

# **Table Understanding for Information Retrieval**

**Ashwini Pande**

Thesis submitted to the faculty of  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

**Master of Science**

in

Computer Science and Applications

**Dr. Roger Ehrich, Chair**

**Dr. Edward A Fox**

**Dr. Christopher North**

**August 19<sup>th</sup>, 2002**

**Blacksburg, VA, USA**

**Keywords**

**Information retrieval, Table detection, Statistical crosscorrelation, detection  
heuristics, Odessa digital library**

# **Table Understanding for Information**

## **Retrieval**

**Ashwini Pande**

### **Abstract**

This thesis proposes a novel approach for finding tables in text files containing a mixture of unstructured and structured text. Tables may be arbitrarily complex because the data in the tables may themselves be tables and because the grouping of data elements displayed in a table may be very complex. Although investigators have proposed competence models to explain the structure of tables, there are no computationally feasible performance models for detecting and parsing general structures in real data. Our emphasis is placed on the investigation of a new statistical procedure for detecting basic tables in plain text documents. The main task here is defining and testing this theory in the context of the Odessa Digital Library.

# Acknowledgements

I would like to express my gratitude to my advisor, Dr. Roger Ehrich, for his support, advice and encouragement. Without his tireless reading and constructive criticism, this work would not have been possible.

I would also like to thank my thesis committee, Dr. Edward Fox and Dr. Christopher North for their contribution to the improvement of this thesis.

Finally, I am grateful to the Department of Computer Science, Virginia Tech, for providing

# Table of contents

Abstract.....	ii
Acknowledgements.....	iii
Table of contents.....	iv
Table of figures.....	v
<b>Introduction.....</b>	<b>1</b>
1. Introduction.....	1
1.1 The Odessa digital library.....	4
1.2 The Odessa system architecture.....	5
1.3 High-level description of the Odessa system.....	7
<b>Research Objectives.....</b>	<b>9</b>
2. Research Objectives.....	9
2.1 Problem.....	9
2.2 Goals.....	10
<b>Review of Previous Work.....</b>	<b>11</b>
3. Review of Previous Work.....	11
<b>Statistical Crosscorrelation for Table Detection.....</b>	<b>15</b>
4. Statistical Crosscorrelation for Table Detection.....	15
4.1 What is crosscorrelation?.....	15
4.1.1 String homomorphism.....	16
4.1.2 Sequential transducer mapping.....	17
4.1.3 Context maps.....	17
4.2 Forward and backward aggregates.....	18
4.2.1 Computing the forward aggregate.....	18
4.2.2 Computing the backward aggregate.....	19
4.3 The forward and backward crosscorrelations.....	20
4.4 The heuristics of line mappings.....	21
4.5 Crosscorrelation results.....	22
<b>Table Detection Heuristics.....</b>	<b>25</b>
5. Table Detection Heuristics.....	25
5.1 Table body detection heuristics.....	26
5.2 Table header detection heuristics.....	27
<b>Integration with the Odessa Digital Library.....</b>	<b>32</b>
6. Integration with the Odessa Digital Library.....	32
<b>Experimental Results.....</b>	<b>35</b>
7. Experimental Results.....	35
<b>Conclusions.....</b>	<b>41</b>
8. Conclusions.....	41
<b>Appendix A - Character maps.....</b>	<b>43</b>
<b>Appendix B - Pseudo-code for table detection heuristics.....</b>	<b>45</b>
<b>Appendix C - Pseudo-code for table header detection heuristics.....</b>	<b>47</b>
<b>References.....</b>	<b>49</b>
<b>Vita.....</b>	<b>50</b>

# Table of figures

Figure 1: Typical table layouts .....	2
Figure 2 : One-dimensional vertical table.....	3
Figure 3 : The Odessa system architecture diagram .....	6
Figure 4 : User interface for the Odessa system .....	8
Figure 5 : Search result using the Odessa system .....	8
Figure 6 : Crosscorrelation in free text .....	23
Figure 7 : Crosscorrelation result for a table followed by free text.....	24
Figure 8 : Crosscorrelation results for an irregular table .....	24
Figure 9 : Table detection using crosscorrelation values.....	26
Figure 10 : File with a single line header separated from table body .....	28
Figure 11 : Table with header attached to the body but with a special line separator .....	29
Figure 12 : A single line table header separated from the table body.....	29
Figure 13 : Table with a multi-line header.....	30
Figure 14 : Header above the detected SOT separated by a special line .....	30
Figure 15 : Header above the detected SOT .....	31
Figure 16 : Architecture diagram for the new system.....	34
Figure 17 : Table with irregularities .....	36
Figure 18 : Search result when table is misidentified .....	36
Figure 19 : Aligned text, misidentified as a table .....	37
Figure 20 : File containing multiple tables .....	38
Figure 21 : Search result with table headers .....	39
Figure 22 : Crosscorrelation results with numerical data .....	40

# Introduction

## 1. Introduction

Tables play a very important role in presenting information. They not only store information in a manner that makes it understandable but they also present information in a simplified and compact way. It may be easy to point out a table in a book, but a precise definition of a table is elusive. The Oxford English Dictionary defines a table as: “An arrangement of numbers, words or items of any kind, in a definite and compact form, so as to exhibit some set of facts and relations in a distinct and comprehensive way, for convenience of study, reference or calculation” [10]. The main function of a table is to present detailed information in a compact way such that the ability to search and compare the information is enhanced.

There are a number of possible representations for a tabular structure. The photocomposition of tables can be one of the most challenging aspects of document typesetting. A table may contain different kinds of objects such as text, graphics and mathematical formulas. From a logical point of view tables are multidimensional objects, but many of the most important ones are one-dimensional. The dimensionality of the table refers to the degree to which tables are recursively nested within tables rather than to the number of geometric dimensions. Although a table with a simple row and column structure is a popular type of table, tables can be much more complex. We call this visual presentation, the “layout” of a table. Figure 1 illustrates various table layouts. Depending upon the layout of the table, it might be classified as a partitioned table (Figure 1a), an over-expanded table (Figure 1b), or a basic table of the type of primary interest in this research (Figure 1c) [10].

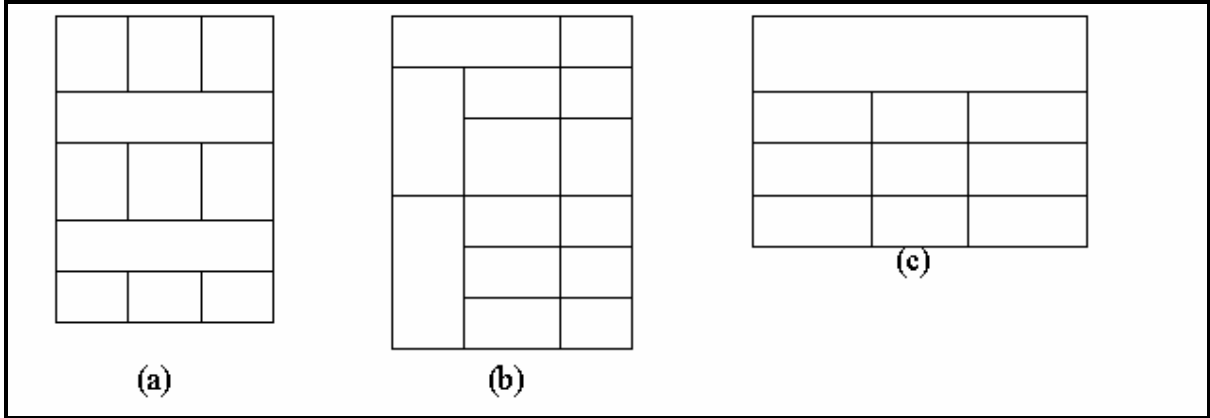


Figure 1: Typical table layouts

The structural complexity of a table depends directly on the layout of the table. Detection of the tables presented in layouts 1a and 1b would be difficult compared to the layout in 1c. We call tables represented by layouts 1a and 1b multidimensional tables. The table represented by layout 1c can be considered a one-dimensional table. Since tables have these different presentation styles it becomes necessary to extract the presentation style from real textual information and exploit it to extract the actual table data. Due to the arbitrarily complex and often irregular nature of tables, for the purpose of this thesis we decided to concentrate on well-formed one-dimensional vertical tables.

A table is one-dimensional if all of the table elements are atomic, that is, if none of them are themselves tables. A table is vertical if it has only a column header at top. A one-dimensional vertical table consists of a header and a body. Its structure is determined by the number of columns in the table and the parts belonging to the header and the body. There might be separators between the header and body or between columns that are not part of the table but which give visual clues about its structure. Figure 2 shows an example of a one-dimensional vertical table.

Child Name	Birth Date	Father	Mother	Comments
Baierl, Magdalena	15 Aug 1883	Carl	Sommerfeldt, Eva	Kischinew record
Brickmann, Emilie	22 Sep 1883	Friedrich	Oberlander, Marie	Kischinew record
Denning, Christian	20 May 1883	Carl	Wallenwein, Friedricke	Kischinew record
Gunsch, Emilie	22 May 1883	Johann	, Christine	Kischinew record;
Gunsch, Magdalena	22 May 1883	Johann	, Christine	Kischinew record;
Gunsch, Sophie	16 Feb 1883	Friedrich	Friedrich, Christine	Kischinew record
Hamann, Catharine	6 Apr 1883	Gottlieb	Jesske, Julia	Kischinew record
Kreuger, Wilhelm	22 Jul 1883	Ferdinand	, Christine	Kischinew record
Moritz, Louise	23 Jan 1883	Samuel	Christmann, Caroline	Kischinew record
Neumann, Jacobine	18 Aug 1883	Heinrich	Walz, Rosine	Kischinew record
Rehmann, Johannes	25 Dec 1883	Samuel	Berg, Julianna	Kischinew record
Rittmueller, Elisabeth	7 Jun 1883	August	Rehmann, Juliane	Kischinew record
Saas, Friedrike	22 Sep 1883	Johann	Bonert, Dorothea	Kischinew record
Strebel, Heinrich	15 Aug 1883	Wilhelm	Quale?, Christine	Kischinew record
Walz, Wilhelm	21 Feb 1883	Johann	Walker, Catharina	Kischinew record

Figure 2 : One-dimensional vertical table

Unlike HTML documents, plain text documents may not contain any easily identifiable structural clues about the presence of tables or layout. In HTML documents with tables contain not only the content but also an encoding of their structure. This encoding of tabular structure makes it easy to parse HTML documents. Plain text documents contain no such structural encoding. Presence of structure in plain text documents can be detected only with the help of visual clues unless natural language processing is employed to extract the semantics of the table contents. Now that most HTML pages are composed using authoring tools, HTML documents have a regular format, and the chances of having malformed syntax are small. On the other hand, many plain text documents are typed by humans. They may be visually irregular and may contain formatting elements. Since plain text documents have no embedded syntax, we must rely on the visual layout for identifying tabular structure. Using those clues we can extract the table as an entity, dissect the extracted table entity to understand its structure, and finally use the obtained formatting information to extract data from the table (such as a particular column).

For the purpose of this thesis, experiments were done with the Odessa Digital Library. The documents in this library are mostly plain text documents containing a mixture of tabular information (structured text) and unstructured or free text. This thesis proposes a novel approach for extracting structural information at document indexing time so that stored table

information can be used at retrieval time to improve the display of retrieved information. For example, if a hit at search time happens to be in a table row, then returning the table header along with the actual table information can help users better understand the meaning of the returned results.

## 1.1 The Odessa digital library

An effort has been underway since 1993 to build a digital library of large plain text documents such as books and tables. The purpose of Odessa is to provide a research environment with a browsable and searchable document set and a user base from which feedback about the user interface can be obtained easily. As of August 2002 the library held over 1000 documents averaging 250 KB each. One of the main features of Odessa is that it is designed to conserve network bandwidth and user search effort by returning actual information in context rather than document hyperlinks. Most digital libraries return hyperlinks to returned documents. This requires the end user to download each document to perform a secondary search. Instead, Odessa's goal is to return each hit with sufficient context to enable the end user to decide on relevance without downloading and searching individual documents.

