FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Table Extraction for Information Retrieval

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Software Engineering and Internet Computing**

by

**BSc. Amin Mirdamadi**
Registration Number 0625268

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Allan Hanbury
Assistance: Dr. Florina Piroi

Vienna, 21st July, 2018

Amin Mirdamadi

Allan Hanbury

# Erklärung zur Verfassung der Arbeit

BSc. Amin Mirdamadi
Tullnerbachstraße 66-68 3002 Purkersdorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 21. Juli 2018

Amin Mirdamadi

# Danksagung

An dieser Stelle möchte ich all jenen danken, die mich im Rahmen dieser Masterarbeit begleitet haben.

In erster Linie möchte ich Herrn Prof. Allan Hanbury und Frau Dr. Florina Piori herzlich für ihre enorme und profunde Kenntnis des Feldes Information Retrieval aussprechen, die es mir ermöglicht hat, wertvolle Einblicke in dieses Thema zu gewinnen. Ihre kontinuierliche Unterstützung , Erreichbarkeit und die direkte Kommunikation erlaubte mir erfolgreich zu arbeiten.

Mein aufrichtiger Dank geht an meine Frau für ihre Geduld und Aufmerksamkeit während der Zeit meiner Diplomarbeit. Sie ermutigte mich ständig, mein Bestes zu geben.

Mein besonderer Dank gilt meinemr Mutter und meinem Vater, Herrn Prof. Dr. Shamseddin Mirdamadi, für ihre Motivation und Unterstützung, die sie mir zeitlebens zukommen ließen. Mein Dank gilt auch meiner Familie, weil sie mich immer unterstüzt haben.

# Acknowledgements

Here I take the chance to thank everyone who helped me during my thesis work.

First and foremost, I would like to express my sincere gratitude to Prof. Allan Hanbury and Dr. Florina Piroi for their enormous and profound knowledge of the field Information Retrieval that enabled me to gain valuable insights into this subject and also their continuous reachability, support and direct communication provided me with the environment for successfully working on this thesis.

My deepest and sincere thanks go to my wife for her love and patience during all these years of my thesis work. She continuously encouraged me to do my best and helped me to finish my thesis.

My special thanks go to my father Prof. Shamseddin Mirdamadi and my mother for their motivation and support through my life, and I also want to thank my family because they are always supportive.

# Kurzfassung

Jedes Jahr werden in Information Retrieval wissenschaftliche Arbeiten in verschiedenen Feldern veröffentlicht, die wertvolle Informationen wie Tabellen enthalten, wobei fast alle Arbeiten im PDF-Format gespeichert sind [1]. Die unschätzbaren Daten in diesen Dokumenten und Veröffentlichungen sind in wissenschaftlichen Berichten von entscheidender Bedeutung, werden jedoch häufig von Suchmaschinen ignoriert. Die Extraktion tabellarischer Daten mithilfe moderner Tools und die Indexierung durch Suchmaschinen ermöglicht Forschern eine einfache Nutzung der extrahierten Informationen in verschiedenen Forschungsprojekten und Studien.

In dieser Arbeit evaluieren wir die bekannten Tabellenextraktionswerkzeuge und bewerten ihre Anwendung im Bereich Information Retrieval (IR). Wir haben ein Framework entwickelt, das die Ergebnisse der verschiedenen Table-Extraction-Tools aufnimmt und mit Methoden im IR-Evaluationsbereich vergleicht. Aus diesem Grund haben wir die Tabellenextraktionswerkzeuge ausgewählt, getestet und mit einer manuell erstellten Grundwahrheit bewertet. Um ein Ranking-System zu erstellen, berechnen wir einen Score für jedes der ausgewählten Tools, außerdem haben wir die generierte Klassifizierung mithilfe von IR-Experimenten ausgewertet. Daher haben wir ein Framework mit drei Schritten entwickelt und ein Testverfahren für die Analyse von bis zu 5870 PDF-Dateien vorbereitet. In dieser Arbeit beschränken wir die Verarbeitung von Tabellenextraktionswerkzeugen auf PDFGenie, PDF2Table und PDF2HTML.

Der erste Teil des Frameworks ist der Vorbeitungsschritt, in dem wir einen Service bereitstellen, der das Ergebnis der Tabellenextraktionstools verarbeitet, die normalerweise im XML- oder HTML-Format vorliegen. Dieser Dienst wandelt die bereitgestellten Tabellen in ein Standard-JSON-Format um und speichert sie in der Datenbank.

Der zweite Teil des Frameworks ist der Validierungsschritt, in dem wir die Leistung der Tabellenextraktionstools überprüfen. Das Framework bietet die Funktionalität zum Berechnen eines Wertes zwischen null und eins. Der berechnete Scorewert basiert auf dem Vergleich der extrahierten Tabellen mit den Extraktionstools und der bereits erstellten Grundwahrheit. Der Score ermöglicht das Erstellen einer Rangliste von Tabellenextraktionstools, die die Effektivität und Leistung der Tabellenextraktionstools anzeigen.

Der dritte Teil des Rahmens befasst sich mit der Bewertung, indem wir das Ergebnis des Validierungsteils untersuchen und die Genauigkeit der erzielten Ergebnisse überprüfen.

Hierfür stellen wir eine Reihe von Diensten zur Verfügung, die die von den Tabellenextraktionstools extrahierten JSON-Tabellen aus der Testsammlung in die Suchmaschine Solr indiziert. Der Evaluierungsprozess ähnelt dem Cranfield-Ansatz [2]: Wir definieren eine Reihe von Fragen und bewerten die Antworten der Suchmaschine.

# Abstract

Every year scientific papers and journals publish in different domains (fields) and contain valuable information embedded in tables, which are digitally stored in PDF format [1]. The invaluable data in these documents and publications are crucial in scientific reviews, yet are frequently ignored by search engines. Extracting tabular data by using modern tools and indexing by search engines allows researchers easy usage of the extracted information in various research projects and studies.

In this thesis, we evaluate the known table extraction tools and assess their application to the Information Retrieval (IR) field. We have developed a framework that takes the results of the various table extraction tools and compares them with methods in the IR evaluation area. For this reason, we selected the table extraction tools, tested and ranked them with a manually created ground-truth. To create a ranking system, we compute a score for each of the selected tools, furthermore we evaluated the generated classification by using IR experiments. Hence, we have developed a framework with three steps and prepared a test procedure to analyze as many as 5870 PDF files. In this thesis, we limit the processing of table extraction tools to PDFGenie, PDF2Table and PDF2HTML.

The first part of the framework is the preprocessing step where we provide a service that processes the outcome of the table extraction tools, which are usually in XML or HTML format. This service transforms the provided tables by the tools that are in XML or HTML format into a standard JSON format and stores them in the database.

The second part of the framework is the validation step where we validate the performance of the table extraction tools. The framework provides the functionality to calculate a score, which is a value between zero and one. The calculated score value is based on the comparison of the extracted tables by the extraction tools and the already created ground-truth. The score allows creating a ranked list of table extraction tools that show the effectiveness and performance of the table extraction tools.

The third part of the framework deals with evaluation as we examine the outcome of the validation part of the framework and verify the accuracy of the obtained result. For this, we provide a set of services that indexes the JSON tables extracted by the table extraction tools from the test collection into the search engine Solr. The evaluation process is similar to the Cranfield approach [2]: we define a set of questions and evaluate the answers provided by the search engine.

# Contents

CHAPTER 1

# Introduction

Information Retrieval (IR) is a set of techniques and approaches that retrieve relevant information for a given query when applied to a collection of documents or data [3].

Scientific documents in different domains consist of text, images and tabular data that could be used in different research area when they could be retrieved by IR systems. Tables contained in the documents are an effective way to convey structured data, where a two-dimensional layout serves to communicate, grouping connections and constraints. The richness of the information conveyed by a table makes extracting material from tabular data more complex than from a paragraph written in text format. At the same time, tables embedded in documents do not follow a common language and layout, their encoding is different depending on the software used to create such document containing tables. When tabular data is structured, it can easily be extracted from an HTML file; however, the same extractions process could be evasive when stored in an unstructured file format like image or pdf.

In systematic reviews, tabular data has a central role in presenting measurement values and data points provide the ability to manipulate, aggregate and use in different studies, which gives more to the analysis. The same methodology is applicable to different empirical domains, such as computer science, and its subdomains, e.g. IR.

## Problem statement

Every year scientific papers and journals in different domains are published which also contain tables. Search engines tend to ignore tables or only use very simple text indexing to find and use the relevant information imbedded in the documents. One interesting feature of IR is its ability to explore many different ways of extracting relevant data by combining table extraction tools with search engines that allow querying table-specific information.

