60]:

- Transform from XML to a simple textual format (extract data).
- Transform from XML to a publishing format for printing. Here we target document that has little if any logical relation to the input document, so usually we go through the intermediate step of generating Formatting Objects (XSLT-FO). Then the FOs are transformed to printer-ready output (PDF, PCL, etc) by a FO Processor (e.g. Apache-FOP).
- Transform from XML to an XML format for publication, such as XHTML (and its poor cousin HTML), SVG, MathML, etc. Here the target document may contain some "garbage" (presentation stuff), but a lot of it has logical relation to the source.
- Transform from one XML schema to another XML schema. Here almost all of the generated data is logically related to the source.

  Most relevant to data integration are the first and last tasks in the list above:

- Extraction can pick up data fields from a semi-structured document and use them in further processing (e.g. to save to database).
- Transformation of data-centric XML schemas is key to XML data processing.

The interface between an application and an XML file is the *parser*. A parser reads an XML file and gives the application access to the content and structure of this XML file. There are two common types of XML parser application programming interfaces (APIs): *Document Object Model* (DOM) and *Simple API for XML* (SAX). These parsers have different properties and are suitable for different purposes.

The DOM parser, for example, is a parser that represents an XML document as a tree, whereas each element in the document is a node. DOM allows an API to access and modify parts of the document and to navigate in the document. DOM requires the document's entire structure in memory. Thus, it uses much memory and is slow.

The SAX parser, on the other hand, doesn't build the whole structure of an XML document, but scans an XML document and fires the events, such as element start or end. The handlers, implemented by the application programs, receive these events and do appropriate processing. SAX is fast and good suited for large documents, but it allows no re-processing.

There are many free parsers available in different programming languages which makes the work of the programmers much easier. [Jaideep & Ramanujan, 2000]

3.6 Limitations of XML

So far we have seen what XML actually is, how XML documents look like, what possibilities it offers, and its application areas. Now it is time to look at the limitations of XML, because it cannot be said that XML is the perfect solution for every purpose. It is limited in terms of the data types it supports: XML is a text-based format and has no means for supporting complex data types such as multimedia data. Further, XML is limited in terms of security: XML has no

facilities to ensure a secure exchange of XML documents. XML documents are only simple text documents without any encryption facilities. [Roy & Ramanujan, 2000]

CHAPTER 4

Comparison of Existing PDF Extraction Tools

I will give you a short report about how good existing conversion tools work. There exist just a few *pdf-to-xml* converters, thus I will begin with the comparison of *pdf-to-html* converters. The reason why I actually look for a conversion tool that converts a PDF file to an XML or a HTML file is that they are more structured (marked-up) documents and more useful for further processing.

Because I work with the extraction of table information from PDF-files, I will only compare the tools in respect of their ability to extract table information and not in respect of all the features they have. I will apply the tools on two tables. The first one is a "simple" table with only a few disjoint rows and columns. The second one is a more "complex" table with spanning rows and columns. Both tables are stem from the paper [Pinto et al, 2003]. The first table is on page six and the second is on page three of the cited paper.

Label	# of Labels	Recall	Precision
NONTABLE	14118	.979	.954
BLANKLINE	6186	.993	.998
SEPARATOR	492	.896	.942
TITLE	192	.542	.897
SUPERHEADER	92	.652	.909
TABLEHEADER	236	.462	.340
SUBHEADER	49	.918	.616
SECTIONHEADER	186	.441	.701
DATAROW	4497	.864	.912
SECTIONDATAROW	716	.547	.678
TABLEFOOTNOTE	163	.687	.896
TABLECAPTION	20	.000	.000

Figure 4: Example for a simple table

Figure 1: Example Table Snippet

Principal Vegetables for Fresh Market: Area Planted and Harvested by Crop, United States, 1997-99 1/ (Domestic Units)

Crop	Area Planted			Area Harvested		
	1997	1998	1999	1997	1998	1999
	Acres					
Artichokes 2/	9,300	9,700	9,800	9,300	9,700	9,800
Asparagus 2/	79,530	77,730	79,590	74,030	74,430	75,890
Beans, Lima	2,700	3,000	3,200	2,500	2,000	2,900
Beans, Snap	90,260	94,700	98,700	82,660	87,800	90,600
Broccoli 2/	130,800	134,300	137,400	130,800	134,300	137,300
Brussels Sprouts 2/	3,200	3,200	3,200	3,200	3,200	3,200
Cabbage	77,950	79,680	79,570	75,230	76,280	74,850

Figure 5: Example for a complex table

4.1 Comparison of PDF to HTML Converters

There are a few PDF to HML converters available on the web. Some of them are commercial and others are for free. I tried some of them to see whether they produce usable results or not.

4.1.1 PDF 2 HTML

This tool was developed by Gueorgui Ovtcharov and Rainer Dorsch. It is an open source tool that you can download for free[6]. The installation is quite simple. You just have to download and extract the zip-file, and the program is ready to run. On the website there is also a GUI for this program, in case you do not want to run the program from the command line.

You can see the standard look of the tool in Figure 6. With the “More Options” button you get the look in Figure 7, where you can additionally choose the creation of an XML document, otherwise the tool creates only HTML documents.

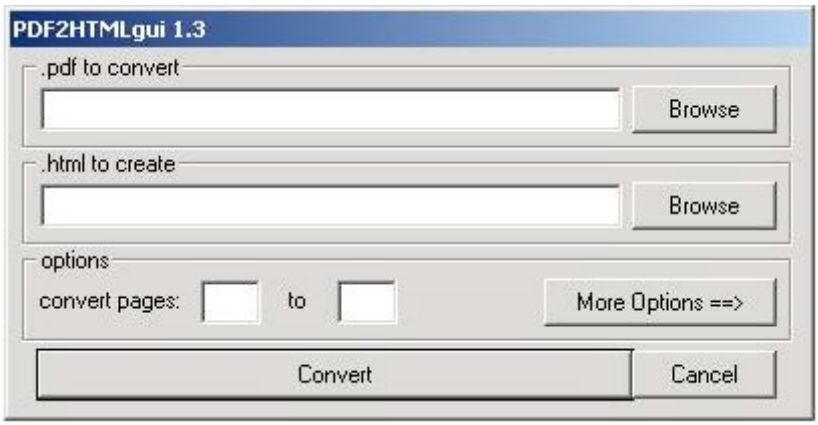

Figure 6: User interface of the pdf2html tool

[6] http://pdftohtml.sourceforge.net/

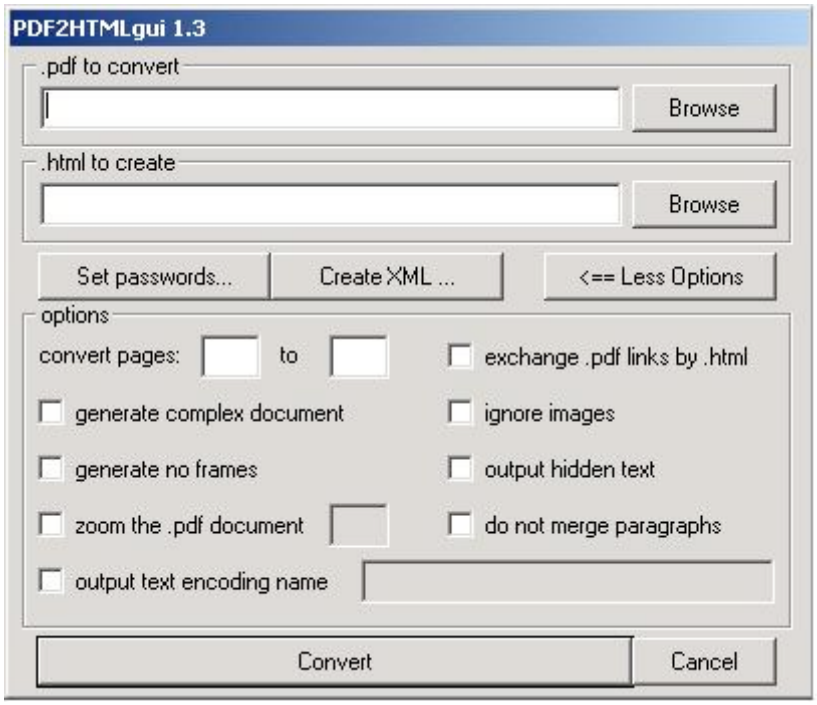

Figure 7: Extended user interface of the pdf2html tool

The resulting HTML-code for the table in Figure 4 is as follows:

```
Label<br># of Labels<br>Recall<br>Precision<br>
NONTABLE<br>14118<br>.979<br>.954<br>
BLANKLINE<br>6186<br>.993<br>.998<br>
SEPARATOR<br>492<br>.896<br>.942<br>
TITLE<br>192<br>.542<br>.897<br>
SUPERHEADER<br>92<br>.652<br>.909<br>
TABLEHEADER<br>236<br>.462<br>.340<br>
SUBHEADER<br>49<br>.918<br>.616<br>
SECTIONHEADER<br>186<br>.441<br>.701<br>
DATAROW<br>4497<br>.864<br>.912<br>
SECTIONDATAROW<br>716<br>.547<br>.678<br>
TABLEFOOTNOTE<br>163<br>.687<br>.896<br>
TABLECAPTION<br>20<br>.000<br>.000<br>
Table 4: CRF Continuous on Test Set<br>
```