The existing system is an indexed retrieval system that searches for keywords at document-level granularity. The query language at present consists only of keywords with wildcard suffixes. Odessa relies heavily on a document dictionary that keeps track of document locations and attributes such as revision date, search category, title, and document style. The document style specifies how returned information is to be formatted. For example, hits in free text documents might be returned with n lines of context, while table rows might be returned without any context. The problem with this approach is that tabular data is returned without any table header information, and that the screen information display format for mixed documents containing both free text and tables does not adapt to changes in the document structure.

## 1.2 The Odessa system architecture

The Odessa system consists of different types of files such as data files and program files. They are listed below [1].

System data files:

- bindocs.txt: Document information such as revision date, format and document title.
- pathfile.txt: The physical location of each document.
- urlfile.txt: The URL for each document.
- catfile.txt: The names of the search categories associated with each document.
- stop.txt: A sorted list of stop words.
- lexicon: The lexicon generated by BuildIndex.cpp
- docs.def: Document definition file, generates lexicon, pathfile.txt, catfile.txt, urlfile.txt and bindocs.txt.

System programs:

- BuildIndex: The indexing program.
- isearch: A program that analyzes a query, searches the lexicon and index, creates a list of documents containing the query, and performs sequential searches on those documents.
- binindex: Creates the memory files from the document dictionary.
- unload.sh: The shell script that deletes the data files from memory.

Since this is a relatively small retrieval system, Odessa has no document updating capabilities. When changes are made to files, the entire document set is reindexed, which requires 8 minutes per GB on a 500Mhz DEC Alpha. The indexing process requires that the search program be disabled and shared memory files deleted before indexing.

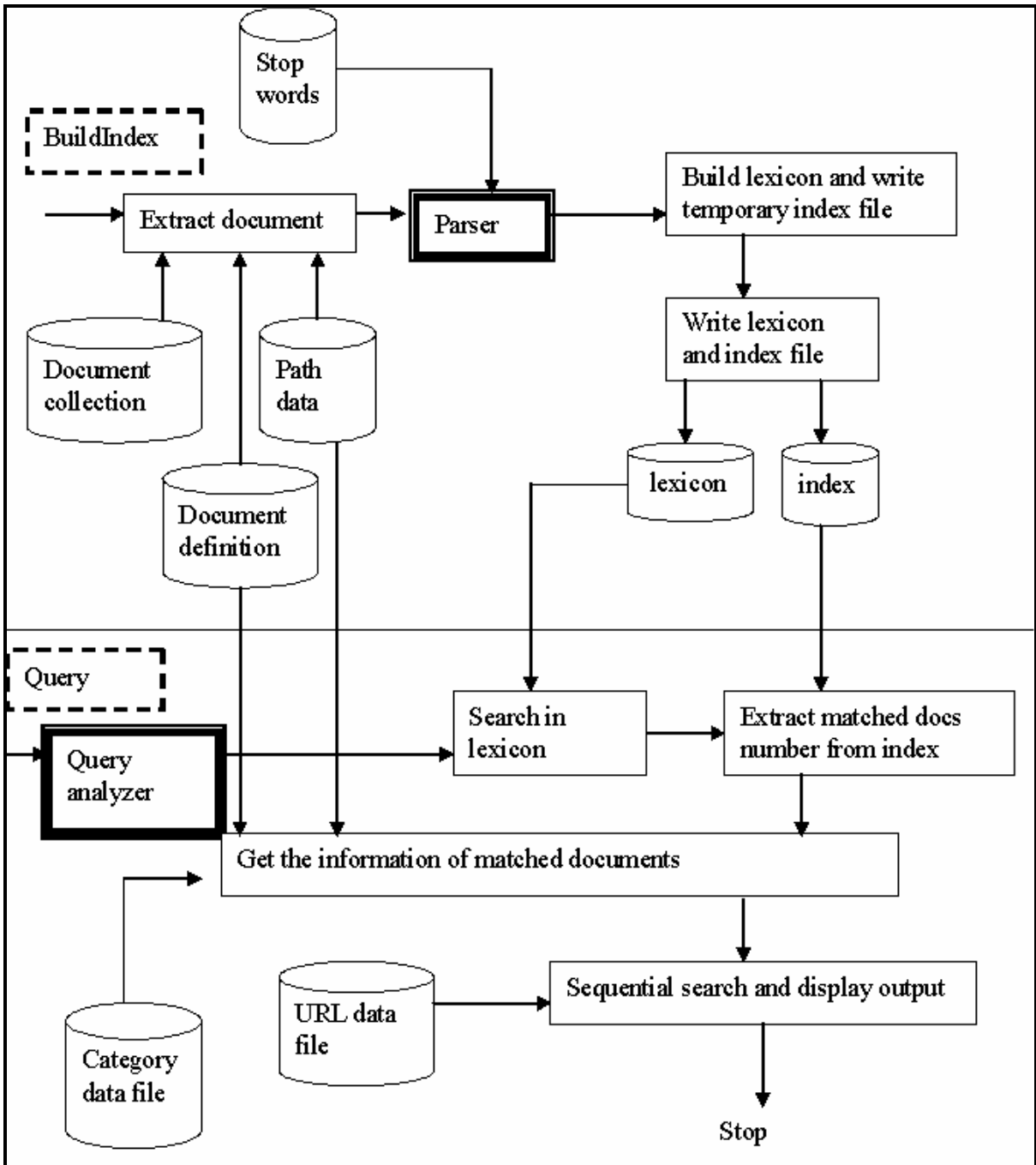


Figure 3 : The Odessa system architecture diagram

Key:

: data files    
  : process    
  : program    
  : command

## 1.3 High-level description of the Odessa system

### Indexing:

1. For every document in the system, parse the document and store the terms (keywords) of the document. A trie-based algorithm removes stop words [6].
2. Count the number of occurrences of each term, and generate a temporary index file.
3. Sort the temporary index file to obtain the final index file.
4. Sort the lexicon.

### Searching:

1. Enter the keyword-based query. The query system supports wild card characters.
2. Perform a binary search for the keyword in the lexicon. For keywords with wild cards perform a binary search on the prefix in the lexicon.
3. For every matched term, locate the term in the index file and retrieve the corresponding document information.
4. Search each located document sequentially for the keyword.
5. Output lines containing the search term according to the formatting specified in the document dictionary.

Figure 4 shows the system interface by means of which the user can enter a query. Figure 5 shows the search results.

## Full Text Indexed Search

Full text retrieval systems are essential for locating information in a large document collection. The Odessa library encourages the use of full text retrieval systems on home computers, and broad support is planned to provide users with assistance in locating relevant documents to download. Until then, we are providing some general retrieval capability to demonstrate the power of full text retrieval in several of the content areas of the Odessa collection.

The following text box can be used to enter a single word or word prefix that will be matched against words of documents in the respective data categories. The search is case independent and will produce a hyperlinked document containing all search hits. **Descriptions of the files will be returned only if they contain matches.** These descriptions are hyperlinks to the actual source documents. The search results document may be stored and referenced locally in your home computer by your web browser.

A search term must begin with a letter, but subsequent characters may be letters, numerals, or underscores. Search terms must be single words or word prefixes; phrases are not searchable. By default, only exact matches are located in the text, but % and \* wildcard matches are permitted **at the end** of query terms. % matches any single character, and \* matches a string of any length. Here are some examples:

xyz matches all occurrences of xyz

xyz\* matches all terms of all lengths that begin with xyz

xyz%%%% matches all 6-letter terms beginning with xyz

xyz%%%\* matches all terms of at least 5 letters that begin with xyz

But xy\*z, x%\*yz, and xy%\*z would not interpret % or \* as wildcards because they don't appear at the end

Search String:

Data Category:

Figure 4 : User interface for the Odessa system

first principal of Fisk School, now Fisk University, and Mrs. Ogden taught the primary classes. She also discovered and gave the first training to the

**File: [Akaska SD, Golden Anniversary, 1907-1957 . . . . .](#)**

- Occurrence #56, Line 958

Emanuel Eiteneier was born January 25, 1877 at New Nassau, a small village in South Russia. There he grew to young manhood, working for his father on the farm. At the age of 21 he was drafted into the Russian Army and served for 4 years. After receiving his discharge he was employed by the water department at Odessa. While there he met and married Maria Wacker on January 25, 1904. At

**File: [Original Homeland & Emigration of Germans to Bessarabia. . .](#)**

- Occurrence #57, Line 554

Bihlmeyer	Johann Chr(istian)	Baach/Waiblingen/Wuerttemberg	Gnadental
Bieske	Gottlieb	Germany	Tarutino
Bietz	Adam	Gnesen/Prussia	Alt Posttal
Bietz	Friedrich	Maschelwitz(?) Prussia	Alt Posttal
Bietz	Phillip	1799 Katzelnbogen/Hessen-Nassau	Kloestitz
Billigsmailer	Jakob	1792 Graefenhhausen/Rhine Hessen	Leipzig
Binder	Gottlieb	1773 Stawiszyn/Poland	Kulm
Binder	Elisabeth	1800 Stawiszyn/Poland	Kulm
Bindevald	Michael	1798 Hasloch/Palatinate/Bavaria	Borodina

- Occurrence #58, Line 1068

Frey	Jakob	Baihingen/Nagold/Wuerttemberg	Teplitz
Frey	Johann	1798 Wuerttemberg	Alt Elft
Frey	Johann Peter	1805 Bnin today Poland	Kloestitz
Frey	Susanna (Mrs)	1814 Poland	Alt Elft
Frey	Philipp	1799 Katzelnbogen/Hessen/Nassau	Kloestitz
Freyer	Gottfried	Poland	Beresina
Frick	Christian	Althausen/Mergentheim/Wuerttemberg	Gnadental

Figure 5 : Search result using the Odessa system

# Research Objectives

## 2. Research Objectives

Since Odessa is a network-based retrieval system it does not have an integrated file browser to enable an end user to examine search results. Instead search results are presented in a context sufficient to enable the searcher to evaluate the utility of the returned information. The open problem to be addressed is that of determining for each retrieved search result a context that will facilitate the end-user's evaluation of the utility of that result.

The current approach is a manual approach in which the library administrator reviews each document and specifies in the document dictionary an output format appropriate to that document. For example, if a document consists of a collection of small self-contained paragraphs, it might be reasonable to return the entire paragraph containing a search hit with the query term highlighted. The two drawbacks to this procedure are 1) that the process is manual and 2) that there is no good way to handle a heterogeneous document containing a mixture of different textual structures, for example, free text and tables.

Therefore the objective of this research is to explore ways to examine document structure dynamically and adjust the contextual information display to the end user automatically.

### 2.1 Problem

After considerable experience with the Odessa system it became clear that the two most important types of documents were free text and tables. Apparently free text can be handled reasonably well by returning each search result with  $n$  lines of context preceding and following the search result, possibly bounding the context within the current paragraph. Tables, however, need to be treated completely differently. Individual table records are independent and require no context. However, table records may make little sense without the information in the corresponding table headers.

Since there is no reasonable way to locate tables and their headers in documents manually, the challenge is to develop procedures for locating table structures automatically. Once these have been located at indexing time, this information must be stored and accessed at search time to produce an appropriate tabular information display. The only clues about the location of tabular structure or textual information are in the spatial alignment of information fields plus certain anomalies that signal the presence of a table header and the end of a table. The problem then becomes one of identifying computational algorithms and logic to accomplish this goal.

## 2.2 Goals

- Develop a methodology for efficient table detection.
- Develop heuristics to extract that structure from free text.
- Use the table structural information in the retrieval process so that tabular data can be better presented in the search results.
- Table extraction without user intervention and without any prior knowledge of document syntax.

# Review of Previous Work

## 3. Review of Previous Work

Several investigations related to tabular structure in plain text documents have been reported in the literature. Some of the efforts are related only to locating tables in text while others are related to exploiting tabular data for the purpose of search and retrieval.