An integral part of IR research is the evaluation of the retrieval methods (search engines and extraction tools). The IR evaluation process generally utilizes highly developmental campaigns such as: TREC [4], NTCIR [5], CLEF [6] and ICDAR [7]. Hence, having a framework that could evaluate the table extraction tools in an IR evaluation setting would provide a better understanding of these tools and the algorithms they are based on. This process can in turn lead to performance and quality enhancement of the extraction tools.

## Aim of the work

The aim of this work is to evaluate the known table extraction tools and assess their relevance to the IR field. We designed and developed a framework that took the results of the various table extraction tools and compared them based on methods in the IR evaluation area.

In order to evaluate the table extraction tools, we made use of a test collection technique, which has been created as part of this thesis.
A test collection technique consists of a set of documents, a set of topics or queries and a set of relevant documents to the topics, called relevance assessments.

To accomplish our goal we planned the following steps.

1. Analysis and preparation: We analyzed different table extraction tools and their outcome. Furthermore, we prepared a test collection containing 5870 PDF documents.

2. We processed the outcome of the table extraction tool to create a JSON document for each table and imported them into a database.

3. Validation: Our goal was to create a ranking list of the table extraction tools. For this we planned the following steps:

    - Ground-truth generation: We prepared 40 tables from the PDF documents in the test collection, converted them into JSON format and imported them into database for further processing.
    - Processing and ranking assignment: We processed the same PDF files as for ground-truth by the table extraction tools. Furthermore, we transformed the extracted tables into the JSON tables. We generated a ranking list of the table extraction tools by comparing the provided JSON tables of the table extraction tools with the already created ground-truth.

4. Evaluation: In this step, we want to confirm the obtained ranking of the tools in the previous step. to accomplish this we planned the following:

- Processing and indexing: We processed all PDF documents from the test collection by the table extraction tools. We indexed all the JSON tables for each table extraction tool.
- Ground-truth generation: We created 25 topics from the tables in the PDF documents, with the relevant answers to the topics.
- IR evaluation: We evaluated the outcome of the IR using the TREC EVAL tool and calculated precision, recall, mean average precision and mean reciprocal rank.

5. Experimenting with the framework: We experimented with the introduced framework and showed the calculated ranking of the table extraction tools. Furthermore, we provide the results from the IR evaluation.

## Methodological approach

The methodological approach to reach the expected results consists of the following steps:

- Chapter 1: Introduction

  This chapter provides a short description about the motivation, the problem statement and the objective of the work.

- Chapter 2: Information processing

  This chapter contains a detailed introduction in the fields related to the information processing such as Information Retrieval, Information Extraction, Test collections, Information Retrieval Evaluation and processing the PDF files.

- Chapter 3: Table identification and classification

  This chapter contains approaches to identification and classification of tables embedded in documents. Furthermore, the importance of table classification methods in the IR field discussed and the usage of these methods in the table extractions presented.

- Chapter 4: Table extraction systems

  In this chapter, we analyzed the table extraction tools that are publicly available. For each available table extraction tool, we operated tests with PDF files and published the results.

- Chapter 5: Framework development

  This chapter contains the development of a framework to create a ranking of table extraction tools and then evaluate them.

- Chapter 6: Experiments done with framework

  This chapter provides information about the experiments we have done on extraction tools using the framework.

CHAPTER 2

# Information processing

With the exponential growth of information in digital media, using IR algorithms has become a vital tool in retrieving stored information. Information is held and accessed in different formats such as web, text files, pdf files, etc. Storing documents in PDF format enables users a safe and reliable way to exchange and view electronic documents independent of the environment in which the files were created and viewed.

Information retrieval (IR) [3] is a distinctive set of techniques and approaches that return pertinent information based on a given query out of a collection of documents. For example, a PDF document consist of texts, images, and tabular data, important in the context of extraction and retrieval of such information. Tabular data in documents is a means to convey structured data, where the two-dimensional layout serves to communicate, grouping connections and constraints.

In the Medical domain (field), systematic review of scientific journals provides physicians and researchers up-to-date answers to complex questions in which tabular data plays a critical role [8]. Ordinarily, extracted data allows manipulation, aggregation and application of information to enhance scientific research and new discoveries. The same methodology is germane to different empirical domains (fields) such as computer science, and its subdomains, e.g. IR.

The richness of the information conveyed by a table makes extracting information from tabular data even more complex. At the same time, tables in documents do not always follow a structured language and layout due to different encoding and the software used to create the document containing the tables. For example, structured data can easily be extracted from an HTML file, while the same extraction process is evasive when stored in an unstructured file format like images or PDF records.

In IR, as well as in other domains, published papers and journals usually contain tables. Search engines usually ignore tables, or only perform very simple text indexing on them, which makes it difficult to search, find, and use the relevant information from tables in

documents. Therefore, an interesting subject in IR is to find opportunities to extract the data in a meaningful way by combining table extraction tools with search engines that allow querying table-specific information.

## 2.1 Information retrieval

Information retrieval (IR) [9] is a field that deals with analysis, extraction, storage, search and retrieval of information. Information is chronicled in many formats such as text files, image files, sound files, video files and so on. IR is the process of obtaining information from structured, unstructured, semi-structured or a combination of numerous documents allowing relevant data to be captured by a user query out of the processed documents.

Structured documents refers to documents that follow a structural logic and a methodical scheme. Structured data usually follows a standard language within the SGML families such as XML and HTML. SGML is a Standard Generalized Markup Language defined in ISO standard 8879:1986 for defining generalized markup languages for documents. The first use case of the SGML structure was in IR systems that; based on the given query, the IR system could retrieve parts of documents instead of the whole document. Documents that follow the SGML standard are considered structured documents [10].

Unstructured documents usually refers to digital information that does not have a data model or one that is easily usable by a computer program [11]. The common type of unstructured data is PDF, image and text files. Text mining and Natural Language Processing (NLP) techniques work with unstructured data. Text mining performs extraction of useful and interesting text data, however, NLP finds textual and natural language information using techniques such as text classification, text categorization and document clustering, finding groups of similar documents, information extraction and summarization [12].

Semi-structured information does not follow the formal structure of data-models of relational or data-tables, however, contain markers to separate semantic element of information or data. Dan Smith and Mauricio Lopez in 1997 defined a framework for extracting information from unstructured and semi-structured documents. The framework provides fast and accurate selective access to the extracted information [13].

The main goal of information retrieval systems is to find the relevant information from documents that satisfy the user questions and user information requirements [14]. There are three basic processes that information retrieval system has to support, as we can see in Figure 2.1 [15].

1. **Indexing process** represents the documents in a summarized content form containing informative terms. Indexing process maps the terms to the respective documents containing such information. Using indexing, the retrieval process is simplified since it does not need to do a linear search. The index can be stored in different data structure such as direct index, inverted index, document index

and lexicon. Indexing process has four main stages, the content specification, tokenization of documents, processing of document terms, and index building [16].

2. **Query formulation** is the processing of information needs called query. Query formulation is the interactive communication between the user and the system.

3. **Matching process** is a set of techniques for retrieving the documents that match with the users (query) requirements. The matching process usually results in a ranked list of documents. A ranking algorithm helps users by putting the most relevant documents at the top of the ranked results to help the user to find the information faster. The ranking algorithm uses the frequency distribution of terms over documents, and statistical information of the documents to work effectively.

There are two basic measurement systems for assessing the quality of information retrieval systems, which are precision and recall [17].
Precision is defined as the fraction of retrieved documents that are relevant to a query.
Recall is defined as the fraction of relevant documents to a query that is retrieved.

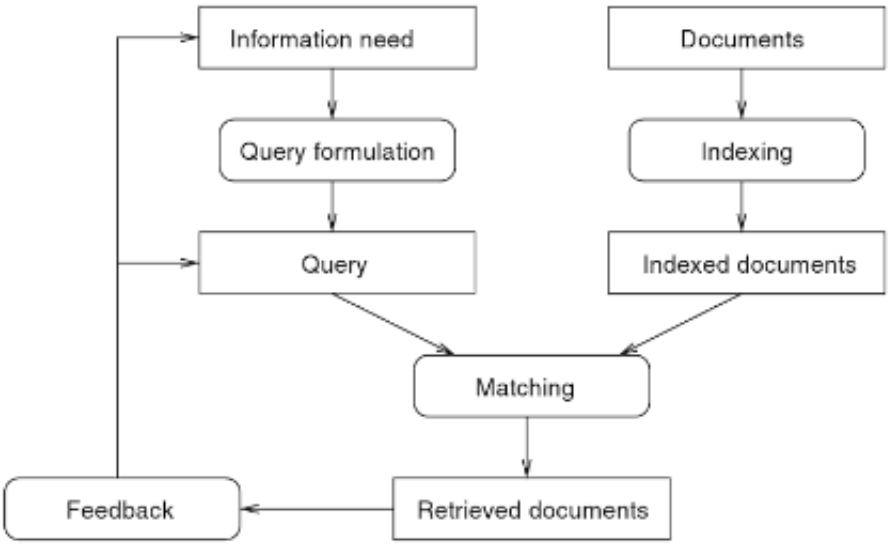



Figure 2.1: Information retrieval processes [15]

### 2.1.1 History

Information retrieval did not begin with the internet, its origins date back to the traditional approach of managing a vast accumulation of information from libraries. The very first computer-based search engines were built in the late 1940s for public use. The Univac machine [18], introduced by Holmstrom in 1948, was an information retrieval machine capable of searching text references related to a subject code. The code and text were

stored on a magnetic steel tape. Indexing is a crucial mechanism in information retrieval. Librarianship was the classic approach to organizing collections of data using the Dewey Decimal System, which assigned numerical code to documents, books, journals, papers, etc. The Uniterm System, which was implemented by Mortimer Taube [19], indexed item by a list of keywords.