Figure 8: The resulting HTML-code for the table in Figure 4

The resulting HTML-code for the table in Figure 5 is as follows:

```
Principal Vegetables for Fresh Market:<br>
Area Planted and Harvested<br>
by Crop, United States, 1997-99 1/<br>
(Domestic Units)<br>
-------------------------------------------------------------------------------<br>
:<br>Area Planted<br>:<br>Area Harvested<br>
Crop<br>:----------------------------------------------<br>
:<br>1997<br>:<br>1998<br>:<br>1999<br>:<br>1997<br>:<br>1998<br>:<br>1999
<br>
-------------------------------------------------------------------------------<br>
```

```
:<br>Acres<br>
:<br>Artichokes 2/<br>:<br>9,300<br>9,700<br>9,800<br>9,300<br>9,700<br>9,800<br>
Asparagus 2/<br>:<br>79,530<br>77,730<br>79,590<br>74,030<br>74,430<br>75,890<br>
Beans, Lima<br>:<br>2,700<br>3,000<br>3,200<br>2,500<br>2,000<br>2,900<br>
Beans, Snap<br>:<br>90,260<br>94,700<br>98,700<br>82,660<br>87,800<br>90,600<br>
Broccoli
2/<br>:<br>130,800<br>134,300<br>137,400<br>130,800<br>134,300<br>137,300<br>
:<br>2/<br>:<br>3,200<br>3,200<br>3,200<br>3,200<br>3,200<br>3,200<br>
Cabbage<br>:<br>77,950<br>79,680<br>79,570<br>75,230<br>76,280<br>74,850<br>
```

Figure 9: The resulting HTML code for the table in Figure 5

The only tags in these HTML-codes are the
 tags, thus the result gives us no information about the table structure. We cannot distinguish between "normal" text and table content.

4.1.2 PDF2HTML by VeryPDF

This program is a standalone tool for converting PDF files into continuous HTML files. You can download a trial version for free.[7] The installation is very easy. You must only double-click on the file you have downloaded and then the program is ready to run as in Figure 10.

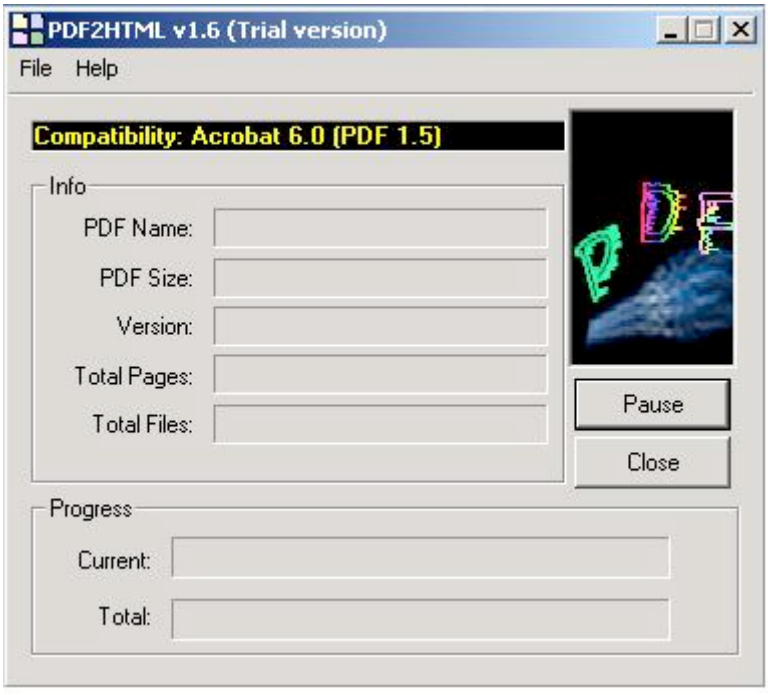

Figure 10: User interface of the PDF2HTML tool by VeryPDF

The HTML-code that this tool produces for the table in Figure 5 is stated below. I do not list the result for the first table, because the program returns very similar results.

```
<div style="position:absolute;top:101;left:210"><nobr><span class="ft1"> Principal
Vegetables for Fresh Market: Area Planted and Harvested</span></nobr></div>
<div style="position:absolute;top:113;left:304"><nobr><span class="ft1"> by Crop, United
States, 1997-99 1/</span></nobr></div>
<div style="position:absolute;top:125;left:357"><nobr><span class="ft1"> (Domestic
```

[7] http://www.globalpdf.com/pdf2htm/index.html

```
Units)</span></nobr></div>
<div style="position:absolute;top:138;left:75"><nobr><span class="ft1">----------
----------</span></nobr></div>
<div style="position:absolute;top:150;left:257"><nobr><span class="ft1">
:</span></nobr></div>
<div style="position:absolute;top:150;left:322"><nobr><span class="ft1"> Area
Planted</span></nobr></div>
<div style="position:absolute;top:150;left:451"><nobr><span class="ft1">
:</span></nobr></div>
<div style="position:absolute;top:150;left:504"><nobr><span class="ft1"> Area
Harvested</span></nobr></div>
<div style="position:absolute;top:162;left:204"><nobr><span class="ft1"> Crop :----------
----------</span></nobr></div>
<div style="position:absolute;top:175;left:257"><nobr><span class="ft1"> : 1997 : 1998 :
1999 : 1997 : 1998 : 1999</span></nobr></div>
<div style="position:absolute;top:187;left:75"><nobr><span class="ft1"> ----------
----------</span></nobr></div>
<div style="position:absolute;top:199;left:257"><nobr><span class="ft1">
:</span></nobr></div>
<div style="position:absolute;top:199;left:433"><nobr><span class="ft1">
Acres</span></nobr></div>
<div style="position:absolute;top:212;left:257"><nobr><span class="ft1">
:</span></nobr></div>
<div style="position:absolute;top:224;left:75"><nobr><span class="ft1">Artichokes 2/ :
9,300 9,700 9,800 9,300 9,700 9,800</span></nobr></div>
<div style="position:absolute;top:236;left:75"><nobr><span class="ft1">Asparagus 2/ :
79,530 77,730 79,590 74,030 74,430 75,890</span></nobr></div>
<div style="position:absolute;top:249;left:75"><nobr><span class="ft1">Beans, Lima :
2,700 3,000 3,200 2,500 2,000 2,900</span></nobr></div>
<div style="position:absolute;top:261;left:75"><nobr><span class="ft1">Beans, Snap :
90,260 94,700 98,700 82,660 87,800 90,600</span></nobr></div>
<div style="position:absolute;top:273;left:75"><nobr><span class="ft1">Broccoli 2/ :
130,800 134,300 137,400 130,800 134,300 137,300</span></nobr></div>
<div style="position:absolute;top:286;left:75"><nobr><span class="ft1">Brussels
:</span></nobr></div>
<div style="position:absolute;top:298;left:75"><nobr><span class="ft1">Sprouts 2/ : 3,200
3,200 3,200 3,200 3,200 3,200</span></nobr></div>
<div style="position:absolute;top:310;left:75"><nobr><span
class="ft1">Cabbage</span></nobr></div>
<div style="position:absolute;top:310;left:257"><nobr><span class="ft1"> : 77,950 79,680
79,570 75,230 76,280 74,850</span></nobr></div>
```

Figure 11: HTML code for the table in Figure 5

The HTML-code contains a more informative tag, namely the <div tag>, which has useful attributes like the absolute coordinates of the table entries. But as you can see, many of the numerical entries are handled as entries of an entire row. Thus it is not useful for my purposes.

4.1.3 Adobe Online Conversion Tool

Adobe® provides a service on his website, where users can have their PDF-files converted into HTML-files simply by specifying the URL of the PDF-file[8]. Then, the browser shows the resulted HTML-file.

For the table in Figure 4 this online-tool returns the following result:

```
<P>Label # of Labels Recall Precision <BR>
NONTABLE 14118 .979 .954 <BR>
BLANKLINE 6186 .993 .998 <BR>
SEPARATOR 492 .896 .942 <BR>
TITLE 192 .542 .897 <BR>
SUPERHEADER 92 .652 .909 <BR>
TABLEHEADER 236 .462 .340 <BR>
SUBHEADER 49 .918 .616 <BR>
SECTIONHEADER 186 .441 .701 <BR>
DATAROW 4497 .864 .912 <BR>
SECTIONDATAROW 716 .547 .678 <BR>
TABLEFOOTNOTE 163 .687 .896 <BR>
TABLECAPTION 20 .000 .000 <BR>
<P>
Table 4: CRF Continuous on Test Set <BR>
<P>
```

Figure 12: The HTML code for the table in Figure 4

As you can see, this HTML-code contains no information about the table structure or even about the occurence of a table.