William Kornfield and John Wattecamps's work, "Automatically Locating, Extracting and Analyzing Tabular Data" [11], describes a method to infer semi-automatically the structure of financial statements/documents. Their system parses the financial documents and extracts the information to form a hierarchical structure. Several types of data structures are generated automatically as the documents are parsed. These data structures are then used for template generation. These templates are normalized financial statements, which are used to map data directly from one table to another across financial institutions. After the tables are detected, they are parsed using an LR (k) parser to generate a parse tree. Since this approach focuses on financial tables, it is easy to generate tokens for the parser because the language used in the financial world is very standard. This approach is very restrictive in the sense that it cannot be applied to tables surrounded by free text. Also the application area for this method is specific to the financial domain. The strength of our approach is that it is not restricted to tables having data from any specific domain and we can successfully identify and extract tables surrounded by free text.

Daniela Rus and Devika Subramanian have made another significant effort in this field. The key idea of their paper, "Customizing Information Capture and Access" [3], is to exploit the underlying structure of an electronic document at various levels of granularity to build the high-level indices with task-specific interpretations. If the documents contain data in the form of tables or graphs then the information access process needs to rely on the structural clues to access information at that level of granularity. Hence for efficient information access, structure detectors are needed that can identify a part of document as table or a section/paragraph. The detection and extraction process is divided into two phases. In the

first phase, the segmentation algorithm breaks the document into pieces at the paragraph level. The second phase is the table detector phase, which analyzes each piece of the document for the presence of a table. They use an image processing approach in which the documents are first scanned and then analyzed pixel-by-pixel which is necessary for analyzing documents using proportional fonts. The segmentation algorithms in this approach rely heavily on the width of white space between the logical units of document like paragraphs and sections. A graph called the white space density graph (WDG) is then generated for each piece of the document. This WDG is generated for every section/paragraph of the document and then analyzed for the presence of a table. The correctness of the segmentation algorithm depends upon the specified value for the width of the white space and on the regularities in the “environment” of the document. Since this approach is an image processing-based approach, they can make use of font properties such as size and width to obtain clues about the underlying structure. The basic distinction between this approach and our approach is that we treat an entire document as one entity, and we do not rely on any typesetting rules for the documents. Their work does not distinguish between various table components (such as the table body or the table header) and lacks the ability to do structured queries on fields of a table.

Shona Douglas, Matthew Hurst and David Quinn’s work resembles the WDG methodology. In their paper, “Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text” [9], they have proposed a new approach for interpreting tables. They call it “Functional View of Tables”. They relate the tabular information to a relational database where the underlying representation of a table is a set of  $n$ -tuples, where  $n$  is the number of domains or value sets in the table. They call this  $n$ -tuple presentation the “canonical form” of a table. This approach emphasizes the layout characteristics of a table. Some of the key steps in the entire process are detecting segments of contiguous spaces, deciding similarity of areas, and detecting columns. This approach matches the text against table prototypes. This work concentrates more on interpreting the detected tabular structure whereas our approach emphasizes detecting the tabular structure.

NoDoSE, the “Northwestern Document Structure Extractor” [7] is an interactive tool for determining semi-automatically the structure of, and extracting the data from a text document. With the help of a graphical user interface, the user can decompose a text file into a hierarchical structure and outline his areas of interest. He can describe their semantics. This task is expedited by a mining component that attempts to infer the grammar of the file using the information that the user has provided so far. Once the format of the document has been determined, its data can be extracted into a number of useful forms. The entire process follows a sequence of modeling, decomposing, and mining the text document and finally exporting the extracted text. Since this is an interactive tool, the user can see the document to identify its structure. Our efforts are concentrated on identifying tabular structures without user intervention.

“Semantic Search on Internet Tabular Information Extraction for Answering Queries” [8] by H.L. Wang, C.L. Sung, S.H. Wu, W.L. Hsu, I.C. Wang, and W.K. Shih presents a novel semantic search approach capable of extracting information from general tables. A system of layout syntax and a set of transformation rules are defined to transform tables into databases without losing their semantic meaning. This effort concentrates on tables that are present in web pages on the Internet. The actual process involves creating an intelligent Internet agent that searches the web pages related to the user’s queries. It then identifies all related tables in all the retrieved web pages and extracts all the necessary information from these tables. In the next stage of processing it integrates the information and finally answers the user queries. Since this effort uses web pages we can assume that most of the web pages are in HTML format where table structure is easy to detect using the <table> tags. Even though it is conceptually somewhat similar to our efforts, there are several ways in which it differs. First of all, search in this case is performed for the documents over the Internet whereas in our case documents reside in a local digital library. Their approach integrates data from different tables to answer the user queries, which is not required for our approach. Also, the system uses layout syntax and transformation rules for transforming the tables into databases, which makes it necessary to know the structure of the table in advance.

The automated structuring community [12] has done some work related to bitmapped images. Their work concerns detecting structure in documents for traditional image processing and pattern recognition tasks. Our work focuses on users of an information retrieval system through table detection and exploitation of structure for formatting retrieved digital documents.

The system that most closely resembles ours is a system called TINITN (Table Information-based Text INquiry) [4] developed by Pallavi Pyreddy and W. Bruce Croft. This is a text table retrieval system. The emphasis here is on exploiting the structure of information in a document to identify tables and their component fields and then provide users with options to query based on those fields. This method uses simple heuristics for table detection and extraction. After the components of the table are extracted they are tagged for retrieval purposes. This approach relies heavily on the alignment of white space. The table extractor generates a Character Alignment Graph (CAG). A CAG is formed by checking for white space alignment in blocks of contiguous lines of text. A number of other heuristic parameters such as holes (number of blanks in a line of text) and gaps (number of occurrences of multiple contiguous blanks per line) are used to check for the gap structure in a block of data. The extracted table entries are then tagged to differentiate between a caption and a table entry. This approach requires the end user to have knowledge of the underlying data set for using the query system. The main difference between this approach and ours is that the end user can query without any preknowledge of the structure of the underlying data. A statistical methodology is a very important part of our work that distinguishes it from the TINTIN approach. The component tagging method requires the modification of data, which is avoided in our case. Our approach also emphasizes the alignment of special characters, such as punctuation characters.

It can be seen that table understanding has been of considerable interest in the retrieval community, but to date little has been achieved toward automatic knowledge-independent table extraction. This is the focus of our work.

# Statistical Crosscorrelation for Table Detection

## 4. Statistical Crosscorrelation for Table Detection

In the absence of *a priori* information about the structure of tables we may find in document text, the only possible approach is to look for column alignments among the beginnings and/or ends of words in adjacent lines of text, which is the basic idea underlying a number of previous approaches to table detection.

One basic technique involves column accumulators that tally weighted counts of characters occurring in the text columns spanned by adjacent text lines. Many different character-weighting schemes are possible, and there are numerous ways to manage the accumulators as successive lines of text are scanned. As text lines are processed, the array of column accumulators must be analyzed for column alignments. Two observations are important in the design of the heuristics: 1) character-whitespace and whitespace-character transitions are especially important features and 2) vertical alignment of identical characters are especially strong indicators of tabular alignment, since the probability of identical characters occurring in the same column of adjacent rows of unstructured text is low. Conditions 1) and 2) provide support for tabular alignments but not for the absence of tabular alignments.

After working for some time to find heuristics that would make the column accumulator approach reliable, we began to look for a more straightforward computational approach for determining the similarity between adjacent lines of text. The technique that emerged involved mapping text lines into number sequences and computing statistical crosscorrelations between these mapped lines.

### 4.1 What is crosscorrelation?

Crosscorrelation is a standard statistical signal detection procedure that is useful for determining the similarity of two signals. To calculate the crosscorrelation between two

signals with range  $-n$  to  $n$  and shift  $k$ , formula (1) is used, where  $f(i)$  and  $g(i)$  are real-valued number sequences.

$$C_{fg}(k) = \frac{1}{2n} \sum_{i=-n}^n f(i) g(i+k) \quad (1)$$

In the case where  $f$  and  $g$  has finite length and a shift is not of interest, we may rewrite the above formula as:

$$C_{fg}(0) = \frac{1}{n} \sum_{i=1}^n f(i) g(i) \quad (2)$$

In this thesis we explore the crosscorrelation concept as a way of computing similarity measures between lines of plain text or between aggregates of lines. The key for doing this is a mapping function  $\tau$ .

$$\tau : \Sigma^* \rightarrow Z^* \quad (3)$$

where  $Z^*$  is set of integers and  $\Sigma$  is the text alphabet. The mapping function  $\tau$  maps a character string to  $L_i \in \Sigma^*$  into a number sequence  $\vartheta_i \in Z^*$  that represents its structure.

There are many possible types of mappings,  $\tau$ , and we present several alternatives.

#### 4.1.1 String homomorphism

Let  $\tau : \Sigma^* \rightarrow Z^*$  be a mapping of characters to the signed integers such that

$$\tau(a_1 \dots a_k) = \tau(a_1) \dots \tau(a_k) \quad (4)$$

### 4.1.2 Sequential transducer mapping

Sequential transducers are state machines in which the mapping of a potential character depends upon its left context.

### 4.1.3 Context maps

A third mapping alternative is what may be called a context map in which the mapping of any character is determined by the character and by its left and right contexts.

Let  $\alpha, \beta \tau : \Sigma \rightarrow Z$  be a mapping of characters to the signed integers determined by left context  $\alpha$  and right context  $\beta$ . Then  $\tau(a_1 \dots a_n) = \alpha_{i, \beta_1} \tau(a_1) \dots \alpha_{n, \beta_n} \tau(a_n)$  where  $\alpha_i$  and  $\beta_i$  are the left and right contexts of character  $a_i$ . For the moment, let us assume we have selected some well-defined mapping function,  $\tau$ .

In order to determine whether a line of text has structure similar to that of a hypothesized table, it is useful to crosscorrelate an unknown text line with an aggregate of  $m$  lines of the table that better represents its global structure.

In order to locate the beginning of a table, each line  $L_i$  is correlated with an aggregate of lines  $L_{i+1}, \dots, L_{i+m}$  called the forward aggregate. Similarly to locate the end of a table, line  $L_i$  is correlated with an aggregate of lines  $L_{i-1}, \dots, L_{i-m}$  called the backward aggregate.

Let  $\vartheta_i = \tau(L_i)$  be the numerical mapping of the  $i^{\text{th}}$  text line,  $L_i$  and let  $ln(L_i)$  be the length of the  $i^{\text{th}}$  line. In order that the correlation  $C_{fg}(0)$  of arbitrarily long random sequences  $f$  and  $g$  approaches 0 and  $C_{ff}(0)$  approaches 1, we normalize  $\vartheta_i$  as follows:

$$\eta = \frac{1}{ln(L_i)} \sum_{j=1} \vartheta_i(j) \quad (5)$$

$$\sigma^2 = \frac{ln(L_i)}{\sum_{j=1}^{ln(L_i)} (\vartheta_i(j) - \eta)^2} \quad (6)$$

and

$$\theta_i(k) = (\vartheta_i(k) - \eta) / \sigma, \quad k=1, \dots, ln(L_i) \quad (7)$$

$\theta_i$  is the normalized numerical mapping of line  $L_i$  and is used for crosscorrelation. Without loss of generality  $\theta_i$  is considered infinitely extended by zeros.

We also need normalized numerical mappings  $\theta_i^f$  and  $\theta_i^b$  for the forward and backward aggregates to be correlated with line  $L_i$ .