Ranked retrieval is an important point in information retrieval that attempts to resolve the problem of sorting the information found based on a query in the search engine such that the best results appear first. The first style used in electro-mechanical and computer-based IR systems was Boolean retrieval where the results were the exact match of the query, and the query was a logical combination of terms. Luhn proposed the term frequency weighting method that uses the frequency of word occurrence in an article as a measurement for ranking [20].

In the 1960's, another important innovation in IR proposed by Gerald Salton was the introduction of relevant feedback [21]; this process was designed to support the iterative search so that the previously searched articles could be marked as important or relevant in an IR system.

In the 1970's, Spärk Jones complimented the Luhn term frequency and introduced the Inverse Document Frequency theory stating that the frequency of occurrence of a word in the document collection is inversely proportional to its significance in retrieval [22].

The ranking function BM25 [23] was an innovation based on the probabilistic retrieval framework still in use in IR systems.
From the 1980's through to the mid 1990's, important enhancements to IR systems such as automatically ranking and learned ranking results. Fuhr describes a method where the retrieval function was learned based on the relevant documents identified for an existing set of queries [24].

With the rise of the internet and the web, online searching quickly became very relevant. Before the web, people searched only selected and specific (terms , words) by IR systems. Researchers and developers realized the importance of precise searching on the web and swiftly found that they could crawl web pages by links between them and provide information to all people using the internet.

In the internet and web era, utilizing the query logs became extremely important when large volumes of people started using web search engines. Search engines could examine user clicks along with the queries performed by users and gather valuable information from the user, then it could apply this information for automated spell correction and automated query expansion [25].

In recent years, with the increased use of mobile devices, social media and the internet, new research areas have emerged. Information retrieval communities try to expand search engines in areas such as filtering and recommendations. User tagging, conversational retrieval as well as collaborative searches provide the tools necessary to support these

new fields. Stuff I've Seen (SIS) [26] is one of the first tools designed to support these areas.

### 2.1.2 Information Retrieval Models

Having a model is important for having a better understanding of the domain. Models provide the guide and meaning for academic discussion and can be seen as blueprints or paths to implement the retrieval system.
An IR model specifies the details of the document representation, the query representation, and the retrieval functionality [27]. IR models can be classified as Boolean model, Vector Space model, Probabilistic model and Inference Network model [15].

#### Boolean Model

The Boolean model is the first model of information retrieval and is described as thinking of a query term as an unambiguous definition of a set of documents. Query terms use Boole's mathematical logic operators to form new sets of documents. Boole defined three operators, the logical product (AND), the logical sum (OR) and the logical difference (NOT). An example of a query would be the term 'Medical' that defines a set of documents that are indexed with the term 'Medical'. Furthermore, we can use the combination of terms using Boole's operators such as *Medical AND Information AND Technology* to restrict the query and be more precise in the search.
The advantage of Boolean retrieval model is that it gives the user control over the system to obtain results that are clear for a set of documents. However, the disadvantage of Boolean retrieval is that it does not provide a ranking of retrieval documents.

#### Vector Space Model

The Vector Space Model [28] is established on Hans Peter Luhn's statistical approach to searching information. Luhn suggested a search method in which the user provides comparable representative documents, then utilizes the similarities between the documents to rank search results in the collection.

Gerard Salton and Michael J. McGill suggested the Vector Space Model, they considered that a document is modeled by a vector of keywords extracted from the data. Next, the terms are weighted to determine the importance and frequency of each keyword. There are several approaches to calculate this; however, the most common way is to use the term frequency-inverse document frequency (*tf-idf*) which is a two-step method. The first step is to determine how often the term $j$ occurs in the document $i$ and the second step is to determine how often it occurs in the whole document collection [15].

#### Probabilistic model

Stephan Robertson first introduced the Probabilistic Retrieval Model. Probability theory is based on an approach that tries to define weighting more formally. Robertson describes

probabilistic retrieval as follows: "If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data" [15].
Ranking documents by their probability of relevance to a given query are the important characteristics of the probabilistic model. Based on this model when the relevant information is available it should improve the overall retrieval performance. However, when no relevant information is available, then the model behaves like the vector space model using idf weights [29].

#### Inference Network Model

The basic document retrieval Inference Network Model consists of a document network and a query network. Documents retrieved in the Inference Network Model are modeled as an inference process in an inference network. In the basic implementation of this model, the document instantiates a term to calculate the equivalent of a score for the document. The weight of the term can be considered as the strength of the term instantiation for a document, and the document ranking is similar to ranking in Vector Space Model [30].

### 2.1.3 Review of some IR systems

This section presents some of the innovative and leading commercial research IR engines.

#### INQUERY

The INQUERY system was established on a form of probabilistic retrieval model called inference net, developed at the University of Massachusetts and was designed to work with large text databases [31]. Bayesian inference networks or Bayes net are probabilistic models that were used in INQUERY. Bayes net is a directed acyclic graph (DAG) where the nodes represent propositional variables and arcs represent dependencies.

The INQUERY framework has three major tasks; creation of a document network, creation of query network and use of networks to retrieve documents. The document parser is one of the key tasks of the system, it analyzes the structure of the document and transforms it into a canonical format, and it recognizes sections that require indexing. By performing the lexical analysis, it extracts words, fields, etc. and indexes the words. INQUERY offers batch and interactive methods of query processing, and an Application Programmers Interface (API) in order for developers to implement their customized front-end user interface to the retrieval engine.

### SMART

The SMART system, developed at Cornell University, is one of the first IR systems supporting automatic term indexing, query enhancements based on relevance feedback, and uses weighted term vectors in a term based vector space for documents and queries [21]. SMART has been widely used in IR research areas and in most cases as a baseline for comparison with other IR methods.

### INDRI

Indri is a search engine that provides state-of-the-art text search and a rich structured query language, developed by the University of Massachusetts and Carnegie Mellon University and is written in c++ [32]. The APIs, however, can be used from Java, PHP, and c++. Indri[1] is flexible and supports popular structured query operators such as and-query, and-not-query, not-query, not-in-query, or-query and can parse several document types such as PDF, XML, HTML, Word, PowerPoint and TREC documents. Indri combines the inference network framework with the new theoretical advances in language modeling. Furthermore, Indri provides new query language constructs incorporating fields, tags, and numbers to support query activity in question answering and cross-lingual retrieval. Indri can handle large collections and is capable of scaling up to cluster machines for efficient retrieval.

### LEXIS/NEXIS

LEXIS NEXIS is a commercial IR system for retrieving legal and newspaper documents, which during the 1970's, pioneered the electronic accessibility of legal and journal documents. It supports traditional Boolean queries, which return exact matches only. Lexis Nexis support combined with traditional Boolean queries and natural language queries used the vector space approach, where significant terms from the user query were identified and irrelevant terms removed, then a formula that weighs the importance of the terms from a query is applied. Terms had high statistical weight when it had low similarity or was uncommon in the document or file [33].

### LUCENE

Lucene, a mature, free, and open source Information Retrieval library developed in Java in 1997, by Doug Cutting is the most successful open source search engine. Lucene enables search capabilities to the applications and is based on the concept of fields, every document is constructed with several fields including document-id, body, URL, and content. Since the first implementation, it has grown to a global project, and hundreds of developers and companies are involved in the development. Lucene is currently hosted by the Apache Foundation and licensed under the liberal Apache Software License [2]. Many

[1] www.lemurproject.org/indri

[2] lucene.apache.org

enterprises have already integrated Lucene into their applications, such as Wikipedia [3] and Cisco [4] [34].

## 2.2 Information Extraction

In recent years, we are confronted with rapidly increasing textual information in digital media in different domains such as online news, government documents, medical information, and social media. The need to have an effective and efficient technique to acquire relevant information from the textual data is increasing in demand. Information extraction (IE) methods try to locate and understand relevant parts of texts, gather information from the texts related to a particular query, and present the information to the user [35].