4.2 Comparison of PDF to XML Converters

The conversion of PDF files to XML documents is a new field. The importance of this field increases with the increasing usage of PDF files, because of its environment independent representing features, and XML, because of its abilities to bring more structure in a document. Despite its importance, there are not many conversion tools available. The ones available are commercial tools, who do not even provide a demo version of the tool. The only possibility to test them is to send the organisation that provide the tool the PDF files you want converted and wait for a reply. I have done this, and I did not get a reply to date. Therefore, this subsection contains the results of only one tool instead of many tools for comparison as I had planned earlier.

The only free available tool I found was the PDF 2 HTML converter which also gives the ability of an XML output.

4.2.1 PDF to XML Converter

As I mentioned in 4.1.1, this tool can also be used to convert PDF-files to XML-files and it can be downloaded for free[9].

The result of this tool for the table in Figure 4 is as follows:

[8] http://www.adobe.com/products/acrobat/access_simple_form.html
[9] http://pdftohtml.sourceforge.net/

```
  <text top="465" left="483" width="34" height="12" font="0">Label</text>
  <text top="465" left="636" width="71" height="12" font="0"># of Labels</text>
  <text top="465" left="722" width="37" height="12" font="0">Recall</text>
  <text top="465" left="776" width="54" height="12" font="0">Precision</text>
  <text top="481" left="483" width="79" height="12" font="0">NONTABLE</text>
  <text top="481" left="672" width="34" height="12" font="0">14118</text>
  <text top="481" left="735" width="24" height="12" font="0">.979</text>
  <text top="481" left="806" width="24" height="12" font="0">.954</text>
  <text top="497" left="483" width="84" height="12" font="0">BLANKLINE</text>
  <text top="497" left="679" width="27" height="12" font="0">6186</text>
  <text top="497" left="735" width="24" height="12" font="0">.993</text>
  <text top="497" left="806" width="24" height="12" font="0">.998</text>
  <text top="512" left="483" width="86" height="12" font="0">SEPARATOR</text>
  <text top="512" left="686" width="21" height="12" font="0">492</text>
  <text top="512" left="735" width="24" height="12" font="0">.896</text>
  <text top="512" left="806" width="24" height="12" font="0">.942</text>
  <text top="528" left="483" width="43" height="12" font="0">TITLE</text>
  <text top="528" left="686" width="21" height="12" font="0">192</text>
  <text top="528" left="735" width="24" height="12" font="0">.542</text>
  <text top="528" left="806" width="24" height="12" font="0">.897</text>
  <text top="543" left="483" width="107" height="12" font="0">SUPERHEADER</text>
  <text top="543" left="693" width="14" height="12" font="0">92</text>
  <text top="543" left="735" width="24" height="12" font="0">.652</text>
  <text top="543" left="806" width="24" height="12" font="0">.909</text>
  <text top="559" left="483" width="107" height="12" font="0">TABLEHEADER</text>
  <text top="559" left="686" width="21" height="12" font="0">236</text>
  <text top="559" left="735" width="24" height="12" font="0">.462</text>
  <text top="559" left="806" width="24" height="12" font="0">.340</text>
  <text top="574" left="483" width="88" height="12" font="0">SUBHEADER</text>
  <text top="574" left="693" width="14" height="12" font="0">49</text>
  <text top="574" left="735" width="24" height="12" font="0">.918</text>
  <text top="574" left="806" width="24" height="12" font="0">.616</text>
  <text top="590" left="483" width="124" height="12"
font="0">SECTIONHEADER</text>
  <text top="590" left="686" width="21" height="12" font="0">186</text>
  <text top="590" left="735" width="24" height="12" font="0">.441</text>
  <text top="590" left="806" width="24" height="12" font="0">.701</text>
  <text top="605" left="483" width="73" height="12" font="0">DATAROW</text>
  <text top="605" left="679" width="27" height="12" font="0">4497</text>
  <text top="605" left="735" width="24" height="12" font="0">.864</text>
  <text top="605" left="806" width="24" height="12" font="0">.912</text>
  <text top="621" left="483" width="137" height="12"
font="0">SECTIONDATAROW</text>
  <text top="621" left="686" width="21" height="12" font="0">716</text>
  <text top="621" left="735" width="24" height="12" font="0">.547</text>
  <text top="621" left="806" width="24" height="12" font="0">.678</text>
  <text top="637" left="483" width="128" height="12"
font="0">TABLEFOOTNOTE</text>
  <text top="637" left="686" width="21" height="12" font="0">163</text>
  <text top="637" left="735" width="24" height="12" font="0">.687</text>
```

```
  <text top="637" left="806" width="24" height="12" font="0">.896</text>
  <text top="652" left="483" width="113" height="12"
font="0">TABLECAPTION</text>
  <text top="652" left="693" width="14" height="12" font="0">20</text>
  <text top="652" left="735" width="24" height="12" font="0">.000</text>
  <text top="652" left="806" width="24" height="12" font="0">.000</text>
  <text top="682" left="522" width="265" height="12" font="0">Table 4: CRF
Continuous on Test Set</text>
```

Figure 13: The XML code for the first table in Figure 4

As you can see, this tool returns an XML file with, among others, “text” objects and informative attributes, like the width and height of the text-object.

CHAPTER 5

Task Description and Implementation

If I have to describe my task in one sentence I would say: “My task is to implement a tool for extracting table information out of PDF files and for storing the extracted information in an XML document.”

In the previous chapters, I first gave you an overview of IP in general and of IE in more detail. My aim was to make clear what approaches exist actually, and how the architecture of an IES looks like. These realisations should assist me to implement a tool that fits the requirements best. Further, I gave you an overview of the two document formats that my tool must handle, PDF and XML. And at least I made a comparison of existing extraction tools, to see which of them gives me the best result to work with. After this comparison I decided to use the “pdf2html” tool. As I had mention in 4.2.1, this tool outputs XML documents with all the text elements of a PDF file in the ordering they occur in the original file. Thus, my task changed to “implement a tool for extracting table information out of an XML document and storing the extracted information in another XML document.”

Now, it would be meaningful to give you an overview about tables in general and about table recognition and table extraction in special.

5.1 Table Extraction

Table extraction is not a trivial task, indeed. It is important for IP purposes because tables are one of the most used elements for structuring information. Tables attract the attention of IP researchers because they contain in general information with a high density.

For better understanding the task of table extraction and its challenges, I will first give an overview of the common behaviour of tables. Then, I will describe the task of *table recognition*, because before table extraction can take place, the tables in a document must be identified. And last, I will explain the task of *table extraction*, its approaches and challenges.

5.1.1 Common Behaviour of Tables

The most common definition for tables is that “tables are a means for presenting and structuring data”. Tables can help the author of a document to present a piece of data in a structured form that helps the reader to better understand relationships between table entries. You can meet tables in almost any context, for example, in books, in scientific journal articles, in financial reports, on web sites, and so forth. [Tupaj, et. al, 1996]

There are no rules for creating tables. Thus, it is hard to classify tables, although some forms of tables have a fixed semantic such as truth tables (i.e., tables used in logic to determine whether an expression is true or whether an argument is valid).
Although, there is no unified theory of tables we can list some properties of tables as follows [Ramel et al, 2003]:

1. **Tables are ad hoc:**

   Every author creates her own table in a form and complexity so that it represents her data best.

2. **Tables have no identifying characteristics in common:**

   I mentioned that tables are ad hoc. Thus, there is no limitation or a predefined standard to build tables. Because tables can vary in structure as their authors, to identify common characteristics for tables is not possible.

3. **Tables have mostly a complex structure:**

   The simplest kind of table is that one where each entry spans over one cell and one row, and for each cell there exists a single header. But the information that authors like to put in tables are often not of a structure that they can be filled in such a simple format. Thus, many tables contain spanning rows or cells which increases their complexity.

4. **Tables have varying formats:**

   Some authors create tables with boundary lines (i.e., lines that points out the boundary of the table and/or the separation areas between rows and cells), whereas other authors just use spaces to achieve a table view. These tables are the most difficult ones to extract. On the one hand there is the recognition task. To detect such tables, many assumptions and rules are needed (e.g. if more than two tabs occur on the same line this line might be a data row). The decomposition task is even harder, because to correctly assign the texts between the tabs requires reliable rules. The developer of an extraction algorithm can build a rule, based on the assumption that there are so many tabs in the line as cells in a table. But when the algorithm has to face a table with a different format it would fail and the decomposition would be not correct.

5. **Tables contain different types of content:**

   Many tables contain text, figures, mathematical formulas, and so forth.

These properties make it harder to extract table information from documents and must be always keep in mind while implementing an extracting tool.

5.1.2 Extraction

In order to extract table information from a document, the first thing to do is to locate the table in the document. This task is referred to as *table recognition*. The difficulty of this task depends on the document structure in which the table is embedded. *Table recognition* in structured documents is easier than *table recognition* in unstructured documents.

Assume that we work with HTML files. HTML has a tag called *table* that indicates the presence of a table. But in the case of HTML files, we have to face the problem that many authors uses table tags for layout purposes only. These kinds of tables are not tables in the common sense and have to be classified as such. Thus, we are again back on square one and must find other methods to detect a table.