## 4.2 Forward and backward aggregates

### 4.2.1 Computing the forward aggregate

Let  $\vartheta_i^f$  be the sum of mapped lines  $\vartheta_{i+1}, \dots, \vartheta_{i+m}$  defined by

$$\vartheta_i^f(k) = \sum_{j=i+1}^{i+m} \tau(L_j)(k) \quad (8)$$

Let the length of the forward aggregate be represented by

$$l = \max(ln(L_{i+1}), \dots, ln(L_{i+m})) \quad (9)$$

We normalize  $\vartheta_i^f$  as follows:

$$\eta = 1/l \sum_{j=1}^l \vartheta_i^f(j) \quad (10)$$

$$\sigma^2 = \sum_{j=1}^l (\vartheta_i^f(j) - \eta)^2 \quad (11)$$

and

$$\theta_i^f(k) = (\vartheta_i^f(k) - \eta) / \sigma, \quad k=1, \dots, l. \quad (12)$$

$\theta_i^f$  is the normalized numerical mapping for the forward aggregate of lines  $(L_{i+1}, \dots, L_{i+m})$  and is used for crosscorrelation. Without loss of generality  $\theta_i^f$  is considered infinitely extended by zeros. Empty lines are not included in the forward aggregate.

#### 4.2.2 Computing the backward aggregate

Let  $\vartheta_i^b$  be the sum of mapped lines  $\vartheta_{i-1}, \dots, \vartheta_{i-m}$  defined by

$$\vartheta_i^b(k) = \sum_{j=i-1}^{i-m} \tau(L_j)(k) \quad (13)$$

Let the length of the forward aggregate be represented by

$$l = \max ( \ln(L_{i-1}), \dots, \ln(L_{i-m}) ) \quad (14)$$

We normalize  $\vartheta_i^b$  as follows:

$$\eta = 1/l \sum_{j=1}^l \vartheta_i^b(j) \quad (15)$$

$$\sigma^2 = \sum_{j=1}^l (\vartheta_i^b(j) - \eta)^2 \quad (16)$$

and

$$\theta_i^b(k) = (\vartheta_i^b(k) - \eta) / \sigma, \quad k=1, \dots, l. \quad (17)$$

$\theta_i^b$  is the normalized numerical mapping for the forward aggregate of lines  $(L_{i-1}, \dots, L_{i-m})$  and is used for crosscorrelation. Without loss of generality  $\theta_i^b$  is considered infinitely extended by zeros. Empty lines are not included in the backward aggregate.

### 4.3 The forward and backward crosscorrelations

In Equation 2, let

$$f = \theta_i \text{ and}$$

$$g = \theta_i^f \text{ or } \theta_i^b$$

Then, letting  $\mu = \min(\ln(L_i), l)$ , the forward crosscorrelation for line  $L_i$  would be:

$$C_{fg}^f(0) = \frac{1}{\mu} \left( \sum_{k=1}^{\mu} \theta_i^f(k) \theta_i^f(k) \right) \quad (18)$$

and similarly, the backward crosscorrelation for line  $L_i$  would be:

$$C_{fg}^b(0) = \frac{1}{\mu} \left( \sum_{k=1}^{\mu} \theta_i^b(k) \theta_i^b(k) \right) \quad (19)$$

For the free text both the forward and backward crosscorrelation values are very low (usually below 0). For perfectly structured text the value will be high (near 1).

## 4.4 The heuristics of line mappings

In Sections 4.1 and 4.3 we showed how crosscorrelation was used to compute a measure of similarity between a line of text and the lines that follow or precede it. The success of crosscorrelation is critically dependent on the choice of mapping,

$$\tau : L_i \rightarrow \vartheta_i$$

A great deal of empirical work involved different types of mappings. In the end we found that a simple homomorphic map gave the best results.

$$\tau(\text{number or letter}) = +1$$

$$\tau(\text{punctuation}) = +2$$

$$\tau(\text{whitespace}) = -1$$

For example, if

L = Eureka SD 1881-1889

then

$$\tau(L) = -1-1-1-1-1-1+1-1-1+1+1+1+1-1-1-1-1+2-1-1-1-1$$

The rationale is simple. Text aligned with text or whitespace aligned with whitespace is positively correlated, while whitespace aligned with text is anticorrelated. Punctuation received a higher weight because it is uncommon, hence more important. Some of the other mappings we tried are given in Appendix A.

## 4.5 Crosscorrelation results

Figure 6 shows the result of the crosscorrelation process applied to sample free text. In the figure the forward crosscorrelation values ( $C_{fg}^f(0)$ ) are shown in Column 1 and the backward crosscorrelation values ( $C_{fg}^b(0)$ ) are shown in Column 2.

-0.026	-0.287	Remarks]. Each Confirmant's record was given a Number, in sequence, by year.
0.009	0.019	The locations of the Ceremonies were sometimes named by both their German and
0.258	0.014	Polish names--e.g., the village of Erdmannsweiler/Kochanow.
0.000	0.000	
-0.172	-0.104	Where possible, I added data elements to each record as follows: name
0.169	0.044	[German/Polish] of the Location of the Confirmation Ceremony as found on the A.
0.042	-0.102	Breyer Map [Loc-A Breyer Map]; name [German/Polish] of the Village on the A.
0.102	0.127	Breyer Map [Vil-A Breyer Map]; and Remarks. The Remarks data element includes
0.040	0.036	notes about where the Locations or Villages can be found in relation to larger
-0.059	0.026	cities [these notes are often included in the "Loc- & Vil-A Breyer Map"
0.097	-0.126	fields]; locations found only on the Karl Stumpp Map VIII; or locations found
-0.007	-0.088	only on modern maps of Poland. In some cases, the listed Ceremony Locations
-0.015	0.061	and Villages could not be found on any map in my possession. However, I have
-0.006	0.114	been able to determine the general location of such Locations/Villages, and
-0.015	0.005	have indicated so in the Remarks. In addition, I have sometimes added notes
0.116	-0.040	about some confirmants, for whom I have found records in Bessarabia or Odessa.
0.000	0.000	
0.085	-0.283	In general, the extraction of the names of the confirmants should be accurate,
0.178	0.123	since most of these data were easy to decipher. However, surname spellings can
0.097	-0.059	vary, so be flexible when searching for a particular surname. The names of the
0.175	0.229	Locations and Villages is another matter. Often these are Polish names and the
-0.101	0.195	scribe did not always use the same spelling from one group of records to
0.277	0.075	another. In addition, the scribe often used abbreviations, which has made
0.098	-0.092	deciphering the resident village name problematical in some cases.
0.000	0.000	
0.300	0.300	DEDICATION
0.000	0.000	
0.070	0.229	The ILOW Parish Confirmations extraction work is dedicated to the memory of
0.048	-0.105	Philipp Jacob Seefried, great great grandfather of my wife, Janice Huber-
0.100	0.222	Stangl. His confirmation record, along with his older brother, Georg Adam
-0.028	-0.041	Seefried, is found in 1813, in Birkenfelde/Brzozow, located south of Rawa,
-0.123	0.039	Poland. It was the search for Philipp Jacob Seefried that led me to the ILOW
0.219	0.152	film, to make the decision to extract the data, and to publish it for all who
0.228	0.117	might be interested. Georg Adam and Philipp Jacob Seefried were born in

Figure 6 : Crosscorrelation in free text

Figure 7 shows the result of the crosscorrelation process with a mixture of free text and tabular data. The forward crosscorrelation values at the beginning of the table are high. The backward crosscorrelation values are low at the beginning because the table data is crosscorrelated with the preceding free text. Similarly, the forward crosscorrelation values at the end of the table are low because table data is crosscorrelated with the following free text.

0.983	0.974	Brunswig, J.A.	8	2S	41W	1928	KS	Cheyenne
0.984	0.982	Brunswig, J.A.	17	2S	41W	1928	KS	Cheyenne
0.976	0.986	Brunswig, J.A.	18	2S	41W	1928	KS	Cheyenne
0.967	0.991	Brunswig, J.A.	5	2S	41W	1907	KS	Cheyenne
0.959	0.995	Brunswig, J.A.	7	2S	41W	1907	KS	Cheyenne
0.939	0.997	Brunswig, J.A.	8	2S	41W	1907	KS	Cheyenne
0.931	0.986	Brunswig, J.A.	17	2S	41W	1907	KS	Cheyenne
0.897	0.989	Brunswig, J.A.	18	2S	41W	1907	KS	Cheyenne
0.833	0.989	Brunswig, J.A.	12	2S	42W	1907	KS	Cheyenne
0.785	0.895	Brunswig, John A.	6	2S	41W	1907	KS	Cheyenne
0.773	0.800	Dankenbring, E.H.	35	3S	37W	1928	KS	Cheyenne
0.613	0.878	Darbey, Theodore	13	1S	42W	1928	KS	Cheyenne
0.554	0.802	Deyle, G.F.	1	1S	37W	1928	KS	Cheyenne
0.088	0.837	Deyle, G.F.	15	1S	37W	1928	KS	Cheyenne
0.000	0.000							
0.000	0.000							
0.027	0.104	It is highly recommended that you use the film list which is maintained on the						
-0.011	-0.016	Bessarabian homepage. The earlier films of our Bessarabian church records were						
-0.110	0.048	proven to be of lesser quality than the 17xxxxx and later series of films. The						
-0.125	0.050	data in this file was taken from some of those earlier filmed data. As we learn						
-0.041	-0.009	of more Bessarabian films being available in the LDS system we will post to the						
0.300	-0.197	homepage film listing.						
0.000	0.000							

Figure 7 : Crosscorrelation result for a table followed by free text

Figure 8 shows the result of the crosscorrelation process for a highly irregular table.

0.805	0.300	Markquart, Johann	27	May	1855			
0.300	0.300	71						
-0.041	0.004	Zeh, Margaretha						
0.532	0.560	Maier, Johann	2	Jun	1855			
0.300	0.300	72						
-0.018	0.109	Schaefer, Christina						
0.546	0.813	Wiese, Friedrich	2	Jun	1855			
0.300	0.300	73						
-0.058	-0.022	Keller, Elisabetha						
0.573	0.746	Thumlert, Konrad	24	Jun	1855			
0.300	0.300	74						
-0.044	0.079	Dietrich, Katharina						
0.655	0.647	Gerstenberger, Samuel	19	Aug	1855		26	
-0.029	-0.173	75 Groom is a widower						
0.672	0.561	Klemens, Margaretha					21	Klemens , Jakob
-0.046	-0.011	of Bergdorf (Gluckstal Colony)						
0.487	0.282	Adam, Andreas	9	Sep	1855	[Benjamin Rossler JJW]	32	
-0.150	-0.010	76 Groom is a widower						
0.462	0.597	Rossler, Katharina				[Scheurer Jjw]	29	
0.007	-0.002	Bride is a Widow						
0.606	0.496	Neitz, Philipp	9	Sep	1855			
0.300	0.300	77						
-0.014	0.091	Ost, Katharina						
0.681	0.586	Schneider, Adam	9	Sep	1855		22	Schneider, Christian
0.300	0.300	78						
0.147	0.387	Benzinger, Christina (Christine)					22	Georg Benzinger, [Johann]
0.657	0.485	Rieger, Martin	23	Sep	1855			
0.300	0.300	79						
-0.040	-0.024	Schmied, Christina						
0.700	0.591	Berzeth, Anton	23	Sep	1855			
0.300	0.300	80						

Figure 8 : Crosscorrelation results for an irregular table

# Table Detection Heuristics

## 5. Table Detection Heuristics

After the forward and backward crosscorrelation values are obtained, the next task is to locate the tables in the text files. We have derived some table body detection heuristics to accomplish this task. In order to increase the accuracy of the heuristics, we had to make some basic assumptions and establish some constraints. They are as follows:

- 1) Tables should have at least 3 lines, so that at least some alignment is detectable.
- 2) Leading whitespace due to the indentation of free text is ignored.
- 3) Our experiments showed that for perfectly aligned tables, the table entries have very high correlation values ( $C_{fg}^f(0) > 0.7$ ), but to minimize decision errors (rejecting correlated lines or accepting uncorrelated lines) our detection threshold was lowered empirically to  $C_{fg}^f(0)$  or  $C_{fg}^b(0) = 0.4$ .
- 4) Our experiments showed that for tables surrounded by free text, the number of lines considered in the forward and backward aggregate affect the crosscorrelation values of the starting and ending rows of the table. If the number of lines in the aggregate is high, then free text surrounding the table reduces the crosscorrelation values of the beginning and ending table rows. This could lead to incorrect table detection. Hence to minimize decision errors, the group size for the forward and backward aggregates were kept at  $m = 10$  in Equations 8-17.