IE systems and IR systems are both challenging and knowledge-intensive to build; however, we can see both techniques as complementary such that IE can use IR for pre-filtering a very large document collection to a smaller subset and IR can use IE to identify structures for intelligent document indexing [36].

Following are some methods of information extraction [35] :

- **Named Entity Recognition (NER)** addresses the problem of identification and classification of predefined types of named entities such as persons, places, time, numerical and currency expressions.
- **Relation Extraction (RE)** refers to the task of detecting and classifying predefined relationship between the identified entities such as Employees Of (Bill Gates, Microsoft).
- **Event Extraction (EE)** is the task of identifying the events inside a text and concluding detailed and structured information about them. It should handle and recognize the event correctly through the identification of the where, why, whom, what. This method usually tries to recognize the events through understanding the relationship between entities.

## 2.3 Test Collections

Test collections model uses cases in ways that enable researchers to evaluate information retrieval systems [37]. The use of test collections and evaluation measures to determine the information retrieval system validity is important in IR and has been in use since the 1950's.

There exists different test collections for various purposes. Many of these collections have been created as part of TREC [5] (Text REtrieval Conference), a series of experimental

[3] www.wikipedia.org
[4] www.cisco.com
[5] trec.nist.gov

evaluation efforts since 1991, by the U.S. National Institute of Standards and Technology (NIST). What TREC does, is to provide a forum for researchers and different groups from universities, industry, and government to test their IR systems on a broad range of problems.

TREC focuses on common problems and provides a forum for participants to present and discuss their findings. It also facilitates direct inter-system comparison, such as what works, and what is not functioning. TREC targets to create reusable test collections utilized by participants [38].

TREC also inspired others to create similar experiments around the world. These include the European INEX[6] effort for XML retrieval, the CLEF [7] effort for multilingual information retrieval, the Japanese NTCIR[8] effort for Asian Language information retrieval, and also the Indian FIRE[9] effort [39].

## 2.4 Information Retrieval Evaluation

The process of assessing how well a system supports and meets the information required by a user is recognized as the evaluation of the information system. To measure information retrieval effectiveness in the standard way, one uses a test collection encompassing documents, a test suite of information expressible as queries, and a set of relevance judgments. A document in the test collection is either relevant or non-relevant with respect to user informational needs and the decision reached is specified as ground-truth judgments of relevance [9].

Precision and recall are one of the calculation methods in IR evaluation, where recall refers to all the appropriate outputs and precisions to only appropriate outputs of the IR system [35].

We can usually differentiate between two classes of evaluation, the system-based evaluation, and user-based evaluation, where user-based evaluation calculates the user satisfaction with the system while the system-based evaluation measures how well an IR system can rank and retrieve documents [40].

User-based evaluation is much more complicated and extremely expensive to implement correctly. The reason is that it requires a large representation of users of the retrieval system. Each IR system must be fully developed and have useful user interface, and each subject must be well trained on all systems. Such requirements usually leads IR researchers to use the system-based evaluation, which is less expensive and time-consuming [41].

System-based evaluation instead relies on the abstraction of the retrieval process, which represents good performance and good document ranking. The abstraction helps researchers control input and configuration variables that affect retrieval performance. We

[6] inex.mmci.uni-saarland.de/
[7] www.clef-campaign.org/
[8] research.nii.ac.jp/ntcir/
[9] fire.irsi.res.in

call these tests, laboratory examinations; they are much less expensive than user-based evaluations [41].

Tests first began during the Cranfield 2 experiment [42]. The experiment introduced a pattern, which has been used by TREC, CLEF, NTCIR, etc [43].

## 2.5 The Portable Document Format (PDF)

Using the PDF file is, of course, one of the best ways for people to exchange and share data. PDF files are universal, the content of PDF files are displayed similarly in any environment. Processing, recognition, and extraction of texts, tables, images and so forth from PDF files is challenging since most of the PDF documents are untagged and do not have basic logical document structure [44].

PDF file format underlies the Adobe[10] intelligent document platform, facilitating the process of creating digital content on diverse platforms and devices. The goal of the PDF file format is to allow users to exchange and view electronic documents easily and reliably, independent of the environment in which they were created [45].

PDF is a layout-oriented representation format, which means that the human readability is in the foreground, rather than the machine readability. Furthermore, the PDF file format has properties that make it one of the most commonly used formats in all environments [46].

- Font Independence: The biggest problem that other formats are facing is that people usually like to use different fonts, and this can be problematic, as the receiver of the file might not have the font of the sender. PDF's solution provides a technique to save the font descriptor in the file permitting the receiver to view the file in the original font.
- Compression: is one the major properties of PDF format. PDF supports some industrial-standard compression filters such as JPEG, CCITT, and LZW. Compression keeps the file size within limits, which allows an easier exchange for users.
- Portability: PDF files are characterized as a sequence of 8-bit binary bytes or as 7-bit ASCII character code. However, regardless of representation type, the PDF file is stored in binary format and not as text. This capability makes the PDF file extremely portable across assorted hardware and operating systems.

[10] www.adobe.com

CHAPTER 3

# Table Classification and Identification

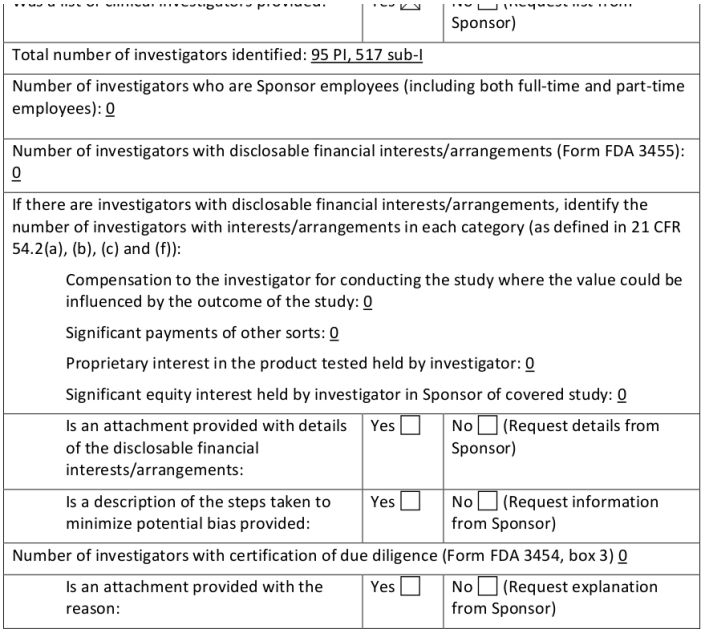
Tables and forms are important segments of the presentation and organization of information in documents. They are widely used to present complex and relevant information. Tables help to recognize information quickly and with less effort. Tables nowadays are an important part of any application and used for various purposes, for example: comparing prices, summarizing business data, showing data from scientific experiments, listing and showing related and unrelated information [47].

A form is a document that is used to collect information based on a predefined template. A form can also be identified and recognized as a particular type of table. In figure 3.1 we see a form that can also be identified as a table [48].

Classification of tables and forms is an important part of the computer-supported recognition of tables and forms. In IR extracting tables is one of the challenges that scientists are working on, and because of a wide variety of usages, it is crucial to understand table types, functions and the purpose of them in documents [49].

Automatic identification of tables in documents is useful in IR as well as other information processing tasks like knowledge extraction, article summarization, data integration, etc.

There are various types of classification approaches; visual table analysis, heuristics-based, and machine learning [50] described herein below:



| | | |
|---|---|---|
| Was a list of clinical investigators provided: | Yes ☑ | No ☐ (Request list from Sponsor) |
| Total number of investigators identified: 95 PI, 517 sub-I | | |
| Number of investigators who are Sponsor employees (including both full-time and part-time employees): 0 | | |
| Number of investigators with disclosable financial interests/arrangements (Form FDA 3455): 0 | | |
| If there are investigators with disclosable financial interests/arrangements, identify the number of investigators with interests/arrangements in each category (as defined in 21 CFR 54.2(a), (b), (c) and (f)):<br>Compensation to the investigator for conducting the study where the value could be influenced by the outcome of the study: 0<br>Significant payments of other sorts: 0<br>Proprietary interest in the product tested held by investigator: 0<br>Significant equity interest held by investigator in Sponsor of covered study: 0 | | |
| Is an attachment provided with details of the disclosable financial interests/arrangements: | Yes ☐ | No ☐ (Request details from Sponsor) |
| Is a description of the steps taken to minimize potential bias provided: | Yes ☐ | No ☐ (Request information from Sponsor) |
| Number of investigators with certification of due diligence (Form FDA 3454, box 3) 0 | | |
| Is an attachment provided with the reason: | Yes ☐ | No ☐ (Request explanation from Sponsor) |

Figure 3.1: Covered Clinical Study form that can be identified as table [48].