An example for unstructured documents can be PDF files. A number of works are done in the field of table extraction from layout based document formats. Some of them focus on methods that rely solely on layout based features such as boundary lines or separator lines between rows and columns. Such methods are not applicable if we have to face tables with no line elements, but spaces, which the author has used to achieve a table-like look.

Many people are interested in the field of table extraction, because tables have in general a high information density. The task of extracting table information from documents is not trivial indeed, and different problems show up depending on the document behaviour.
Several approaches exist in the field of table recognition and extraction (also called *table understanding*) [Wang, 2002]

1. **Predefined layout based approach:** The idea of this approach is to use templates of some table structures. The documents are checked and piece of data that match such a template are identified as tables.

   This is a simple approach and can have a satisfying performance, but it is hard to extend, because it would fail if any table with a different structure than those appearing in the template set.

2. **Heuristics based approach:** In this approach some predefined rules are created (e.g., tables with less then 3 rows are not really tables, and so forth) to use it as a decision base to check whether a piece of data is a table or not.

   It is clear that such a rule set must contain a lot of rules, to cover all the shapes a table can take. As the latter approach, this approach suffers also if a completely different form of a table appears and there is no applicable rule in the rule set.

3. **Statistical or optimization based approach:** This approach either requires no parameters or the needed parameters are obtained via training processes. These parameters are then used in the decision making process.

5.2 The Implementation

In this subsection, I will inform you about the work I had to do, about the initial state at the beginning of my development, and about my approach. Further, I will give an overview of the features of the *Graphical User Interface* (GUI) that I have implemented to give the user the ability to change the results of the extraction process.

5.2.1 Initial State

After the comparison of different conversion tools I have decided to work with the pdf2html converter, because it also allows the conversion in the XML format and this output is the most useful for my purposes. The output is an XML document that contains the text of the PDF file in form of text elements. I will explain you this by an example. The following figure is a snapshot of a part of a table, saved in a PDF file.

Table 1

Risk Factors for Conjunctivitis

Type of Conjunctivitis	Risk Factors
Allergic	
Seasonal	Environmental allergens.*
Vernal	Hot, dry environments such as West Africa; parts of India, Mexico, Central, North, and South America; and the Mediterranean area.
	Environmental allergens for acute exacerbations.*
Atopic	Genetic predisposition to atopy.
	Environmental allergens and irritants for acute exacerbations.*
Ocular cicatricial pemphigoid (OCP)	Unknown (genetic predisposition may exist).
	(Topical drugs may produce OCP-like disease, with spectrum of severity ranging from self-limited to progressive disease indistinguishable from OCP. Associated drugs include idoxuridine, pilocarpine, epinephrine, timolol, and echothiophate.)

Figure 14: Snapshot of a table about conjunctivitis

The pdf2html tool returns for this part of the PDF file the following XML output:

```
<text top="541" left="522" width="67" height="18" font="0"><b>Table 1</b></text>
<text top="579" left="412" width="288" height="18" font="0"><b>Risk Factors for
Conjunctivitis</b></text>
<text top="647" left="258" width="58" height="14" font="3"><b>Type of </b></text>
<text top="665" left="258" width="99" height="14" font="3"><b>Conjunctivitis</b></text>
<text top="665" left="469" width="87" height="14" font="3"><b>Risk Factors</b></text>
<text top="692" left="258" width="55" height="14" font="3"><b>Allergic</b></text>
<text top="719" left="258" width="56" height="14" font="1">Seasonal</text>
<text top="719" left="469" width="165" height="14" font="1">Environmental allergens.*</text>
<text top="743" left="258" width="41" height="14" font="1">Vernal</text>
<text top="743" left="469" width="342" height="14" font="1">Hot, dry environments such as West Africa;
parts of </text>
<text top="761" left="469" width="331" height="14" font="1">India, Mexico, Central, North, and South
America; </text>
<text top="779" left="469" width="183" height="14" font="1">and the Mediterranean area.</text>
<text top="806" left="469" width="321" height="14" font="1">Environmental allergens for acute
exacerbations.*</text>
<text top="833" left="258" width="42" height="14" font="1">Atopic</text>
<text top="833" left="469" width="209" height="14" font="1">Genetic predisposition to atopy.</text>
<text top="860" left="469" width="306" height="14" font="1">Environmental allergens and irritants for
acute </text>
<text top="878" left="469" width="97" height="14" font="1">exacerbations.*</text>
<text top="905" left="257" width="121" height="14" font="3"><b>Ocular cicatricial</b></text>
<text top="905" left="469" width="294" height="14" font="1">Unknown (genetic predisposition may
exist).</text>
<text top="923" left="257" width="134" height="14" font="3"><b>pemphigoid (OCP)</b></text>
<text top="932" left="469" width="342" height="14" font="1">(Topical drugs may produce OCP-like
```

```
disease, with </text>
<text top="950" left="469" width="320" height="14" font="1">spectrum of severity ranging from self-limited
to </text>
<text top="968" left="469" width="312" height="14" font="1">progressive disease indistinguishable from
OCP.</text>
<text top="986" left="469" width="327" height="14" font="1">Associated drugs include idoxuridine,
pilocarpine,</text>
<text top="1004" left="469" width="272" height="14" font="1">epinephrine, timolol, and
echothiophate.)</text>
<text top="1143" left="841" width="8" height="14" font="1">5</text>
```

Figure 15: XML code of the table in Figure 14.

We can see some properties of the output data:

1. The tool returns all of the text chunks in the PDF file as text elements.
2. The tool returns the text elements in the order they appear in the original file.
3. Each text element has five attributes: top, left, width, height, and font. These attributes give information about the exact position of the text in the original file and its size.
4. Text elements that were formatted as bold or italic have additional <*b*> or <*i*> tags.
5. The tool processes the PDF file line by line and splits text chunks that logically belong together. For example, the first row in the table "Type of Conjunctivitis" is split into two text elements, the first one contains the text "Type of" and the second contains the text "Conjunctivitis".

5.2.2 The Approach

After all these considerations I decided to make use of the heuristic based approach. For that purpose, I build a rule set which can be considered in two parts. The first part contains the rules about the table recognition task and the second one contains the rules about the table decomposition task.

The properties of the output data of the pdf2html tool listed in 5.2.1, affect the algorithm to recognize and decompose the tables. Before I give detailed information about the algorithm, I will explain some terms that I will use later on.

- **Single-Line:** A line with only one text element (i.e., just one element with the same top-value).
- **Multi-Line:** A line with more than one text elements (i.e., more than one consecutive text elements has the same top-value).
- **Multi-Line Block:** Group of consecutive multi-lines.
- **Average distance of a multi-line block:** The average distance between the top values of the lines in a multi-line block.
- **Maximum number of elements in a multi-line block:** The maximum number of elements in a multi-line in a multi-line block.

My algorithm extracts the table information in two steps. I called the first step *first classification* and the second step *second classification*. In the following, I will give the pseudo-codes of these two steps.

First classification:

1. Go through all the text elements on a page.
2. For each text element check whether there are other text elements that follow the first

one having the same top value, or not.
3. Put the text elements with the same top value together and mark this line as multi-line.
4. If there are no other text elements with the same top value, mark this line as a single-line.
5. The occurrence of one multi-line starts a multi-line block.
6. The occurrence of a single-line ends a multi-line block.

Figure 16: Pseudo-code for the first classification step

Sometimes it can be that the occurrence of a single-line has not to mean that a multi-line block ends. I will explain this on the following example.

Vernal	Hot, dry environments such as West Africa; parts of India, Mexico, Central, North, and South America;

Figure 17: Example for a single-line that not indicates the end of a multi-line block

Here the text elements "*Vernal*" and "*Hot, dry environments such as West Africa; parts of*" build together a multi-line. Then, the text element "*India, Mexico, Central, North, and South America;*" follows. In this case, this element not indicates the end of the multi-line block, but belong to a text element in the previous line. To take account of such cases, another rule comes into action:

> If a single-line occurs directly after a multi-line, check whether this single text element belongs to any of the text elements in the previous multi-line.

A text element belongs to one text element of the previous line if it lies under one of these text elements.

```
<text top="743" left="258" width="41" height="14" font="1">Vernal</text>
<text top="743" left="469" width="342" height="14" font="1">Hot, dry environments such as West Africa;
parts of </text>
<text top="761" left="469" width="331" height="14" font="1">India, Mexico, Central, North, and South
America; </text>
```

Figure 18: XML code of the table fragment in Figure 17.

As you can see the left value of the mentioned single text element and the second text element of the multi-line are the same, namely "469". In such cases the multi-line block goes on, indicating that the single-line belongs logically to the multi-line block.

After the first classification the multi-line blocks are identified. The next step is to combine multi-line blocks that belong to each other.

Second classification:

1. Go through all the multi-line blocks.
2. Determine the number of lines between each pair of multi-line blocks.
3. If the lines that lie between two multi-line blocks are less or equal than three, the two multi-line blocks belong together with a high probability. Thus, these two blocks are merged (i.e. the second block is attached at the end of the first block).
4. If the lines that lie between two multi-line blocks are greater than three and less or equal than five, these two blocks might belong together but with a less probability. The lines between these two blocks are tried to be assigned to the first block. If this process succeeds and there are no lines left between them, these two blocks are merged.
5. // Once the multi-line blocks are identified, the structure of each multi-line block has to
   // be determined. First, the lines that belong together (e.g., parts of the same sentence)
   // has to be identified.