## 5.1 Table body detection heuristics

1. Start forward scanning and computing forward crosscorrelation values until a high crosscorrelation is found at line  $i_1$  ( $C_{fg}^f(0) \geq 0.4$ ). Line  $i_1$  may be the first row of a table. Continue forward scanning, skipping empty lines where the forward crosscorrelation value would be zero, until the forward crosscorrelation becomes low ( $C_{fg}^f(0) < 0.4$ ) at row  $i_2$ .  $i_2$  should be a row near the end of the table or some irregularity in the table structure such as an usually short or missing table entry or one that is misaligned.
2. Beginning with line  $i_2$  start forward scanning and computing the backward crosscorrelation values. Skip empty lines where the backward crosscorrelation value would be 0. Quit when backward crosscorrelation value becomes low ( $C_{fg}^b(0) < 0.4$ ) at line  $i_3$ . Line  $i_3$  may be the end row of the table or the first row of a new table or free text following the current table.

In Figure 9,  $i_1$  is the first row of the table (from Step 1),  $i_2$  follows the lines with high forward crosscorrelation values (from Step 2) and  $i_3$  is the last row of the table (from Step 3).

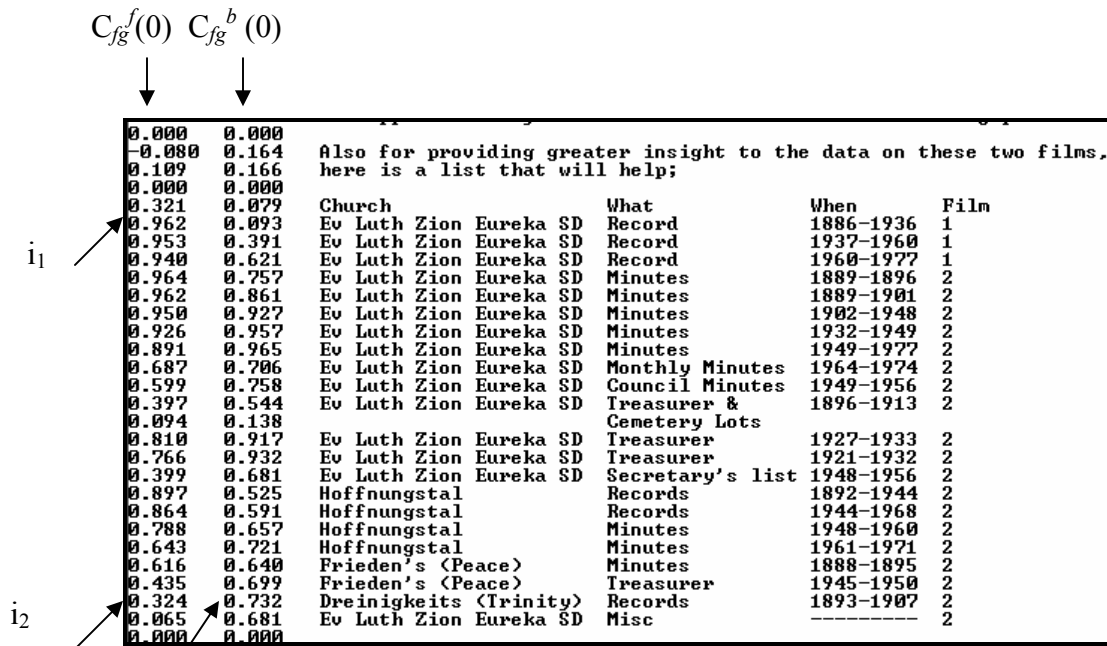


Figure 9 : Table detection using crosscorrelation values

3. If the line  $i_3-1$  is empty, then line  $i_3$  is not the last row of the table. Scan backward until a non-empty line  $i_4$  is found. The table extends from  $i_1$  to  $i_4$ .
4. If line  $i_3+1$  is non-empty then line  $i_3$  is an irregularity in the current table. Under such conditions, check if line  $i_3+1$  has a high forward or backward correlation. If yes, then we can continue forward scanning the backward correlations as in Step 2 to find the correct end row,  $i_4$ , for the table.
5. If neither conditions 3) nor 4) hold then the table extends from  $i_1$  to  $i_3$ .
6. After the first and last table rows are found, store the line numbers of these lines. Now the process begins again, starting with line  $i_4+1$ . Repeat Steps 3 through 6 until the end of the file is reached.
7. After applying these heuristics, all the well-formatted tables are identified.
8. Also it may happen that due to some irregularity in the table, the table is split into several parts. Usually the start row of the second part will follow the end row for the previous part. Hence we scan sequentially through the identified tables and merge those that are immediately adjacent.

## 5.2 Table header detection heuristics

Once we have found a table location in a plain text file, the next task is to locate the table header, which may or may not be included in the table that has been detected. Frequently a header may be detected as part of a table because it has the same alignment as lines in the table body. The table header could be a single line or a multi-line header. The table header is often separated from the table body by an empty line or by a line containing some special characters such as punctuation. The header might be attached to the table body. Hence we

need to derive general header detection heuristics that would take care of these different cases. Since we are concentrating on one-dimensional vertical tables, it is safe to assume that every table will have a horizontal header. The problem is to determine which lines are header lines and which belong to the table body.

Let us refer to the first line of the detected table as the Start of Table or simply, SOT. The problem then is to locate the actual Start of Header (SOH) and Start of Body (SOB), which will be somewhere, near the SOT. In the following discussion, a paragraph boundary is an empty line before the SOT. While considering the SOT, there are three possibilities:

1. The rows before and after the SOT are empty. Under such conditions the SOH is part of the table and is the SOT. Here the SOH is a single-line header. Hence we need to locate the actual SOB, which is the first non-empty row after the SOT. This is illustrated in Figure 10.

The diagram shows a table with a single-line header. The label 'SOT' has an arrow pointing to the first row. The label 'SOH' has an arrow pointing to the first row. The label 'SOB' has an arrow pointing to the first row. The table content is as follows:

Last, First	Date	Year	Place	Father	Mother	Film/It	Pge	Reg	Remarks
Abermann, Jakob	7 May	1833	Neu Arcis	Jakob	Timm, Justine	1766532/1	8	6	
Arndt, Andreas	9 Feb	1840	Neu Arcis	Johann	Grieb, Karolina	1766532/1	31	2	
Arndt, Christian	21 Nov	1858	Neu Arcis	Johann	Grieb, Karolina	1766532/1	101	25	
Arndt, Christoph	22 Dec	1855	Neu Arcis	Johann	Grieb, Karolina	1766532/1	85	17	
Arndt, David	23 May	1835	Neu Arcis	Johann	Grieb, Karoline	1766532/1	13	4	
Arndt, Gottlieb	9 Jan	1845	Neu Arcis	Johann	Grieb, Karolina	1766532/1	40	1	
Arndt, Henriette-Luise	7 Jan	1837	Neu Arcis	Joh.	Grieb, Karoline	1766532/1	22	1	Grieb?
Arndt, Jakob	17 Jul	1842	Neu Arcis	Johann	Grieb, Karoline	1766532/1	35	12	
Arndt, Johann	6 Oct	1838	Neu Arcis	Johann	Grieb, Karoline	1766532/1	26	11	
Arndt, Maria	27 Jan	1847	Neu Arcis	Johann	Grieb, Karolina	1766532/1	47	3	
Arndt, Michael	13 Aug	1849	Neu Arcis	Johann	Grieb, Karolina	1766532/1	55	19	
Arndt, Simon	28 Oct	1852	Neu Arcis	Johann	Grieb, Karolina	1766532/1	66	20	
Barts, Wilhelm	6 Mar	1830	Neu Arcis	Johann	Ziebart, Maria	1766532/1	2	4	

Figure 10 : File with a single line header separated from table body

2. The second possibility is that the row before the SOT is empty but the next row is non-empty. This is a typical case in which a table header immediately precedes the table body. If the nature of the table header is very similar to table entry, then its forward crosscorrelation value would be high. Hence, we need to look at the rows immediately following the start of the table to see if they provide any special clues. If the row following the SOT is a special row (punctuation) then the SOT is the SOH. The first non-empty row after the special row would be the SOB, as in Figure 11.

SOT      SOH      SOB

Last, First	Born	h#	p#	f#	Film/It #	Vil.	Remarks
ADOLF, Alexander	1901		107	202	1766563-2	Paris	
ADOLF, Philipp	1869		142	237	1766563-2	Paris	Brienne (Eigenfeld)
ADOLF, Theresia	1859		172	267	1766563-2	Paris	Brienne
ALBMER, Simon	1875		127	6	1766563-1	Paris	
ALLMER, Alexander	1899		176	271	1766563-2	Paris	
ALLMER, Johanna	1903		176	271	1766563-2	Paris	
ALLMER, Samuel	1869		96	537	1766562-3	Paris	
ALLMER, Simon	1895		168	263	1766563-2	Paris	
BADER, Andreas	1880		150	29	1766563-1	Paris	
BADER, Emil	1924		40	135	1766563-2	Paris	
BADER, Ferdinand	1869		49	490	1766562-3	Paris	
BADER, Ferdinand	1871		79	520	1766562-3	Paris	

Figure 11 : Table with header attached to the body but with a special line separator

If there is no special line immediately following the SOT, then we need to analyze the context immediately preceding the SOT to see if there is any potential table header. We examine the forward correlation value for lines immediately preceding the SOT until we reach a paragraph boundary. If the paragraph contains a single line then it is the SOH and SOT is the SOB as shown in Figure 12.

SOT      SOH      SOB

PAGE 2 DANIEL & KATHARINA OPP FAMILY: [Odessa Twp, 1900]							
Daniel Opp	b	23	Mar	1842	Glueckstal, S	Russia	d Jun 1916
Katharina Opp nee Gohl	b	5	Apr	1847	S	Russia	
Jakob Opp	b	25	Apr	1870	S	Russia	
Daniel Opp	b	10	Jul	1872	S	Russia	
Katharina Opp	b	26	Jul	1874	S	Russia	
Johann Opp	b	3	Feb	1877	S	Russia	[mar Elisabetha Neuharth]
Christian Opp	b	24	Nov	1881	S	Russia	
Friedrich Opp	b	28	Mar	1886	America		[mar Lydia Hoff]
Heinrich Opp	b	27	Apr	1888	America		
Rosina Opp	b	23	Jan	1890	America		
Christina Opp	b	1	Dec	1891	America		

Figure 12 : A single line table header separated from the table body

If the paragraph is very small (2-3 lines) and at least one of the lines has a high forward crosscorrelation with the SOT, then the paragraph is a multi-line table header and the SOT is the SOB as in Figure 13.

Surname, First Name Vil-A Breyer Map	Confirmation D/M Year Location Page Num Remarks	Loc-A Breyer Map
Abraham, Catharina	21 Apr 1808 Wilhelmstahl 274 360 ?Lanieta [Mod Pol Map], NW	Wilhelmstal/Augustopol Kutno
Abraham, Elisabeth Myszory	26 Feb 1811 Wiece 270 31 E of Ilow	Neudorf/Nowawies, by Sladow
Abraham, Jacob Myszory	26 Feb 1811 Wiece 270 26 E of Ilow	Neudorf/Nowawies, by Sladow
Abraham, Johann	24 Mar 1813 Slowick 278 54	Slowik, SE of Ozorkow
Abraham, Peter	25 Apr 1814 Wilce 286 360 Wilkow Polski [Mod Poland]	Wilkow, E of Wyszogrod

Figure 13 : Table with a multi-line header

Otherwise, the SOT is the SOH, and the row following the SOT is the SOB.

3. The third possibility is that a non-empty row exists before the SOT. If there is a special row immediately preceding the SOT then it is clear that the table header lies above the special row. Hence we can backtrack to the paragraph boundary and extract the SOH. Then the SOT becomes the SOB as shown in Figure 14.