## 3.1 Visual table analysis approach

Table recognition by human readers is a visual process based on the existing relationship between the content elements. In this process, alignment among the group of content elements is observed to recognize final tables. Several methods exist that follow the visual process classification approach. Below we describe sparse line detection and robust block segmentation [51].

A table identification method to improve table boundary detection performance by considering the sparse-line property of table rows is proposed in [52] and has the following two steps.
The first step of this method is to find the sparse/non-sparse lines from the document. A document line is sparse if the minimum space gap between a pair of consecutive words within the line is larger than a threshold and that the length of the line is much shorter than a threshold. Sparse line covers document components like tables, mathematical formulas, text in figures, short heading, affiliation, document headers and footers, and references. Non-sparse usually covers document components like titles, abstracts, and paragraphs [52].
The second step of this method is removing the noisy lines and correct detection of sparse lines with feature-based statistical model Conditional Random Field that leads to finding the table boundaries. The performance of this method has a recall above 99% and a precision above 98% [52].

Kieninger proposes an efficient approach to identify tabular structures that focuses on segmentation, block segmentation [53]. The first step is the correct determination of textual units understood as segmentation or structure recognition. The second step is about analyzing geometry arranged blocks entitled structure analysis. The idea is that instead of explicitly looking for separators it identifies words that belong to the same logical unit. The classification method mostly concentrates on effectively identifying the blocks.

## 3.2 Heuristic-based approach

This approach is established on a set of rules that are created to make decisions.

A table classification method is presented in [54]. The search engine in tableseer for identification and extraction of tables uses table metadata to be able to characterize tables occurring in varied and complex documents. The metadata classification categories are 1) table environment/geography (Document-level). 2) table-frame metadata. 3) affiliated table metadata. 4) table-layout metadata 5) table cell-content metadata. 6) table-type metadata.

- Table environment / geography meta data:

  The metadata includes information about the document where the table is located, document type, document page number, document title, document author, document origination, document age and the table starting position. It can facilitate the search by knowing some information about the document.

- Table-frame metadata:

  This metadata consists of information about the boundaries of a table like left, right, top and bottom positions.

- Table affiliation metadata:

  This metadata contains the position of affiliate elements like table caption, table footnote and table reference text.

- Table layout metadata:

  This method captures visualization of a table by saving the table width, table height, number of columns, the number of rows, vertical ruling, horizontal ruling, column width, row length, column header, row headers and horizontal alignments as metadata.

- Table cell content data:

  It refers to values in each cell of a table and enables searching based on cell contents.

- Table cell type metadata:

  It captures the type of a table based on the type of its cells. A cell can be categorized as different types: Numerics, symbols, equations, and text which the method captures as metadata.

Recently, Kim and King presented research and examination on table understanding from the functionality perspective and focus on scientific tables in digital libraries [55]. Table extraction and search engine tools like tableseer [54] and biotext [56] try to solve the problem of accurately extracting tables from documents and filtering the significant results. Kim and Liu suggest categorization of tables based on two aspects: table content materials and table function perspective [57].

Considering the table content material and spatial feature of tables, we can classify them into Background tables, Method/System tables, and experiment tables [55].

- Background tables:

  This classification aims to list and analyze the related studies, provide statistics and data and introduces the paper contribution and implementation agenda to readers. A sample background table is presented in Figure 3.2.

| Conference | No. of Papers | Figures | | Tables | | Algorithms | |
|---|---|---|---|---|---|---|---|
| | | Total | Average | Total | Average | Total | Average |
| SIGIR | 925 | 1990 | 2.15 | 1916 | 2.07 | 75 | 0.08 |
| SIGMOD | 608 | 4303 | 7.08 | 688 | 1.13 | 301 | 0.5 |
| STOC | 406 | 466 | 1.15 | 34 | 0.08 | 74 | 0.18 |
| VLDB | 538 | 5198 | 9.66 | 788 | 1.46 | 287 | 0.53 |
| WWW | 957 | 3735 | 3.9 | 1429 | 1.49 | 142 | 0.15 |

Figure 3.2: A sample background table that provides a statistical information about the distribution of different document-element in different conferences [55].

- Method/System tables:

  These tables are used to discuss the system details, itemise the tentative steps and explain the implementation procedures. A sample system table is shown in Figure 3.3

- Experiment Tables:

  These tables present commentary of experimental result and organized findings, and comparing their results with others. A sample of experiment table is presented in figure 3.4.

Another consideration regarding table classification is the Functional-based classification. Functional-based table classifications are based on the nature of scientific tables [55]. We can categorize this classification into two types:

| Event number | Time | Event |
|---|---|---|
| 1 | 3 | C1 arrives at the attendant |
| 2 | 3 | C1 arrives at CW1 |
| 3 | 8 | C2 arrives at the attendant |
| 4 | 8 | C2 arrives at CW2 |
| 5 | 9 | C3 arrives at the attendant |
| 6 | 11 | C1 leaves car wash |
| 7 | 11 | C3 arrives at CW1 |
| 8 | 14 | C4 arrives at the attendant |
| 9 | 16 | C5 arrives at the attendant |
| 10 | 18 | C2 leaves car wash |
| 11 | 18 | C4 arrives at CW2 |
| 12 | 19 | C3 leaves car wash |
| 13 | 19 | C5 arrives at CW1 |
| 14 | 22 | C6 arrives at the attendant |
| 15 | 27 | C5 leaves car wash |
| 16 | 27 | C6 arrives at CW1 |
| 17 | 28 | C4 leaves car wash |
| 18 | 35 | C6 leaves car wash |

Figure 3.3: A sample method/system table that shows a sequence of events in the car wash [55].

| Sediment | Ni | Cu | Zn | Cd | Pb |
|---|---|---|---|---|---|
| Untreated, pH 7.46 | >3.4 | >3.6 | >3.9 | >2.7 | >2.6 |
| Aerated, pH 7.52 | 3.3 | >3.6 | >3.9 | >2.9 | >2.6 |
| Aerated, pH 5.89 | 2.1 | >3.7 | 1.8 | 1.7 | >2.6 |

Figure 3.4: An example of a table classified as expermiental table [55].

- Commentary tables:

  This type of table is used when the author of the table analyses and compares the contents. An example of comparison table is presented in Figure 3.5.

**Table I.** List of 135 Human Goals Derived From the Psychological Literature

| Abbreviation | Full label |
|---|---|
| Achieving salvation | Achieving salvation |
| Arts | Appreciating the arts |
| Aspirations | Achieving my aspirations |
| Attracting sexually | Being able to attract, please, sexually excite a sexual partner |
| Avoiding failure | Avoiding failure |
| Avoiding guilt | Avoiding feelings of guilt |
| Avoiding rejection | Avoiding rejection by others |

Figure 3.5: An example of a commentary table [55].

- Comparison tables:

  This type of table is used when the author of the table analyses and compares the contents. An example of comparison table is presented in Figure 3.6.

**Table 5** Growth coefficients for perchlorate-degrading microbes at different levels of salinity and ammonium

| Salt | | | | Ammonium | | | |
|---|---|---|---|---|---|---|---|
| Concentration (%) | $\mu$/day$^{-1}$ | $R^2$ | Relative reduction (%) | Concentration (%) | $\mu$/day$^{-1}$ | $R^2$ | Relative reduction (%) |
| 0[a] | 0.168 | 0.68 | 0 | 0[a] | 0.168 | 0.68 | 0 |
| 0.5 | 0.114 | 0.84 | 32 | 0.4 | 0.017 | 0.72 | 90 |
| 1 | 0.103 | 0.86 | 39 | 0.6 | 0.013 | 0.62 | 92 |
| 1.5 | 0.097 | 0.76 | 42 | 1 | 0.016 | 0.53 | 90 |

[a]No additional salt or ammonium was added to the growth medium in the culture tubes. The salt and ammonium concentrations present in the growth medium were <0.02% and <0.08%, respectively.

Figure 3.6: An example of a comparison table [55].

## 3.3 Machine learning approach

In the machine learning classifier domain, different algorithms such as Naive Bayes, Logistic Regression, Decision Tree and Support Vector Machine (SVM) exist.

The classification method presented in [58] identifies the structure and complexity of a table based on the table header using two heuristics while also using machine learning techniques. A table header defined as a line at the top of a table (header row) or the left of the table (header column).

Identifying table header accurately allows the end-user to query a database containing the extracted header data, and to compare the data with others or from other documents. The method proposes two heuristic strategies, local minimum method and machine learning technique, to detect the table header and to separate it from the data part.

**The Local Minimum method** Local Minimum method is based on computer header similarities, the difference between the header row and the data row, and applies a weighted average score to calculate the similarities between each pair of consecutive rows.

**The Machine Learning Technique** The machine learning classifier, the Support Vector Machine (SVM), Logistics regression and Random Forest classifier are utilized for header row detection in tables [58].