6. Go through all lines in the multi-line block.
7. Compare the text elements on one line with the text elements on the next line.
8. If two text elements on continuous lines have almost the same left value, the same format and the difference of their top values are not that big, these text elements assumed to belong together. The difference of their top values has to be less than the average distance of this multi-line block. If two text elements fit these requirements they are merged. To merge two lines, all the text elements on the second line must be merged to the text elements on the previous line.
9. // Once the belonging lines are merged, we have to find a title if there is one.
10. Go maximal ten lines backward and look whether this line contains a title.
11. For each of these lines, check whether the text element on this line lies approximately at the centre of the multi-line blocks side boundaries or the text contains the string “table”.
12. // Whether or not a title is found, the next step is to determine the header part of the // table.
13. The first line with the amount of elements equal to the maximum number of elements in this multi-line block is the last line of the header. Thus, the next line is the first data row.
14. The lines going backward from the last line of the header are being explored.
15. For each text element on the header lines there are two possibilities: this element is itself a super-header (i.e. a header at the top of the hierarchy), this element has a super-header that lies above him. For the latter case the super-header for this element must be determined. There are four possibilities:
    a. The text element lies directly under a text element on the previous line.
    b. The left-point of the text element lies under a text element on the previous line.
    c. The right-point of the text element lies under a text element on the previous line.
    d. The text element lies completely on the left of a text element on the previous line.
    e. The text element lies completely on the right of a text element on the previous line.
16. If case d or e occurs the text element is assigned to the element that is nearest to this one.
17. // After constructing the header part, the data-rows are explored to assign each text // element to a header element.
18. For each line from the first data row line on, assign the elements on this line to a header element on the last header line using the rules listed above.

Figure 19: Pseudo-code of the second classification step

5.2.3 Limitations of the Approach

Because of the complexity of the task, the bugs in the pdf2html tool, and the heuristic based approach which cannot cover all table structure, you cannot assume that this tool returns you a perfect conversion for each table. You should expect that a post-process in form of changes in the user interface must be done in almost each case. Therefore, you should always control the output of the tool and you should use the “with interaction” option of the interface, because to correct the false points as early as possible is the best solution.

All these steps and rules, explained in 5.2.2 have the aim to construct the table as good as possible and likely to its original. But this is often not reached because of several reasons:

The tool that I used for getting the text elements out of the PDF file contains a number of bugs that affect the result of my heuristic based approach. This is a direct result of the nature of this approach. Namely, all the rules are based on assumptions. The basic and most important assumption is that the XML code returned from the pdf2html tool contains correctly extracted data. If that is not the case, the result of my implementation would suffer.

The pdf2html-tool has some known bugs that are listed in the readme-file of the implementation. But the worst bug is the following: “Plain (non-complex) output might not preserve the order of text sometimes.” This is the worst bug, because all the rules I have set, are based on the assumption that the text elements in the pdf2html output file are in the same order as in the original PDF file.

To meet these bugs I decided to implement a graphical user interface which can be used to make changes on the extracted tables.
But there are limitations of the pdf2html tool that cannot be addressed with the GUI. For example, the pdf2html tool has no facilities to extract text from a graphic in a PDF file. Thus, if a table in a PDF file is actually a graphic, no information can be extracted and my algorithm also cannot extract anything.

5.2.4 The Graphical User Interface (GUI)

The graphical user interface (GUI) allows the user to make changes on the extracted table. I will explain the usage and the benefits of this GUI by an example.

After starting the program, the following interface appears on the screen:

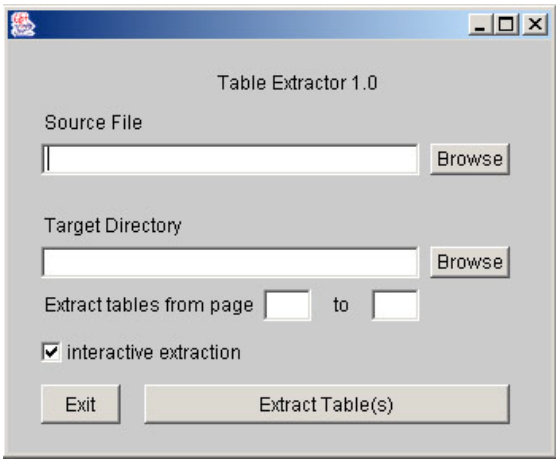

Figure 20: User interface of the extraction tool

Using this interface, the user can browse for a source PDF file and for a directory where the resulting files should be saved. The output of this tool consists of five files saved in the given directory. For a PDF file with the name “example.pdf” the resulting files would be:

1. **example.xml:** This file is the result of the used pdf2html conversion tool.
2. **pdf2xml.dtd:** This is the DTD for the example.xml file.
3. **output.xml:** This file contains the extracted table information in XML form.
4. **tables.dtd:** This is the DTD for the output.xml file.
5. **table_view.xsl:** This file contains the style-sheet for the output.xml file.

The user can set the interval of pages of the source file, which the program has to explore for finding tables. Depending on the selection of the “interactive extraction” checkbox, the program can either extract the tables and then can terminate, or can allow the user to make changes on the extracted tables. I will explain you the latter by an example.

As the source file, I will use the paper [Riloff, 1999] with setting the page interval between page 4 and page 18. On page 18 there is the following table, which is the table I want to be extracted.

Slot	AutoSlog			AutoSlog-TS		
	Recall	Prec.	F	Recall	Prec.	F
Perpetrator	.62	.27	.38	.53	.30	.38
Victim	.63	.33	.43	.62	.39	.48
Target	.67	.33	.44	.58	.39	.47
Total	.64	.31	.42	.58	.36	.44

Figure 21: An example for a table with spanning columns

As you can see, this is a complex table in the sense that it contains spanning columns and a header part with hierarchy level two.

After the program has extracted the tables in the file, the following interface appears.

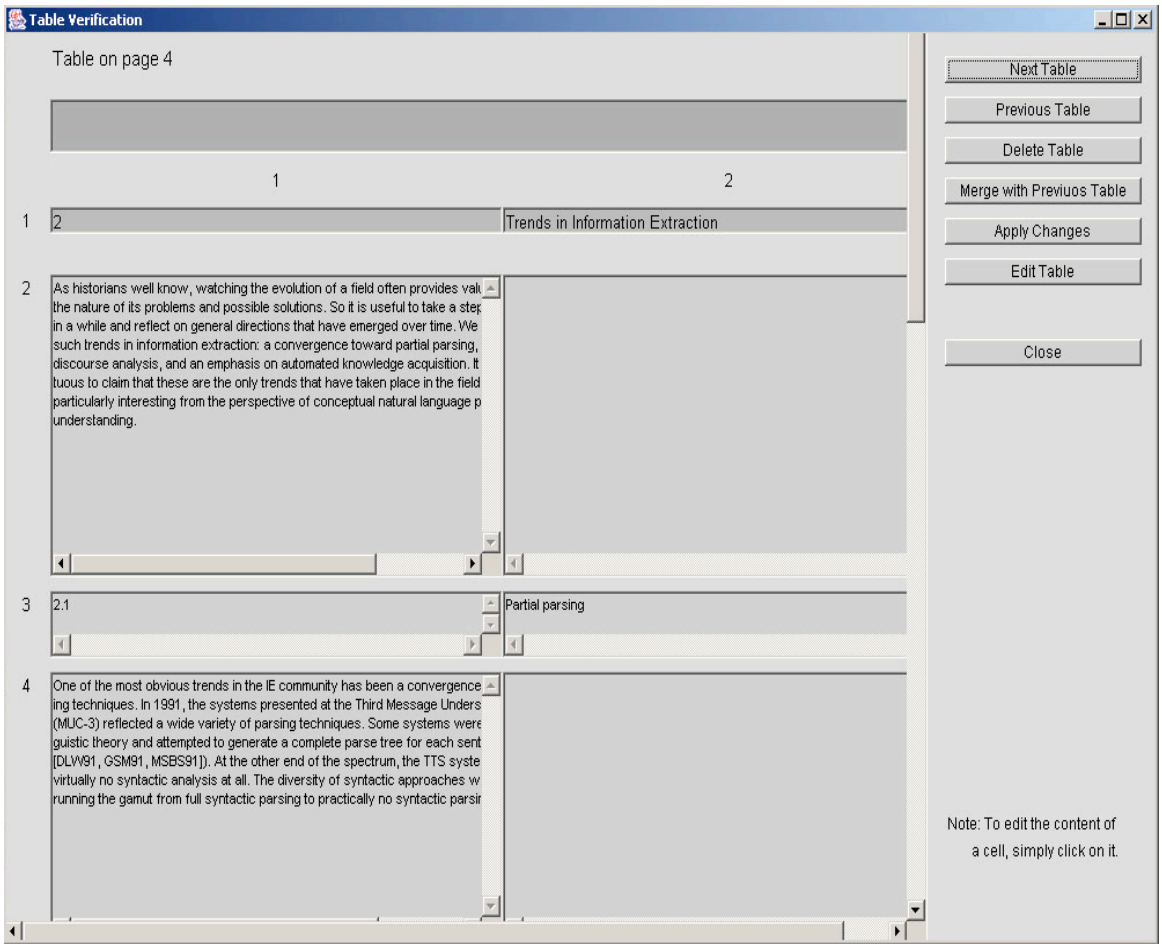

Figure 22: User interface that appear after extracting the paper [Riloff, 1999]

As you can see, the GUI consists of a panel where the extracted information is displayed, and of seven buttons which have various functionalities. First, I will explain the functionalities of the buttons and then the part with the extracted information.