Surname, Given [Maiden]	Died *Burial	Year Time	Father Given Mother Maiden, Given	Husband	Birth Place
Vankitchen, Nicolas	5 Dec *7 Dec	1843 midnight	Alexandre Nohienberg, Emilie		Gantscheski
Tardent, Jacques Emile	19 May *21 May	1844 2 AM	Charles Bolleneegger, Jeanne Phil.		Akermann
Logoz, Catherine [Pelleronz]	14 Jun *15 Jun	1844 6 AM	Jean	Abel	Manheim (colony)
Beck, Sophie Lisbeth	16 Jun *17 Jun	1844 2 AM	Christian Sturm, Catherine		Akermann
Tapis, Louis Christophe	17 Jul *18 Jul	1844 7 AM	Jacques Francois Dell, Catherine		Chabag
Rigger, Martin	26 Sep *28 Sep	1844 5 PM	Jean Laurim?, Regina		Ittrigen, Wue
Bfroth, Elisabeth [Schneider]	8 Nov *10 Nov	1844 5 PM	Peter	Andre	Russelsheim

Figure 14 : Header above the detected SOT separated by a special line

If there is no special row immediately preceding the SOT, then we can backtrack to the paragraph boundary to find the SOH. Then the SOT becomes the SOB as in Figure 15.

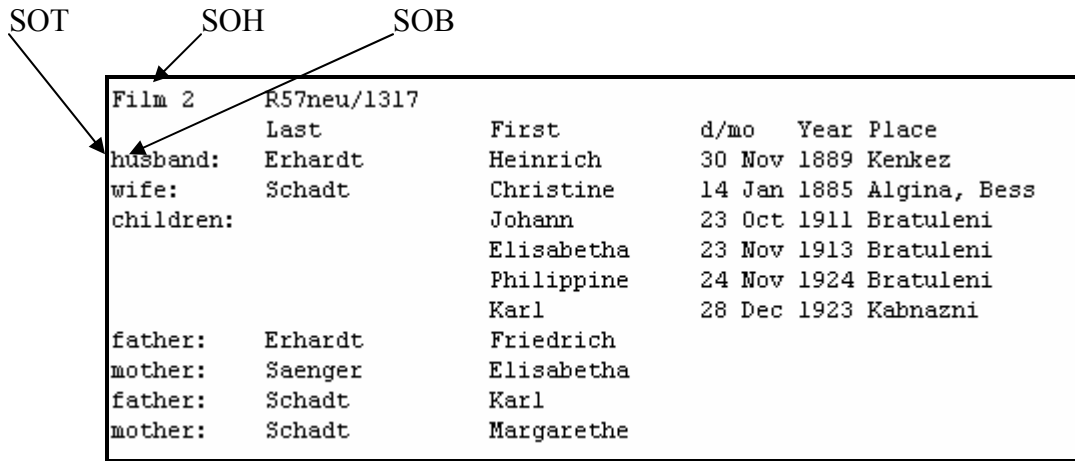


Figure 15 : Header above the detected SOT

# Integration with the Odessa Digital Library

## 6. Integration with the Odessa Digital Library

At indexing time, the existing system uses a document dictionary (docs.def) to obtain document information. This document dictionary is modified to include additional information so that the library administration may exclude a particular text file from table processing. The table detection system uses this information to analyze tables. This is the only functional connection between the existing system and the table detection system. If all the text files are excluded from table processing, the existing system behavior is not affected. Since both the systems operate independently of each other, the overall system is fault tolerant. Files can be processed for table detection at indexing time or separately. After the files are processed for table detection, a binary file containing fixed length records is generated (bintables.txt). This binary file stores the filename, the table locations, and the header locations for all tables.

This table information file is then loaded into memory at search time and is accessed by the search program by which the search results are formatted. The existing search code is modified to suit the requirements for displaying the table information. When table headers are included in the search results, the existing information display is modified to add the table header along with the search results and display the table headers in a different color. The information in the document dictionary is used at search time to prevent the display of table information from excluded files. For these files, the search program produces the default information display specified in the document dictionary.

When documents are modified or added into the existing system, the table detection system will process only newer documents. Similarly, when a document is deleted, the record for that document would be removed from the binary file containing the table information. The table detection system takes as input a text file containing the filenames to be processed and the following options.

tables [-a] [-d] [-u] filename

tables - the executable program

a – add to the library, process for table detection.

u – update in the library, process again for table detection.

d – delete from the library, remove the table records from bintables.txt

filename – file containing the list of documents to be processed.

This way the table detection system keeps the bintables.txt up to date when the document collection of the existing system is modified.

When the command line options are omitted, then the table detection system uses the entire document collection for table analysis.

Figure 16 shows the architectural diagram for the new system.

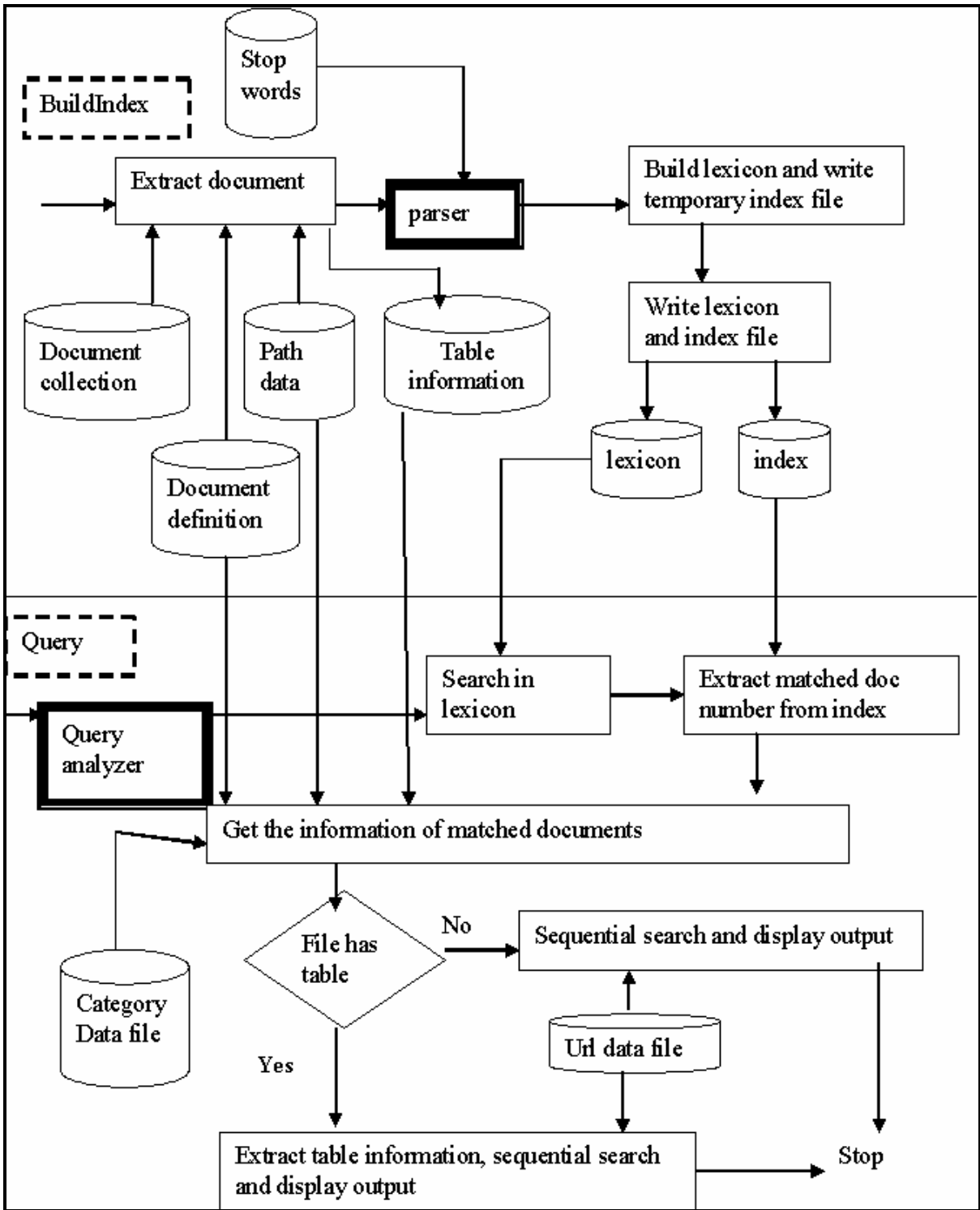


Figure 16 : Architecture diagram for the new system

Key:



# Experimental Results

## 7. Experimental Results

The Odessa Digital Library contains 749 plain text files. Each of these 749 files was analyzed for tables. When the table body detection heuristics were applied to each of these files, it was found that a total of 599 files contained both tabular data and free text.

Our goal is to identify tables as accurately as possible, given the misaligned and missing information typically found in tables. The table extraction process itself is subject to precision and recall trade-offs. Precision (finding only the table lines) may have to be sacrificed in order to improve recall (finding all tables in the text along with some non-table lines) [4]. We decided to concentrate on improving the precision of the extraction process at a small cost to recall. The table detection process in no way compromises the information display of the original Odessa library. If a table is missed completely, Odessa's presentation is unchanged. If, on the other hand returned information is found to be in a table, the header information is added to the normal display in a pastel color. If the header is incorrectly detected, it does not affect the normal Odessa display.

The only situation in which the normal Odessa information display could be changed is when a table is detected in free text when in fact no table is present. Because we have decided to set our correlation threshold to favor recall, such changes are infrequent.

Tables tend to have irregularities due to misalignment of text. Due to misalignment of text in files, it is difficult to find more accurate and general table body detection heuristics that would take care of all the irregularities and yet still manage to detect all the other tables properly. When we analyzed the results, we found that tables were misidentified in only 20 files. When we further analyzed the misidentified tables we found that the misidentification was mainly caused due to irregularities in the files (16 files). In the remaining 4 files the misidentification was due to non-tabular data being arranged in some regular format that appeared tabular. Thus detection results were almost 96% accurate. Due to the limitations of

the table body detection heuristics, it was observed that if a file contains a table that has many irregularities, the table body detection heuristics tend to split a single table into multiple tables. Figure 17 shows a typical scenario where text is identified as a table but not as a single entity.

GANDY, PAULA				
GAUS, MARI	* 15 Sep 1930			
GERLACH, ANDREA ANTONELLA	* 29 Jul 1978	* HELENA		
GERLOCK, BARBARA ELLEN	* 28 Apr 1960	* SALINA		
GIACHINO, VENETIA ANNE	* 15 Mar 1962	* PLACER COUNTY		
GIENGER, CHRISTINA				
GIMBEL, ELIZABETH	* May 1858		+ 23 Sep 1926	+ TRIPP
GIMBLETT, WENDY SUE	* 4 Jan 1956	* KING COUNTY		
GOHL, FRIEDRICH				
GOHL, ROSEANNA	* 25 May 1897		+ 5 Feb 1955	+ LODI
GOODMAN, DOROTHY MARIE				
GOYETTE, WILBURETTA A				
GREEN, INA				
GREGSON, ALICE				
GRENZ, GOTTFRIED				
GRENZ, GOTTFRIED	* 10 Sep 1881	* GUELDENDORF	+ 1955	+ STREETER
GRIFFIN, JACKIE LOU	* 6 Jul 1944	* DALLAS		
GUMS, AARON JOSEPH	* 29 Aug 1982	* SACRAMENTO CTY		
GUMS, ALICE	* 24 Apr 1929	* STREETER		
GUMS, ALVIN	* 19 May 1927	* STREETER		
GUMS, ALVIN	* 1 Feb 1925	* STREETER	+ 11 May 1976	+ FARGO
GUMS, ALVINA	* 2 Nov 1923	* STREETER	+ 6 Sep 1988	+ HARLOWTON
GUMS, AMANDA ROSE	* 4 Jul 1992	* SACRAMENTO		
GUMS, AMY JO	* 15 Nov 1978			
GUMS, AMY MARIE	* 27 Feb 1977	* ITHACA	+ 28 Feb 1977	
GUMS, ANGELA LUKACS		* AUSTRIA		
GUMS, ANNA	* 20 Jul 1929	* STREETER		

Figure 17 : Table with irregularities

Figure 18 shows a search result where the table is split into multiple parts and only one part is identified. Since the table is misidentified, the header detection heuristics have also failed to identify the correct table header.