Next Table: There is always the possibility that a file contain more than one table. In such cases the user might want to go through all the extracted tables to control and perhaps correct them.

Previous Table: In some cases the user might want to go back and take a look at a previous table and perhaps make additional changes on it.

Delete Table: Sometimes it can happen that the extracted information is not a real table or of not interest for the user. If the user don't want this part to be in the output file, a click on this button is sufficient.

Merge with Previous Table: Many files contain tables that are laid out on consecutive pages. Although the program consists of some rules for detecting such cases, it could happen that the program fail in such a situation. With this button, the user can easily merge a table which originally belongs to the previous table. In this context merge means that the actual table is attached to the end of the previous table.

Apply Changes: After making changes to the table structure, the user must click this button before closing the GUI or going to the next/previous table to apply these changes.

Edit Table: After clicking this button the following interface appears:

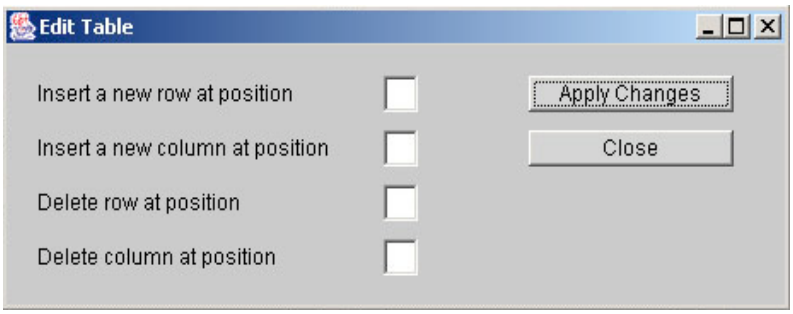

Figure 23: The interface that enables the user to change the table structure

This interface allows the user to insert additional rows/columns or to delete rows/columns in the table, by easily setting the index number of the row or column.

Close: A click on this button terminates the extraction process for the actual input file and closes the GUI.

Now, its time to explain the extracted information displayed in Figure 22. As you can see, the extracted content is not a real table. In fact, it is the part of the file displayed in Figure 24.

2 Trends in Information Extraction

As historians well know, watching the evolution of a field often provides valuable insights into the nature of its problems and possible solutions. So it is useful to take a step back every once in a while and reflect on general directions that have emerged over time. We will discuss three such trends in information extraction: a convergence toward partial parsing, a healthy respect for discourse analysis, and an emphasis on automated knowledge acquisition. It would be presumptuous to claim that these are the only trends that have taken place in the field, but these three are particularly interesting from the perspective of conceptual natural language processing and story understanding.

2.1 Partial parsing

One of the most obvious trends in the IE community has been a convergence toward partial parsing techniques. In 1991, the systems presented at the Third Message Understanding Conference (MUC-3) reflected a wide variety of parsing techniques. Some systems were grounded in linguistic theory and attempted to generate a complete parse tree for each sentence in a text (e.g., [DLW+91, GSM91, MSBS91]). At the other end of the spectrum, the TTS system [DGCN91] did virtually no syntactic analysis at all. The diversity of syntactic approaches was quite remarkable, running the gamut from full syntactic parsing to practically no syntactic parsing and everything in

Figure 24: Source of the extracted information in Figure 22

This part is extracted as a table because the pdf2html tool has extracted the string “2” as one text element and the string “Trends in Information Extraction” as another text element, both with the same top value. The same, for the line pointed at with the other arrow. Therefore, my heuristic identifies the lines pointed at with arrows, as multi-lines. The texts between these two multi-lines are identified as single-lines, and thus the heuristic merges the two multi-lines and creates the look in Figure 22. In this case the “Delete Table” button could be used by the user.

After deleting the table the next table appears on the screen.

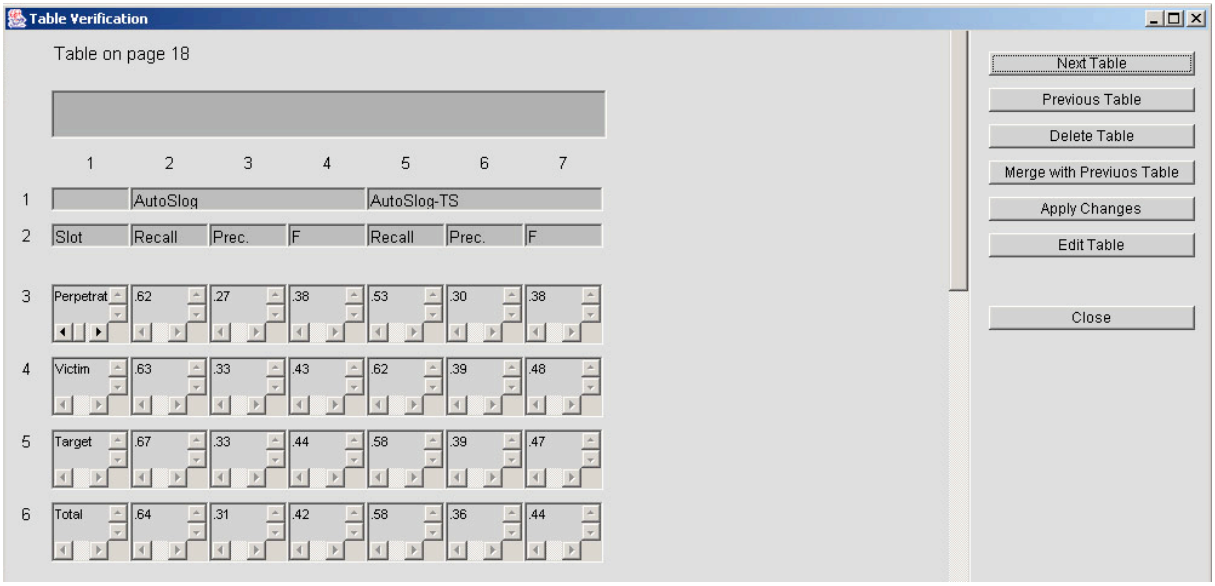

Figure 25: Part of the user interface for the extracted information of the table in Figure 21

As you can see, this is the extracted information for the table in Figure 21 and it is extracted correctly. Now, the user can change the table if she wanted to. There are several possible changes. Either the properties of the cells or the structure of the table (i.e. number of rows and columns) could be changed. To change the properties of a cell, the user has simply to click on it. After clicking on a cell, for example on the cell at position [3,5], the following interface appears.

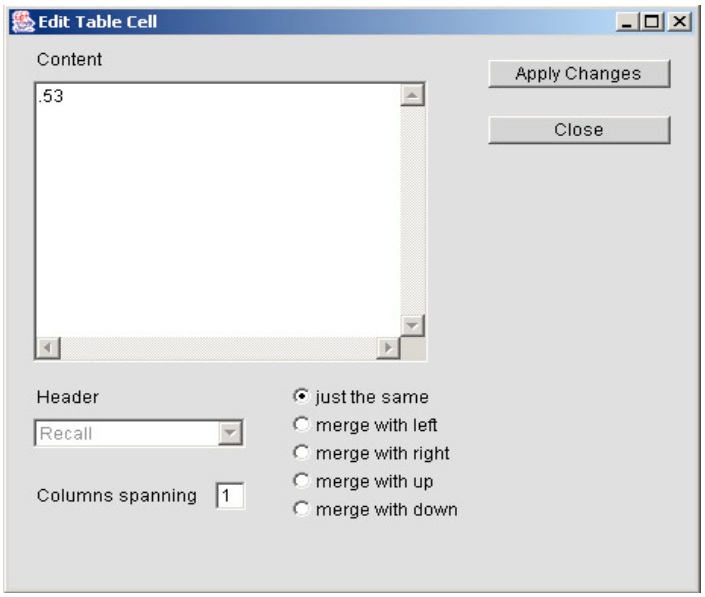

Figure 26: Content of the cell at position [3,5] in Figure 25.

The user can change the following properties of a cell:

Content: The content of the cell can be easily changed by clicking on the text area and change the text within.

Header: The pop-up menu allows the user to change the header to which the cell belongs to. This pop-up menu is not accessible for all kinds of cells, but only for cells that are in the header part of the table. If the cell is a header cell, it can be assigned to another header cell in the hierarchy level above.

Assume, that the user will change the super-header (i.e., header element at the top level of the hierarchy of the header part) of the cell at position [2,2] in Figure 25. After clicking on the cell, the following interface appears where the pop-up menu is activated.

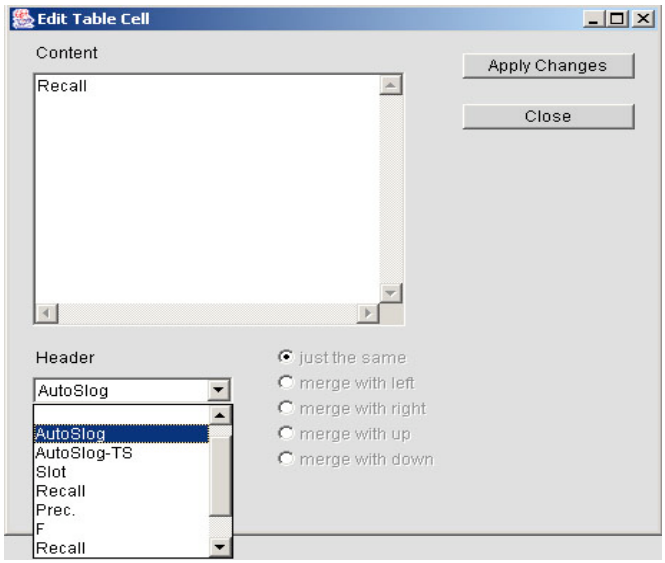

Figure 27: Interface for changing the properties of a header cell with an active pop-up menu

Now the user selects the entry that is empty which refers to the first cell in the table. After applying the changes and closing the cell editing interface the following result is shown in the GUI.

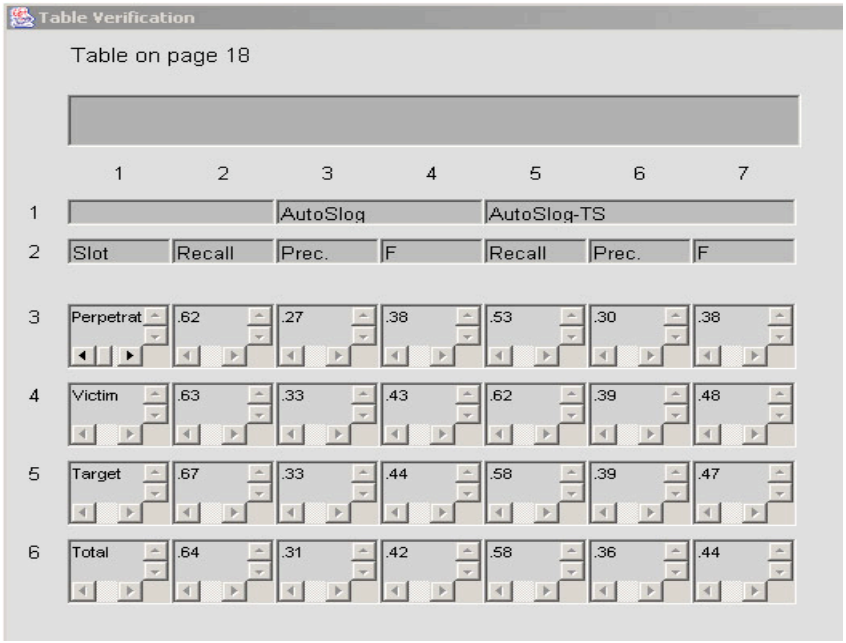

Figure 28: Part of the result after applying the changes in Figure 27

Columns spanning: The user can change the amount of columns the cell is spanning. If the user increases the amount of the column spanning, the cell would only be extended and the other cells originally been on the right side of the cell would be shifted to the right. The program don't verifies the new structure of the table, thus the user must be sure that the table has, in the end, the structure he wants.

The location of the content of the cell: It could happen that some text that was identified by the program as a cell, is not really a cell on its own, but belongs to another neighbour cell. In such cases the user can easily merge the content of this cell to a cell above, below, left or right. Whereas, merging means again attaching the content at the end of the neighbouring cell.

5.2.3 Experimental Results

I will add some source-result pairs which are the results of the testing process.

	AutoSlog			AutoSlog-TS		
Slot	Recall	Prec.	F	Recall	Prec.	F
Perpetrator	.62	.27	.38	.53	.30	.38
Victim	.63	.33	.43	.62	.39	.48
Target	.67	.33	.44	.58	.39	.47
Total	.64	.31	.42	.58	.36	.44

Figure 29: Example for a table with spanning columns

	AutoSlog			AutoSlog-TS		
Slot	**Recall**	**Prec.**	**F**	**Recall**	**Prec.**	**F**
Perpetrator	.62	.27	.38	.53	.30	.38
Victim	.63	.33	.43	.62	.39	.48
Target	.67	.33	.44	.58	.39	.47
Total	.64	.31	.42	.58	.36	.44

Figure 30: Output of my implementation for the table in Figure 29

Table 2

Natural History of Conjunctivitis

Type of Conjunctivitis	Natural History	Potential Complications
Allergic		
Seasonal	Recurrent.	Minimal, local.
Vernal	Onset in childhood, with chronic course with acute exacerbations. Gradual decrease in activity within 2-30 years.	Eyelid thickening, ptosis, keratinization, conjunctival scarring, neovascularization, thinning, infection, ulceration.
Atopic	Onset in childhood, chronic course with acute exacerbations. Gradual resolution in 40% of those afflicted.[4]	Eyelid thickening or tightening, loss of lashes; conjunctival scarring; corneal scarring, neovascularization, thinning, infection, ulceration.
Ocular cicatricial pemphigoid (OCP)	Onset at mean age of 70 years. Slowly progressive chronic course with "spontaneous" remissions and exacerbations.	Conjunctival scarring and shrinkage; severe tear deficiency; corneal scarring, neovascularization, ulceration; bacterial conjunctivitis; cicatricial lid changes; trichiasis.
Mechanical/ Irritative		
Giant papillary (GPC)	Chronic gradual increase in symptoms and signs with contact lens wear.	Corneal neovascularization, scarring.
Viral		
Adenoviral	Self-limited with resolution of symptoms and signs within 5-7 days. Persistence of symptoms and signs for weeks to months is rare.	Mild cases: none. Severe cases: superficial keratitis, including epithelial erosions and subepithelial infiltrates, membranous conjunctivitis, conjunctival scarring.
Herpes simplex (HSV)	Usually subsides without treatment within 4-7 days unless complications occur.	Epithelial keratitis, stromal keratitis, conjunctival scarring.

Figure 31: Example for a table with a simple structure but with much text

Table 2 Natural History of Conjunctivitis

Type of Conjunctivitis	Natural History	Potential Complications
Allergic		
Seasonal	Recurrent.	Minimal, local.
Vernal	Onset in childhood, with chronic course with acute exacerbations. Gradual decrease in activity within 2-30 years.	Eyelid thickening, ptosis, keratinization, conjunctival scarring, neovascularization, thinning, infection, ulceration.
Atopic	Onset in childhood, chronic course with acute exacerbations. Gradual resolution in 40% of those afflicted.	Eyelid thickening or tightening, loss of lashes; conjunctival scarring; corneal scarring, neovascularization, thinning, infection, ulceration.
Ocular cicatricial pempligoid (OCP)	Onset at mean age of 70 years. Slowly progressive chronic course with "spontaneous" remissions and exacerbations.	Conjunctival scarring and shrinkage; severe tear deficiency; corneal scarring, neovascularization, ulceration; bacterial conjunctivitis; cicatricial lid changes; trichiasis.
Mechanical/ Irritative		
Giant papillary (GPC)	Chronic gradual increase in symptoms and signs with contact lens wear.	Corneal neovascularization, scarring.
Viral		
Adenoviral	Self-limited with resolution of symptoms and signs within 5-7 days. Persistence of symptoms and signs for weeks to months is rare.	Mild cases: none. Severe cases: superficial keratitis, including epithelial erosions and subepithelial infiltrates, membranous conjunctivitis, conjunctival scarring.
Herpes simplex (HSV)	Usually subsides without treatment within 4-7 days unless complications occur.	Epithelial keratitis, stromal keratitis, conjunctival scarring.