<b>File: <u>Walworth County SD, 1910 Census (LaR. Ketterling). . . . .</u></b>				
Dickman, Anna (servant)	15	R	1908	1A Selby City
Doerr, Carl	33	R	1893	1B Java town
\$				
Doerr, Jakob	43	R	1893	8B Glenn
\$				

Figure 18 : Search result when table is misidentified

One other form of misidentification occurs in cases where a section of free text appears to be properly aligned but is not actually part of a table. The table body detection heuristics fail under such conditions, and there is little that can be done without additional information about the text. Figure 19 shows one such example where a table is incorrectly identified.

Ei\#.11.2		- Jacob		
		b. 13 NOV 1841	d. 06 OCT 1849	
Ei\#.11.3	>	- Katharina		--> Ei-5
		b. 06 FEB 1844		
Ei\#.11.4		- Christoph		
		b. 18 JAN 1846	d. 04 FEB 1846	
Ei\#.11.5	>	- Johanes		=>> Ei-6
		b. 21 APR 1847		
Ei\#.11.6	>	- Gottfried		--> Ei-13
		b. 12 JUN 1849		
Ei\#.11.7	>	- Jakob		--> Ei-15
		b. 11 MAY 1852		
Ei\#.11.8	>	- Barbara		--> Ei-16
		b. 30 JUN 1854		
Ei\#.11.9		- Gottlieb		
		b. 16 FEB 1858	d. 25 JUL 1858	
Ei\#.11.10		- Karl		
		b. 04 AUG 1859	d. 24 DEC 1859	
Ei\#.11.11		- Margaretha		
		b. 12 NOV 1860	d. 14 NOV 1860	
Ei\#.11.12	>	- Gottlob		--> Ei-17
		b. 28 JAN 1862		

Figure 19 : Aligned text, misidentified as a table

Due to such misalignments in text and the limitations of table body detection heuristics, it was observed that the total number of tables detected per file could range from 1 to a maximum of 400 tables. Due to this large variance in the total number of tables per file, we decided to limit the number of tables per file, which considerably simplified coding. When the number of tables per file was limited to 10, a total of 518 files were selected for table processing at the retrieval time, which seemed to be a reasonable number when compared to the total number of files detected with tables (which was 599).

The major success of this table detection approach was that it successfully identified multiple tables in a single file. Figure 20 shows an example of a file that has 3 tables. The table detection heuristics successfully identified them as 3 separate tables.

Film 2	R57neu/1317					
	Last	First	d/mo	Year	Place	Remarks
husband:	Drummer	Valentin	5 Oct	1916	Alt Oneschti	m. 27 Sep 1940 in Kischinew
wife:	Schnell	Margarethe	16 Apr	1924	Alt Oneschti	
children:	no entry					
father:	Drummer	Jakob	12 Jul	1871	Gerstnitz	d. 3 Feb 1930 in Alt Oneschti
mother:	Purpur	Margarethe	22 Dec	1877	Neu Strimba	
father:	Schnell	Wilhelm	23 May	1904	Alt Oneschti	
mother:	Melchert	Magdalena	11 Oct	1907	Alt Oneschti	
Film 2	R57neu/1317					
	Last	First	d/mo	Year	Place	Remarks
husband:	Erhardt	Heinrich	30 Nov	1889	Kenkez	m. 5 Feb 1911 in Kischineff
wife:	Schadt	Christine	14 Jan	1885	Algina, Bess	
children:		Johann	23 Oct	1911	Bratuleni	
		Elisabetha	23 Nov	1913	Bratuleni	
		Philippine	24 Nov	1924	Bratuleni	
		Karl	28 Dec	1923	Kabnazni	
father:	Erhardt	Friedrich				
mother:	Saenger	Elisabetha				
father:	Schadt	Karl				
mother:	Schadt	Margarethe				geb. _?
Film 2	R57neu/1317					
	Last	First	d/mo	Year	Place	Remarks
husband:	Eichele	August	26 Dec	1910	Neu Saratzika	m. 24 Oct 1934 in Kischinew
wife:	Rauch	Elisabeth	12 Oct	1914	Strimba	
children:		Ferdinand	4 Jan	1935	Alt Oneschti	d. 11 Feb 1941 in Budweis
		Johann	12 Jun	1937	Principera El.	d. 25 Aug 1937 in Alt Oneschti
		Carline	31 Dec	1938	Principera El.	
father:	Eichele	Johann		1870	Grossliebental	d. 12 dec 1939 in Alt Oneschti
mother:	Romin	Ottilia		1882	Sarata	d. 1913 in Marienfeld
father:	Rauch	Ferdinand	14 Nov	1885	Strimba	d. 1903
mother:	Hollinger	Katharina	28 Jun	1892	Neu Scholtoi	

Figure 20 : File containing multiple tables

After the tables were properly identified, the next task was to apply the table header detection heuristics to locate the headers of the tables in each of these 518 files. It was observed that for all the files with properly identified tables, the table headers were also properly identified. The accuracy of table header detection heuristics was good enough to identify single as well as multi-line headers correctly. The table header detection heuristics failed only for files in which the tables themselves were misidentified.

After we gathered all the results, we modified the existing system's search process so that when a hit is found within a table in a file, we display not only the table entry but also the table header in the information display.

Figure 21 shows an example of the search result when the header feature for the tables was added. This figure shows that a single-line header and a multi-line header are both successfully identified.

**File: [Arndt, ND \(Towner Co.\) Birth Records \(J. Grissom\). . . . .](#)**

Last, First Name	D/M Year	Father	Mother
?, Lucie Orla Anna	12 Feb 1909		
Dams, Dorothea Soyfia	21 Apr 1907	August Dams	Martha Bork
Dams, Hanry Clarence	13 Feb 1908	August Dams	Martha Bork
Haller, Martha Frederika Charlotte	12 Feb 1902	Johann Haller	Katherine Hamerli
Johnson, Martha Caroline	2 Jan 1902	Martin Johnson	Martha Johnson
Langer, Martha Hildegard	16 Mar 1904	Joseph Langer	Emma Kuhnt
Richard, Martha Erna	20 Aug 1912	Fredrich Richard	Agate Sikartschuk
Schumacher, Martha Linda Maria	15 May 1911	John Schumacher	Karolina Triebwasser

**File: [Eureka SD Lutheran Marriage Records - Index \(D. Wahl\). . . .](#)**

Groom	Age	Bride	Age	Date of	Reg	Pag
Last Name, First		Last Name, First		Marriage		
Bohle, Friedrich	22	Klein, Martha	20	23 Mar 1919	363	286
Brosz, Heinrich	28	Lapp, Martha	19	28 Sep 1915	319	334
Heer, Wilhelm	22	Stoebner, Martha	20	2 Nov 1919	378	288
Kolb, Christian	24	Mehlhaf, Martha	21	6 Jan 1914	298	332
Lapp, Adolph	23	Thurn, Martha	19	5 Dec 1920	405	292
Maier, Jakob	29	Pressler, Martha	22	6 Jan 1920	385	290
Nies, Jakob	26	Aippersbach, Martha	19	30 Oct 1919	377	288
Tempel, Philipp	26	Harr, Martha	26	24 Apr 1919	365	288

Figure 21 : Search result with table headers

The data collection in the Odessa digital library is mostly genealogical data. We also experimented with a small set of plain text files (about 10 files) containing financial data. It was observed that the statistical crosscorrelation technique works exceptionally well with numerical data. Figure 22 shows the crosscorrelation results with numerical data, which shows very strong correlations. Financial tables tend to be densely populated with entries, whereas tables of genealogical information tend to be sparse, hence more challenging, since information is often unknown.

		Table	Line	Code	Year	Annual	Qtr1	Qtr2	Qtr3	Qtr4
0.516	0.300	1.02	1	QGDP	1987	6113.259	6013.328	6077.162	6128.104	6234.441
0.862	0.457	1.02	2	QPCE	1987	4113.375	4049.739	4101.538	4146.999	4155.275
0.919	0.751	1.02	3	QPCESTD99	1987	455.193	435.231	453.800	472.793	458.953
0.869	0.849	1.02	4	QPCESTN99	1987	1274.515	1265.952	1275.034	1275.994	1281.047
0.860	0.905	1.02	5	QPCESETSS	1987	2379.284	2346.803	2367.766	2390.649	2411.917
0.856	0.835	1.02	6	QGPDI	1987	879.265	863.358	863.942	860.500	929.258
0.915	0.915	1.02	7	QIFIX	1987	855.981	837.209	853.120	867.999	865.622
0.955	0.912	1.02	8	QIFIXNR	1987	572.471	553.979	566.805	585.117	583.984
0.905	0.899	1.02	9	QINRS	1987	224.284	217.330	219.022	228.280	232.505
0.875	0.864	1.02	10	QIPDE	1987	359.974	348.137	358.787	368.674	364.297
0.945	0.912	1.02	11	QIFIXRES	1987	290.683	291.341	294.104	289.317	287.964
0.775	0.784	1.02	12	QCBI	1987	29.588	33.883	17.263	-2.086	69.294
0.717	0.735	1.02	13	QNETEXP	1987	-156.217	-161.326	-159.540	-153.171	-150.832
0.967	0.939	1.02	14	QX0000	1987	407.954	383.583	399.344	416.731	432.169
0.954	0.929	1.02	15	QXMB0000	1987	271.422	252.134	262.820	278.614	292.120
0.949	0.953	1.02	16	QXS0000	1987	139.073	134.521	139.646	140.413	141.713
0.952	0.954	1.02	17	QMC0000	1987	564.171	544.909	558.884	569.902	583.001
0.926	0.956	1.02	18	QMMB0000	1987	445.793	431.929	440.692	451.305	459.260
0.918	0.973	1.02	19	QMS0000	1987	120.170	114.433	120.146	120.211	125.889
0.821	0.762	1.02	20	QGAAA	1987	1292.465	1278.359	1289.123	1292.379	1309.983
0.958	0.953	1.02	21	QFAAA	1987	597.761	588.554	597.298	598.213	606.979
0.921	0.929	1.02	22	QFDAA	1987	450.198	438.966	449.776	456.431	455.620
0.933	0.955	1.02	23	QFNAA	1987	146.519	148.781	146.477	140.468	150.347
0.918	0.918	1.02	24	QSLAA	1987	695.641	690.726	692.756	695.106	703.966
0.866	0.878	1.02	25	QRESIDT102	1987	-37.152	-44.115	-39.251	-35.033	-30.153
0.863	0.800	1.02	1	QGDP	1988	6368.359	6275.851	6349.841	6382.344	6465.242
0.835	0.783	1.02	2	QPCE	1988	4279.456	4227.971	4256.777	4291.554	4341.415
0.920	0.898	1.02	3	QPCESTD99	1988	481.547	481.732	481.498	474.889	488.057
0.869	0.862	1.02	4	QPCESTN99	1988	1315.141	1295.812	1307.413	1321.422	1335.950
0.860	0.893	1.02	5	QPCESETSS	1988	2477.213	2443.727	2461.734	2491.564	2511.766
0.856	0.854	1.02	6	QGPDI	1988	902.814	884.608	902.453	907.511	916.694
0.915	0.945	1.02	7	QIFIX	1988	887.136	871.468	886.675	891.222	899.190
0.955	0.935	1.02	8	QIFIXNR	1988	603.609	590.478	603.385	606.731	613.836
0.905	0.909	1.02	9	QINRS	1988	227.110	225.357	229.621	226.883	226.584
0.875	0.858	1.02	10	QIPDE	1988	386.945	375.992	384.726	390.061	397.008
0.945	0.916	1.02	11	QIFIXRES	1988	289.214	286.965	288.954	290.150	290.780
0.776	0.785	1.02	12	QCBI	1988	18.379	17.208	18.920	18.515	18.874

Figure 22 : Crosscorrelation results with numerical data

The table body detection heuristics worked well on these values, detecting these tables properly. The table header detection heuristics also worked well and detected all table headers properly.