Figure 32: Output of my implementation for the table in Figure 31

Table 4

Systemic Antibiotic Therapy

Cause	Drug of Choice	Dosage
Adults		
Gonococcus*	Ceftriaxone	1g IM, single dose Lavage infected eyes
Chlamydia	Azithromycin or Doxycycline	1g orally single dose 100 mg orally twice a day for 7 days
Children†		
Gonococcus		
Children who weigh < 45 kg	Ceftriaxone	125 mg IM, single dose
Children who weigh > 45 kg		Same treatment as adults
Chlamydia		
Children who weigh < 45 kg	Erythromycin base	50 mg/kg/day orally in 4 divided doses for 10-14 days
Children under 8 years old who weigh > 45 kg	Azithromycin	1 gm orally, single dose
Children 8 years old or older	Azithromycin or Doxycycline	1 gm orally, single dose 100 mg orally, twice daily for 7 days
Neonates		
Ophthalmia neonatorum Caused by *N. gonorrhoeae*	Ceftriaxone	25-50 mg/kg IV or IM, single dose, not to exceed 125 mg
Chlamydia	Erythromycin	50 mg/kg/day orally divided into four doses daily for 10-14 days

Figure 33: Example for a table with a more complex structure

Cause	Drug of Choice	Dosage
Adults		
Gonococcus*	Ceftriaxone	1g IM, single dose Lavage infected eyes
Chlamydia		
	or	
	Doxycycline	100 mg orally twice a day for 7 days
Children		
Gonococcus		
Children who weigh < 45 kg	Ceftriaxone	125 mg IM, single dose
Children who weigh > 45 kg		Same treatment as adults
Chlamydia		
Children who weigh < 45 kg	Erythromycin base	50 mg/kg/day orally in 4 divided doses for 10-14 days
Children under 8 years old who weigh > 45 kg	Azithromycin	1 gm orally, single dose
Children 8 years old or older	Azithromycin	1 gm orally, single dose
	Doxycycline	100 mg orally, twice daily for 7 days
Neonates		
Ophthalmia neonatorum	Ceftriaxone	25-50 mg/kg IV or IM, single dose, not to exceed 125 mg
Caused by N. gonorrhoeae		
Chlamydia		

Figure 34: Output of my implementation for the table in Figure 33 without any post-processing

Cause	Drug of Choice	Dosage
Adults		
Gonococcus*	Ceftriaxone	1g IM, single dose Lavage infected eyes
Chlamydia	Azithromycin	1g orally single dose
	or	
	Doxycycline	100 mg orally twice a day for 7 days
Children		
Gonococcus		
Children who weigh < 45 kg	Ceftriaxone	125 mg IM, single dose
Children who weigh > 45 kg		Same treatment as adults
Chlamydia		
Children who weigh < 45 kg	Erythromycin base	50 mg/kg/day orally in 4 divided doses for 10-14 days
Children under 8 years old who weigh > 45 kg	Azithromycin	1 gm orally, single dose
Children 8 years old or older	Azithromycin	1 gm orally, single dose
	or	
	Doxycycline	100 mg orally, twice daily for 7 days
Neonates		
Ophthalmia neonatorum Caused by N. gonorrhoeae	Ceftriaxone	25-50 mg/kg IV or IM, single dose, not to exceed 125 mg
Chlamydia	Erythromycin	50 mg/kg/day orally divided into four doses daily for 10-14 days

Figure 35: Output of my implementation for the table in Figure 33 after post-processing with the GUI

Bibliography

Alexiev W. *Data Integration Survey*. Sirma AI Corp, 2004.

Angeles M.del P, MacKinnon L.M. *Solving Data Inconsistencies and Data Integration with a Data Quality Manager*. Technical Report, Doctoral Consortium at BNCOD, 2004.

Bagga A. *A Short Course on Information Extraction: A Proposal*. Department of Computer Science, Duke University, Durham, 1998.

Belkin N, Croft, B. *Information Filtering and Information Retrieval: Two Sides of the Same Coin?*. Communications of the ACM, 35(12):29-38, 1992.

Bienz T, Cohn R. *Portable Document Reference Manual / Adobe Systems Incorporated* (second printing). Reading, England: Addison-Wesley, 1996.

Borgida A. *Information Integration*. 2003.

Breu M, Ding Y. *Modelling the World: Databases and Ontologies*. Whitepaper by IFI, Institute of Computer Science, University of Innsbruck, 2004.

Brujn J.d. *Semantic Information Integration Inside and Across Organizational Boundaries*. Master thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Netherlands. Digital Enterprise Research Institute, October 2003.

Cardie C. *Empirical Methods in Information Extraction*. Al Magazine, 18(4):65-80, 1997.

Chidlovskii B. *Automatic Repairing of Web Wrappers*. Proceeding of the Third International Workshop on Web Information and Data Management, p.24-30, New York, NY: ACM Press, 2001.

Chinchor N.A. *Overview of MUC7/MET-2*. In Proceedings of the Seventh Message Understanding Conference (MUC7), San Diego, GA, 1998.

Ciravegna F. *Challenges in Information Extraction from Text for Knowledge Management*. IEEE Intelligent Systems and Their Applications, 16(6):88-90, 2001.

Cowie J, Lehnert W. *Information Extraction*. Communications of the ACM, 39 (1):80-91, 1996.

Del Bimbo A. *A Perspective View on Visual Information Retrieval Systems*. Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libriaries. Washington, DC: IEEE Computer Society, 1998.

Gaizauskas R, Wilks Y. *Information Extraction: Beyond Document Retrieval*. Journal of Documentation, 54(1):70-105, 1998.

Gartner H.J. *Extraction of Semantic Information from Layout-Oriented Data*. Master's Thesis, Graz University of Technology, Graz, Austria, October 2003.

Grishman R. *Information Extraction: Techniques and Challenges*. In Maria Teresa Pazienza (Ed.), Information Extraction. Lecture Notes in Artificial Intelligence. München, Germany: Springer Verlag, 1997.

Gross M, *Converting From PDF To XML & MS Word: Avoiding The Pitfalls*. Whitepaper, 2003.

Halevy A, Li C. *Information Integration Research: Summary of NSF IDM Workshop Breakout Session*. Seattle, Washington, 2004.

Harman D. *Overview of the First Text REtrieval Conference (TREC-1*). National Institute of Standards and Technology, Gaithersburg, Maryland, 1993.

Ingwersen P. *Information Retrieval Interaction*. London, UK: Taylor Graham Publishing, 1992.

Jhingran A.D, Mattos N, Pirahesh H. *Information Integration: A Research Agenda*. IBM Systems Journal, 41 (4), 2002.

Kauchak D, Smarr J, Elkan C. *Sources of Success for Information Extraction Methods*. Technical Report. Dept. of Computer Science, University of San Diego, 2002.

Klinkenberg R, Renz I. *Adaptive Information Filtering: Learning in the Presence of Concept Drifts*. In learning Text Categorization, p: 33-40. Menlo Park, California: AAAI Press, 1998.

Kushmerick N, Thomas B. *Adaptive Information Extraction: Core Technologies for Information Agents*. In Intelligent Information Agents R&D in Europe: An AgentLink perspective, 2002.

Lenzerini M. *Data integration: A Theoretical Perspective*. Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, New York, NY, USA: ACM Press, 2002.

Leymann F, Roller D. *Using Flows in Information Integration*. IBM Systems Journal, 41 (4): 732-742, 2002.

Marsh E, Perzanowski D. *MUC-7 Evaluation of IE Technology: Overview of Results*. 1998.

Merz, T. *Web Publishing with Acrobat/PDF*. München, Germany: Springer Verlag, 1998.

Mresse, M. *Information Retrieval – Eine Einführung*. Stuttgart, Germany: Teubner, 1984.

Nohr, H. *Grundlagen der Automatischen Indexierung: Ein Lehrbuch*. Berlin, Germany: Logos Verlag, 2003.

Olsson T. *Information Filtering with Collaborative Interface Agents*. Department of Computer and Systems Sciences, The Royal Institute of Technology, 1998.

Pinto D, McCallum A, Wei X, Croft B.W. *Table Extraction Using Conditional Random Fields*. SIGIR'03 Toronto, Canada, p.235-242. New York, NY, USA: ACM Press., 2003.

Pollitt A.S. *Information Storage and Retrieval Systems: Origin, Development and Applications*. Chichester: Horwood, 1989.

Ramel J.-Y, Crucianu M, Vincent N, Faure C. *Detection, Extraction and Representation of Tables*. Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03), 2003.

Ray E.T. *Learning XML*. O'Reilly & Associates, Inc, 2001.

Rijsbergen C.J. van. *Information Retrieval*. London: Butterworths, 1979.

Riloff E. Information *Extraction as a Stepping Stone toward Story Understanding*. Department of Computer Science, University of Utah, Salt Lake City, 1999.

Roy J, Ramanujan A. *XML:Data's Universal Language*. IT Pro May 1 June 2000, p.32-36, 2000.

Salton G, McGill M.J. *Introduction to Modern Information Retrieval*. New York, NY: McGraw-Hill, 1983.

Tupaj S, Shi Z, Dr.Chang C.H, Alam H. *Extracting Tabular Information From Text Files*. 1996. EECS, Tufts University, Medford, USA, 1996.

Voorhees E.M. *Overview of TREC 2003*. National Institute of Standards and Technology. Gaithersburg, Maryland, 2003.

Wang Y. *Document Analysis: Table Structure Understanding Zone Content Classification*. Doctoral dissertation, Washington University, 2002.

Weisz L. *Adobe PDF Conversion: How, For Whom, And When?*. Whitepaper, Data Conversion Laboratory, www.dclab.com/pdf-conversion.pdf

Yang J, Kim J, Doh K.-G, Choi J. *Wrapper Generation by Using XML-Based Domain Knowledge for Intelligent Information Extraction*. Proceedings of IJCAI-2001 Workshop on Adaptive Text Extraction and Mining, 2001.

Nimble. *Next-Generation Data Integration: Harnessing Data for Business Advantage*. Whitepaper, Nimble Technology, Inc. www.nimble.com.

Zisman A. An *Overview of XML*. Computing & Control Engineering Journal, p.165-167, 2000.