# Conclusions

## 8. Conclusions

We have derived statistical table detection and extraction techniques for plain text files. These have been extensively tested using the Odessa digital library as a testbed. The empirical results have demonstrated that a statistical method based on crosscorrelation, which uses table detection heuristics to interpret these results, performs well. The heuristics also make use of additional clues about the table structure to separate the table header from the table body.

At this stage, the table detection system is a very basic system. We have only investigated the issues related to one-dimensional vertical tables. This system needs to be further expanded to handle more complex tables such as nested tables, partitioned tables, over-expanded tables, etc. This would require modifications to the existing system to incorporate the table layout related issues.

The current system is limited to 10 tables per file. This forces us to eliminate some properly detected tables. A better methodology needs to be adopted to consider all the detected tables at search time so that more headers will be provided in the search output.

Since the text files in the Odessa library have no TAB characters, we have no technique for making use of them in files containing TABs. TABs however are strong indicators of column alignment, since they don't normally occur in free text. The current weight scheme gives equal weights to all the punctuation characters. Further experiments are needed to adopt a variable weight scheme for punctuation characters and to analyze the effect of such a weight scheme on the crosscorrelation values

The weight scheme can be modified to extract additional information about underlying text such as capitalization of text, text in larger font, and text in bold, etc. Incorporating all this information would certainly make the weight scheme more accurate improving the overall

crosscorrelation results. One of the advantages of using this crosscorrelation algorithm is that, it does not need to take into consideration the issues related to non-proportional fonts. This is mainly because the weight scheme operates on individual characters and then uses only the generated number sequences for the lines to be crosscorrelated.

Further experiments can be done with the value of  $m$  (see Equations 8 and 13) used for calculating the aggregates. It is possible to keep track of crosscorrelation values as they are being generated. When a high forward crosscorrelation value is found, the value of  $m$  can be gradually incremented to crosscorrelate more table lines. On the other hand, when the forward crosscorrelation becomes low, the value of  $m$  can be reduced. The effect of such implementation cannot be predicted.

The current system detects the start and end of the tables. Given the location of a table, we could use this information to identify the actual table columns using techniques such as column accumulators. This information might be helpful for formulating more meaningful queries if table semantics were known.

## Appendix A - Character maps

In this work a number of mapping functions,  $\tau$ , were investigated. Five of them are given in the following table; the last one is the one that gave the best performance.

In this table, the string to be mapped is  $L = a_1 a_2 a_3 \dots a_l$ . Strings are processed from left to right so that when the  $i^{\text{th}}$  character  $a_i$  is processed, the mapping of  $a_{i-1}$  is known. Note that any mapping other than that specified in the following table has a weight of 0. Only mappings 2 and 3 consider the left or right context of character  $a_i$  for weight assignment.

Key:

w= whitespace character.

~w = Non-whitespace character

D= Numeric digit [0-9]

P = Punctuation character [ $\sim$ ,!,@,#,\$,%,&,\*,(,)\_,+,-,=,;,|,:]

A= Alphabetic character [A-Z,a-z]

\* = Don't care!!

$\phi$  = Start of line.

Scheme	Left Context ( $a_{i-1}$ )	Character $a_i$	Right Context ( $a_{i+1}$ )	Weight ( $a_i$ )
<b>1</b>	*	w	*	-1
	*	$\sim w$	*	1
<b>2</b>	$\phi$	$\sim w$	*	2
	*	$\sim w$	w	2
	$\sim w$	$\sim w$	*	2
	Weight ( $a_{i-1}$ ) = 2	$\sim w$	*	2
	$\sim w$	w	*	-2
	Weight ( $a_{i-1}$ ) = -2	w	*	-1
	*	w	$\sim w$	-2
	$\phi$	$\sim w$	*	2
<b>3</b>	*	$\sim w$	w	2
	$\sim w$	$\sim w$	*	2
	Weight ( $a_{i-1}$ ) = 2	$\sim w$	*	2
	$\sim w$	w	*	-2
	Weight ( $a_{i-1}$ ) = -2	w	*	-1
	*	w	$\sim w$	-2
	*	*	P	3
	$\phi$	P	*	2
<b>4</b>	*	w	*	3
	*	D	*	2
	*	A	*	1
	*	P	*	2
<b>5</b>	*	w	*	-1
	*	$\sim w$	*	1

Table 1: Experimental weight schemes

## Appendix B - Pseudo-code for table detection heuristics

rows = Total number of lines in the input text file.  
 $FC_i$  = Forward correlation value for the  $i^{\text{th}}$  row  
 $BC_i$  = Backward correlation value for the  $i^{\text{th}}$  row  
SRT = Start row of the table  
ERT = End row of the table  
Table[][] = Stores final start and end of detected tables

// scan the input file line by line and find out the potential table locations

```

while (i < rows) { // Find a table
  if ( i == rows -1 ) then break;
  if (  $FC_i \geq 0.4$  ) { // High forward crosscorrelation
    SRT = i; // Store SRT
    proceed = true;
    while ( proceed ){ // Find the final ERT
      end = false;
      while ( !end ){ // Find the potential ERT
        while (  $FC_i \geq 0.4 \parallel FC_i == 0$  ){ // All high forward
          i++;
          if ( i == rows-1 ) break;
        } // End of All high forward
        while (  $BC_i \geq 0.4 \parallel BC_i == 0$  ){ // All high backward
          i++;
          if ( i == rows - 1 ) break;
        } // End of All high backward
        ERT = i; // store the potential ERT
      }
      if (ERT < rows -1 ){
        if (  $FC_{ERT-1} == 0$  ) { // potential ERT = final ERT
          proceed = false;
          break;
        }
        if (  $FC_{ERT+1} \geq 0.4 \parallel BC_{ERT+1} \geq 0.4$  ) { // ERT = irregularity
          i = ERT+1;
          if ( i >= row-1) proceed = false; break;
        }
      }
    }
  }
  else proceed = false; break;
} // End of Find the final ERT

```

// Following piece of code checks if we need to backup ERT

```

backup = true;
while backup) { // Backup ERT
  if (  $FC_{ERT-1} == 0 \ \&\& \ BC_{ERT-1} == 0$  ){

```

```

    p = ERT -1;
    while (FCp == 0)
        p--;
    ERT = p; // Store ERT
    backup = false;
}
if (ERT - SRT >= 3){
    Table[count][0] = SRT; // Store the final start of table
    Table[count][1] = ERT; // Store the final end of table
    count++;
}
}
} // End of Backup ERT
} // End of High forward correlation
else
    i++; // End of Find a table
}

```

// At this stage all the tables from a file are detected. Now merge immediately adjacent tables to reduce the number of table generated due to irregularities.

```

while ( i < count){ // Scan all tables
    SRT = Table[i][0];
    while ( i < count && end == false ){ // Merge all adjacent tables
        while ( Table[i][1]+1 == Table[i+1][0] || Table[i][1] == Table[i+1][0])
            i++;
        ERT = Table[i][1];
        Table[j][0] = SRT; // Store the start of table
        Table[j][1] = ERT; // Store the end of table
        j++;
        end = true;
    } // End of Merge adjacent tables
    i++;
    end = false;
} // End of Scan all tables
}

```

Final number of tables in the file = j;

## Appendix C - Pseudo-code for table header detection heuristics

Tstart = start of the table  
 Tend = end of the table  
 Hstart = start of the header  
 Hend = end of the header  
 count = total number of tables  
 special row = row of punctuation

```

while ( i < count) { // Scan all tables
  p = Tables[i][0];
  if ( FCp-1 == 0 && FCp+1 == 0){ // Header separated from body
    Hstart = Hend = p; // Store the start and end of header
    p++;
    while ( FCp == 0) p++;
    Tstart = p; // store the start of table
    Tend = Tables[i][1]; // store the end of table
  } // End of Header separated from body
  if ( FCp-1 == 0 && FCp+1 != 0){ // Header may be attached to body
    if ( FCp+1 == special row ){ // Next row special row
      Hstart = Hend = p; // store the start and end of header
      p++;
      while ( FCp == 0 || FCp == special row)
        p++;
      Tstart = p; // store the start row of table
      Tend = Tables[i][1]; // store the end row of table
    } // End of next row special row
  } // Scan previous context
  else {
    proper = false;
    Tstart = p; // store the start of table
    while ( FCp == 0)
      p--;
    Hend = p; // store the end of header
    while (FCp != 0)
      p--;
    Hstart = p; // store the start of header
    if ( Hend - Hstart <= 3) then proper = true;
  } // End of Scan previous context
  if (!proper){
    Tstart = Hstart = Hend = Tables[i][0]; // store table information
    Tend = Tables[i][1];
  }
} // End of Header may be attached to body
if ( FCp-1 != 0){ // Header attached to body
  Tstart = p; // store the start of table

```

```
Hend = p-1;
while (FCp != 0)
    p--;
Hstart = p; // store the start of table
Tend = Tables[i][1]; // store the end of table
} // End of Header attached to body
i++;
} // End of Scan all tables
```

At the end store all the table information in the binary file, bintables.txt. This information will be accessed by the search program at search time.

# References

- [1] Xin Zhou, Fast Web-Based Information Retrieval System. Independent study report, Department of Computer Science, Virginia Tech, pp. 1-25, 2000.
- [2] Witten, I.H., Moffat, A., and Bell, T.C., Managing Gigabytes. New York: Morgan Kaufmann, 1999.
- [3] Daniela, Rus and Devika Subramanian, Customizing Information Capture and Access. ACM Transaction on Information Systems. Volume 15, pp. 67-101, 1997.
- [4] Pallavi Pyreddy and W. Bruce Croft, TINTIN: A System for Retrieval in Text Tables. Proceedings of the Second ACM International Conference on Digital Libraries, pp. 193- 200, 1997.
- [5] Bell, T. C., Cleary, J. G., Witten I. H., Text Compression. Prentice Hall Advanced Reference Series in Computer Science. pp. 238-239, 1990.
- [6] Dr Alan Fekete., Design and Data Structure (CS2002), Radix Search Trie Structure. <http://www.ug.cs.usyd.edu.au/~cs2/dds/trie.html>.
- [7] Adelberg, B., NoDose: A Tool for Semi-Automatic Extracting Structured and Semistructured Data from Text Documents. SIGMOD Record, Vol. 27, pp. 283-294, 1998.
- [8] Wang, H.L., Wu, S.H., Wang, I.C., Sung, C.L., Hsu, W.L., Shih, W.K., Semantic Search on Internet Tabular Information Extraction for Answering Queries. Conference on Information and Knowledge Management, Mclean VA, USA, pp. 243-249, 2000.
- [9] Douglas, Shona., Hurst, Matthew., Quinn, David., Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text. Fourth Annual Symposium on Document Analysis and Information Retrieval, University of Nevada, Las Vegas, pp. 535-545, 1995.
- [10] Xinxin Wang, Tabular Abstraction, Editing and Formatting. PhD thesis, Department of Computer Science, University of Waterloo, Canada, pp. 1-174, 1996.
- [11] Kornfield, William and Wattecamps, John, Automatically Locating, Extracting and Analyzing Tabular data. 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Melbourne, Australia, pp. 347-348, 1998.
- [12] H.Fujisawa, Y.Nakano and K.Kurino, Segmentation Methods for Character Recognition from Segmentation to Document Structure Analysis", Proceedings of the IEEE, Vol. 80, pp. 1079 -1092, 1992.

# Vita

Ashwini Pande was born on November 13<sup>th</sup> 1977 in Aurangabad, India. She graduated from Pune Institute of Computer Technology, Pune, India in 1999 with a degree in Computer Engineering. She then worked as a software developer with Morelinux.com.

She is currently pursuing Masters Degree in Computer Science at Virginia Tech. During this academic phase, she worked as research assistant and teaching assistant.