In [59] a deep learning method for table detection is presented. This method uses Faster R-CNN [60] (a state-of-the-art object detetion networks) as a basic element of deep network for table identification. The Faster-RCNN computes region proposals itself and then tries to determine whether the selected area is part of a table or not. The experiments using this system show that deep learning based system is robust for table detection as it is not dependent on hand engineered features.

## 3.4 Conclusion

Classification methods described in this section are the visual table analysis, table function or role analysis, and machine learning methods that help extraction tools to have better results.

There is an opportunity to use the table classification outside of the extraction tools to have a pre analysis of tables in documents and, with the help of classification and machine learning, advise the best extraction tool for a document. In this way, we can have overall better results in table extraction methods.

CHAPTER 4

# Table Extraction Systems

The problem related to understanding table has attracted interest from different areas such as database and document engineering communities in the last years.
It is commonly recognized that table recognition and extraction consists of solving three problems [62].

- Locating the regions of a document with tabular content, in short, table detection.
- Reconstruction of the cellular structure of tables, in a nutshell, table structure recognition.
- Rediscovering the meaning of tabular structure, in short, table interpretation.

As mentioned, the aim of the work part of this thesis, is to evaluate the known table extraction tools and to understand how they work, as well as to choose some of them for further experiment within the framework presented in Section 5. We present the following table extraction tools we found in the literature and online commercial sources. We tried to find both available open source and commercial tools.

## 4.1 PDFTOHTML/PDF2HTMLEX

PDFTOHTML [63] is an open source software developed by Gueorgui Ovtcharov and Rainer Dorsch. It can convert pdf files into HTML as long as it is not an image. This tool is one of the very first to perform well in information extraction from PDF files. It is used by other extraction tools to perform additional processes.

Other tools can use the HTML output of PDFTOHTML for further processing. PDF2Table uses the output of PDFTOHTML and identifies tables from the HTML, then extracts this part from the whole HTM file.

There is a newly reworked version of PDFTOHTML named PDF2HtmlEX [63] developed by Lu Wang and Wanmin Liu.
It generates HTML presentation documents using modern web technologies such as HTML5, CSS3, Javascript, in a way that most of the PDF features can be reconstructed. Their method can retain fonts, mathematical formulas, and images and shows correctly in HTML format. PDF2HTMLEX can separate resource files (fonts, CSS, Javascript) from the main HTML file; it reduces the size of the HTML file and improves efficiency.

Text extracted from PDF files is translated to native HTML text element and then put into the same position as in original pdf format using HTML styles. Fonts can also be extracted and converted to Web fonts [64] to have similar font looks as in the PDF file. If the tables are not able to be recognized correctly, they will be extracted as images.

We performed some tests on the PDF2HTMLEX and we observed that in most cases it converts the PDF file into an HTML containing image data.

## 4.2 PDFGenie

PDFGenie is a commercial PDF extraction tool that can extract tables, text, and reading order from PDF files in the form of HTML and XML output [65].

Our tests show that PDFGenie works well with PDF files that contain tables and can correctly generate HTML files with table tags just in case the PDF file contains Tables. A sample HTML in 4.1 shows the extracted table from a PDF document. Our tests also show that PDFGenie has trouble with correctly recognizing words in the older documents.

Listing 4.1: A sample extracted table from a PDF document using PDFGenie

```
...
<table style="font-size:8.96638pt;color:#000000" class="f5"
    bbox="198.24,571.993,385.347,646.38" border="1">
  <tr>
    <th rowspan="1" bbox="198.24,638.112,280.94,646.38"><span
       style="color:#000000">Model</span><span
       style="color:#000000" dir="ltr">P@10</span></th>
    <th rowspan="1" bbox="280.94,638.112,340.4,646.38"
       dir="ltr"><span style="color:#000000">MAP</span></th>
    <th rowspan="1" bbox="340.4,638.112,385.347,646.38"
       dir="ltr"><span style="color:#000000">NDCG</span></th>
  </tr>
  <tr dir="ltr">
    <td rowspan="1"
       bbox="198.24,626.712,280.94,638.112"><span>DEMO</span><span
       style="color:#000000">4040</span><span style="color:#000000"
       dir="ltr">.</span></td>
    <td rowspan="1" bbox="280.94,626.712,340.4,638.112"
       dir="ltr"><span style="color:#000000">0</span><span
```

```
        style="color:#000000" dir="ltr">2666</span><span
        dir="ltr">DEMO</span></td>
    <td rowspan="1" bbox="340.4,626.712,385.347,638.112"><span
        style="color:#000000">0</span><span style="color:#000000"
        dir="ltr">3637</span><span style="color:#000000"
        dir="ltr">.</span></td>
  </tr>
...
</table>
...
```

Another important attribute of PDFGenie is to support the XFDF (XML Forms Data Format) ISO standard [66]. Using XFDF in the same folder as the original PDF file loads an annotated layer above the original PDF that can be used to visually highlight regions in tables. Other structures can be used for easy and quick visual validation of document recognition. It is useful when generating ground-truth, since PDFGenie could generate initial labels for hundreds of documents automatically and later any PDF annotator tool could be used to verify or correct the regions manually.

The main advantage of annotations is that they are decoupled from the content stream and they are easier to manipulate compared to "tagged PDF". There are limitations for using annotation to label PDF regions such that disjointed regions may overlap or nest. PDFGenie, however, uses a workaround for this problem by adding extra properties to annotation dictionaries.

PDFGenie can accept ground-truth XFDF files along with the input PDF and can compute the error rate and other statistics that can be used to evaluate the discrepancy from the optimal output.

PDFGenie accepts as input various arguments. The argument -o is used for specifying the output folder to store the output files. It supports as well argument –pass to specify a password for a PDF file, in case they are password secured. The -x or –xfdf argument is supported as well for generating the XFDF file.

PDFGenie works on major operating systems such as Linux, Mac OSX and Windows. It accepts various parameters for different purposes.

## 4.3 PDFBOX

The Apache PDFBox library is an open source Java tool for working with PDF files. It can extract texts, split a PDF file into many files or merge PDF files into one PDF file. It can also extract or fill in data from PDF forms, validate PDF files against the PDF/A-1b standard [67], save PDF's as image files such as PNG or JPEG, create a PDF from scratch with embedded fonts and images and at last digitally sign a PDF file. It also has the option to extract texts in HTML format.

Figure 4.1 is a sample table with low complexity that is extracted with pdfbox and converted to html. The html result is as follow:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><head><title></title>
<meta http-equiv="Content-Type" content="text/html; charset="UTF-8">
</head>
<body>
<div style="page-break-before:always;
   page-break-after:always"><div><p> 
</p>
<p>Label  2000  2001  2002 
</p>
<p>Freshman  2200  2150  2000 
</p>
<p>Sophomore  2000  2050  2010 
</p>
<p>Junior  1992  1987  2000 
</p>
<p>Senior  1875  1990  1900 
</p>
<p>Graduate  500  475  510 
</p>
<p>Total  8566  8652  8420 
</p>
<p> 
 
 </p>
</div></div>
</body></html>
```

As we can see, the extracted result is hard to use for futher processing by another tools, since there is no specific HTML tags that would represent a table such as <table>.

| Label | 2000 | 2001 | 2002 |
|---|---|---|---|
| Freshman | 2200 | 2150 | 2000 |
| Sophomore | 2000 | 2050 | 2010 |
| Junior | 1992 | 1987 | 2000 |
| Senior | 1875 | 1990 | 1900 |
| Graduate | 500 | 475 | 510 |
| Total | 8566 | 8652 | 8420 |

Figure 4.1: Table extraction with low complexity using PDFBox.

Implementation of an algorithm that identifies keywords in scholarly documents and makes them searchable is part of the CiteSeer X suite in the preprocessing step of this

algorithm in order to extract texts that use PDFBox[1] library [68].

## 4.4 PDF2table

PDF2table uses the pdf extraction tool PDFTOHTML outcome to extract the table information. PDFTOHTML provides for each text chunk in the pdf file a text element in XML with the following attributes:

- top = vertical distance from the top of the page
- left = horizontal distance from the left border of the page
- width = width of the text chunk
- height = height of the text chunk
- font = this attribute describes the size, family, and color of the text chunk

PDF2table tries to decompose a table by utilizing two heuristics groups, the table recognition, and table decomposition. The table recognition heuristic deals with the problem of identifying a table from the XML file. Before running the table recognition algorithms, it first preprocesses the result of PDFTOHTML by sorting all text elements according to their top values. The recognition heuristics first utilizes the information from the sort algorithm and classifies the table based on single-line and multi-line objects, it then detects multi-line objects and merges multi-line block objects that may belong to the same table.

Table decomposition heuristics is the next step after table recognition and deals with the problem of identification of header elements and the assignments of data cells to header elements. It first decomposes the columns of each multi-line block object and then the second algorithm assigns text element to the columns.

This approach also has some limitations and cannot always return correct results. One of the main limitations is that it depends on the outcome of the PDFTOHTML tool, which leads to incorrect extraction whenever PDFTOHTML provides incorrect or inaccurate results. Other limitations include difficulty with identifying tables without labels and tables that are positioned vertically. The xml output of tables generated by the Pdf2table have the following format:

```
<tables>
<table>
<title></title>
<header>
<header_line>
<header_element></header_element>
```

[1] http://pdfbox.apache.org

```
</header_line>
</header>
<tbody>
<data_row></data_row>
<cell></cell>
</tbody>
</table>
</tables>
```

An evaluation sample of a simple table recognition task with PDF is shown in Figure 4.2 [69].
My tests in a linux environment shows that it works in cases that a PDF page contain only one table, however in cases where text and tables are combined in PDF pages, it usually identifies more tables so that this tool recognizes also some part of text as tables.

| | Amounts of samples | Recall | Precision |
|---|---|---|---|
| Lucid tables | 50 | 0.84 | 0.97 |
| Complex tables | 100 | 0.92 | 0.95 |

Figure 4.2: Evaluation result of the table recognition task [69]

## 4.5 Tabula

Tabula [70] is a tool for extracting data tables from the PDF files. Tabula recognizes tables by using position (x and y coordinates) of each character on the page. Tabula gets this data by using the JRuby script that drives the Apache PDFBox java library and allows extraction of data in CSV format, through a simple web interface.

Tabula-java is the java version of Tabula that allows extraction of tables from PDF files using the command-line tool; this is helpful for combining with other tools or software. An important point to mention is that tabula works only with text-based PDF's and not scanned documents.

For tables with ruling lines, Tabula clusters are words that vertically overlap each other. The row boundaries are the bounding boxes of each detected cluster of words. For tables without graphic separators, Tabula runs an analogous procedure to identifying the column boundaries and then it clusters the words that overlap horizontally. The bounding boxes of those clusters are the column boundaries.

Applying Tabula on different tables in PDF files shows that it works well for well-formed tables that lines have separators and can be easily differentiated by this tool. It works best with tables that do not contain rows or column spanning several cells.

Tabula, however, cannot correctly extract tables when tables consist of wrapped lines. Tabula also cannot detect scanned PDF files and only work based on text-based PDF files.

Figure 4.3 is a simple table that is obtained correctly using Tabula, both via the user interface and java command-line tool. The CSV file is as follows:

```
Method,Collection,a,B,g,y,Error
LR linear,INEX,0.97,-60.43,,0.12,0.053
LR log,INEX,-9.12,-2,,9.7,0.011
LR quadratic,INEX,0.97,-209.58,41064.69,0.18,0.022
LR linear cnst.=0,INEX,2.59,-23.4,,0,0.060
LR linear,TREC,1.07,-6.93,,0.13,0.091
LR log,TREC,-6.23,-0.5,,3.43,0.012
LR quadratic,TREC,1.07,-28.03,660.81,0.16,0.041
LR linear cnst.=0,TREC,2.65,-2.69,,0,0.094
```

Table 1: Coefficients derived using linear regression

| Method | Collection | $\alpha$ | $\beta$ | $\delta$ | $\gamma$ | Error |
|---|---|---|---|---|---|---|
| LR linear | INEX | 0.97 | -60.43 | | 0.12 | 0.053 |
| LR log | INEX | -9.12 | -2 | | 9.7 | 0.011 |
| LR quadratic | INEX | 0.97 | -209.58 | 41064.69 | 0.18 | 0.022 |
| LR linear cnst.=0 | INEX | 2.59 | -23.4 | | 0 | 0.060 |
| LR linear | TREC | 1.07 | -6.93 | | 0.13 | 0.091 |
| LR log | TREC | -6.23 | -0.5 | | 3.43 | 0.012 |
| LR quadratic | TREC | 1.07 | -28.03 | 660.81 | 0.16 | 0.041 |
| LR linear cnst.=0 | TREC | 2.65 | -2.69 | | 0 | 0.094 |

Figure 4.3: Simple Table extraction using Tabula [71].

Figure 4.4 is a more complex table that could not be extracted correctly by Tabula via both user interface and java command-line tool. The csv resulted from tabula for Figure 4.4 is as follow:

```
Method,MAP,P@5,P@10,P@20
Method,MAP,P@5,P@10,P@20
LR linear,0.0729,0.355,0.339,0.334
LR log,0.1004,0.397,0.366,0.315
LR quadratic,0.0668,0.389,0.389,0.359
JM,0.0667,0.303,0.245,0.216
LR linear (cv),0.0862,0.331,0.324,0.299
LR linear,0.0286,0.283,0.253,0.213
LR log,0.0633,0.359,0.312,0.273
LR quadratic,0.0654,0.304,0.247,0.222
JM,0.0307,0.214,0.238,0.231
LR linear (cv),0.0355,0.345,0.339,0.333
Odds,0.0800,0.348,0.348,0.323
ZL,0.0780,0.338,0.324,0.307
BM25,0.0063,0.096,0.087,0.070
Odds,0.0572,0.232,0.211,0.191
ZL,0.0611,0.279,0.233,0.228
BM25,0.0844,0.445,0.432,0.352
...
```

Table 2: Retrieval results: empirical smoothing vs. standard retrieval methods (INEX / TREC)

| Method | MAP | P@5 | P@10 | P@20 |
|---|---|---|---|---|
| LR linear | 0.0729 | 0.355 | 0.339 | 0.334 |
| LR log | 0.1004 | 0.397 | 0.366 | 0.315 |
| LR quadratic | 0.0668 | 0.389 | 0.389 | 0.359 |
| JM | 0.0667 | 0.303 | 0.245 | 0.216 |
| LR linear (cv) | 0.0862 | 0.331 | 0.324 | 0.299 |
| Odds | 0.0800 | 0.348 | 0.348 | 0.323 |
| ZL | 0.0780 | 0.338 | 0.324 | 0.307 |
| BM25 | 0.0063 | 0.096 | 0.087 | 0.070 |

| Method | MAP | P@5 | P@10 | P@20 |
|---|---|---|---|---|
| LR linear | 0.0286 | 0.283 | 0.253 | 0.213 |
| LR log | 0.0633 | 0.359 | 0.312 | 0.273 |
| LR quadratic | 0.0654 | 0.304 | 0.247 | 0.222 |
| JM | 0.0307 | 0.214 | 0.238 | 0.231 |
| LR linear (cv) | 0.0355 | 0.345 | 0.339 | 0.333 |
| Odds | 0.0572 | 0.232 | 0.211 | 0.191 |
| ZL | 0.0611 | 0.279 | 0.233 | 0.228 |
| BM25 | 0.0844 | 0.445 | 0.432 | 0.352 |

Figure 4.4: Coimplex Table extraction using Tabula [71].

Tabula is a good choice because of its straightforward interface when extracting simple tables from PDF files and can also be one of the best options for integrating other tools or frameworks using the java library. The output is CSV or JSON which is very helpful for further processing.

## 4.6 Traprange

Traprange [72] is software that functions best with PDF files having a high density of table data.

The table recognition is to find the table data by identifying table column and row. A column can be identified as rectangular that does not overlap with other rectangular spaces of other columns. A row can be identified as words in the same horizontal alignment. If a column has multi-line, then each line will be on a different row. Considering Figure 4.5 the extraction result from the yellow row would be 2 rows:

- 1: 'LedgerID, '', 'Sales Ledger Account', 'FK to this customer's record in'
- 2: NULL, NULL, NULL, 'Ledgers table'

| Table | Customers | | |
|---|---|---|---|
| Field Name | Data Spec. | Caption | Comment |
| ID | I AUTOINC | ID | Primary Key |
| CustType | C(1) | Customer Type | A = cash, R = credit account |
| Status | C(1) | Status | N = normal, C = closed, S = suspended |
| Ledger_ID | I | Sales Ledger Account | FK to this customer's record in Ledgers table |
| AccountCode | C(6) | Account Code | Users-visible unique account number |
| CustName | C(32) | Customer Name | Name of company or organisation |
| CustEmail | C(45) | Main Email Address | |
| Contact | C(32) | Contact Name | Name of main contact person |

Figure 4.5: Trap range[2] sample extraction

The Traprange  method is based on results of the PDFBox API which extracts the text from PDF files. The PDFBox API provides four primary functions:

- PDDocument: provides information about the whole PDF document file.
- PDPage: provides information about each PDF page in the file.
- TextPosition: provides or represents an individual word or character in the document.

With these functions Traprange calculates and processes the words by using the TextPosition function from PDFBox API that returns text parts as shown in Figure 4.6 with the following important attributes:

- x: horizontal distance from the left of the page.
- y: vertical distance from top of the page.
- maxX: equals x + with of the text chunks.
- maxY: equals y + height of the text chunks.

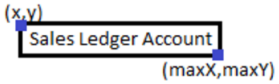



Figure 4.6: Returned text by PDFBox API [72].

The most important part of the Traprange procedure is to detect and identify the bounds of each row and column in the table to extract the content of the table. The process of detection has two attributes.

- LowerBound: contains the lower endpoint of the range.
- UpperBound: contains the upper endpoint of the range.

The traprange process loops through all texts of the page to find and assign the range of each text into horizontal and vertical axis. The second algorithm just loops through all texts again and classifies them into cells of a table.

For evaluation purpose, experiments were done using PDF files with high density tables. The evaluation for extracting tables from PDF files show that it works in some ways

[2]https://dzone.com/articles/traprange-method-extract-table

better than other open source tools like PDF2Table, PDFTOHTML and PDF2TEXT [72].

Traprange can be implemented in other programming languages by replacing PDFBox by a corresponding PDF library to extract text chunks and using this information as input data for both algorithms.

## 4.7 Tableseer

Automatic table extraction and search in digital libraries especially when these libraries contain PDF files is a challenging problem that Tableseer [73] is trying to solve. This system detects tables from PDF documents, extracts table metadata, represents the tables with the metadata file, assigns each table a unique identifier (indexing), and provides a user-friendly search interface to allow users the possibility to search on tables. It has been validated on three aspects based on PDF documents: table detection, table metadata extraction, and table ranking. The ranking method is an innovative algorithm that ranks the tables based on user queries.

The Tableseer architecture consists of essential components: 1) table crawler, 2) table metadata extractor, 3) table metadata indexer, 4) table ranking algorithm, and 5) table searching query interface. The table crawler component in Tableseer crawls online scientific documents from the open-access digital libraries and scientific web pages. The table metadata extractor has three main parts: a text information stripper (TIS) that extracts the words out of the documents based on the position in the pdf file, font size and other information and construct a new TXT file with all the information of the words extracted.

Tableseer, by using a novel page box-cutting method, identifies tables in documents. Tableseer classifies table metadata into six categories and indexes them using the Lucene toolbox for further search and reuse of the data.

1. Table environment and geography metadata that has information of the exact location of the table in the document and as well as general information like document title, author, etc.

2. Table-frame metadata that retrieves the information from the table on whether the table has frames around it.

3. Table affiliation metadata that consists of information about table caption, table caption position and table footnotes.

4. Table layout metadata captures the table visualization information that is for example table width, table length, the number of columns, the number of rows, column width, row length, column headers, row headers.

5. Table cell content data enables the search of tables based on the content of their cells by capturing the information and values in table cells.

6. Table cell type metadata, captures the type of the cell in the table.

TableRank [73] is a new ranking algorithm that returns the matched tables based on a user query in a descendant order according to their relevant scores. The significant difference between TableRank and other modern web search engines is that TableRank rates the <query, table> pairs instead of <query, document> pairs. TableRank determines the final ranking score based on features of the levels 1) the term level, 2) the table level and 3) the document level. Further, it applies each level separately to weight the terms in the vector space and then aggregates them to calculate the final score.

Table detection and extraction were experimented on 200 documents with 397 tables and the results show that Tableseer detected 371 tables. This produces a score of 100% on precision and a recall value of 93.5% [73].

## 4.8 TAO

TAble Organization (TAO) [74] tries to address the problems of table detection and extraction from PDF files. It not only provides automatic table detection and extraction in digital documents but also generates a detailed representation of table elements for structural and functional analyses. The TAO framework is shown in Figure 4.7 and is capable of detecting and recognizing tables within PDF documents. To accomplish this process, TAO combines a supervised machine learning method with layout heuristics and consists of three main modules: 1) document conversion, 2) table detection, and 3) table extraction.

TAO accepts as input a PDF document and using PDFMiner converts the PDF document into an XML format containing all elements in the document. The table detection module parses the XML output from the previous module, identifies the table candidates and stores them. For this TAO performs a structural analysis to determine sets of text boxes that are likely table candidates.

The table detection process consists of four steps: 1) comprehensive identification of text boxes within text groups, 2) distance calculation among text boxes, 3) identification of structural relationships, and 4) finally the generation of table candidates.

The third module, table extraction analyses the table candidates from module 2 and tries to find particular cells and their content by performing table recognition and table composition tasks. Table recognition has the function to identify cells in the table and to detect specific table elements. Table composition performs table separation in the document and classifies elements of the tables as a header that represents a label denoting table information and data, which is the content or body of the table. The table composition task gets all cells on a page and calculates row separation.

The output of the TAO system is a document in JSON format that contains table information extracted from the PDF document. TAO optionally stores the table information in a NoSQL database for easy access and search.

An evaluation process is defined in [74] such that detecting tables means identifying all the tabular elements in a PDF document. One of the experiments performed on CORNELL Dataset 1 and TAO's output has achieved an F1-measure of 87.0% for table detection and 91.1% for table recognition [74].

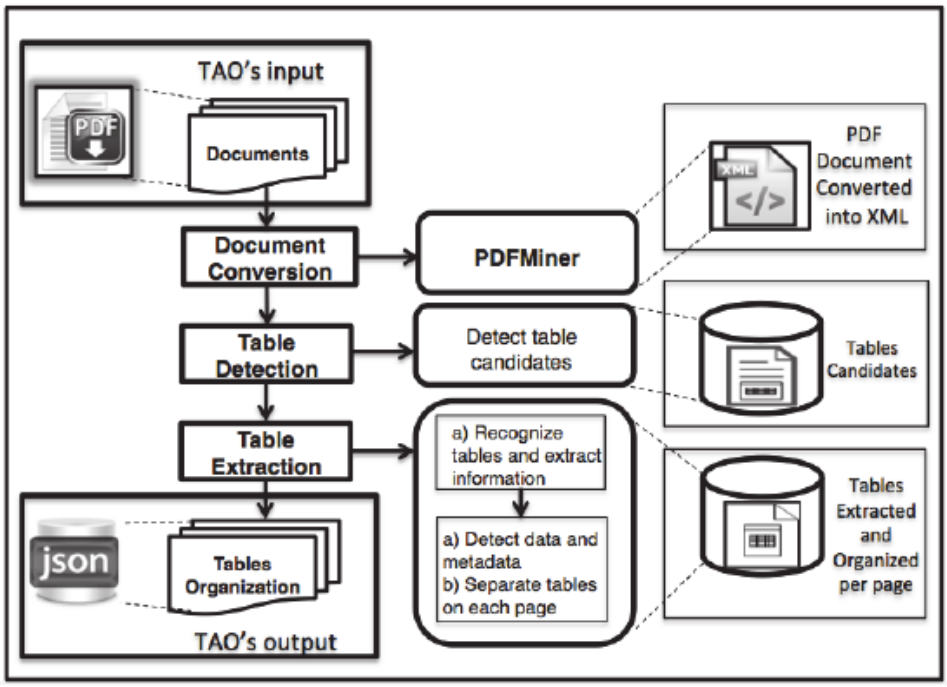



Figure 4.7: The TAble Organization (TAO) System [74].

## 4.9 TINTIN

Table INformation-based Text INquery TINTIN [75] is a system that uses heuristic methods to extract structural elements from the text and separate the tables. TINTIN can index the extracted and retrieved information to allow users the possibility to query and search on the extracted data. This system has been validated on retrieving more than 6,500 tables from the Wall Street Journal database.

The TINTIN architecture is shown in Figure 4.8 and is constructed from a document preprocessing module and an indexing module.

The preprocessing module is based on two main parts.

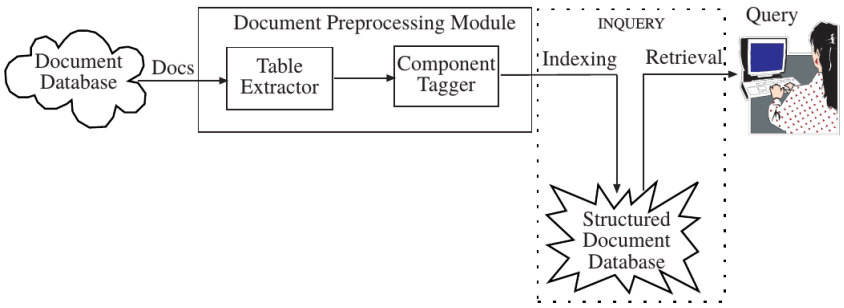



Figure 4.8: TINTIN Architecture [75].

The first part is the table extractor that works on the idea of identifying tables based on aligned white spaces in the document. The underlying data-structure that the extractor relies on is Character Alignment Graph (CAG) also termed holes and gaps. A hole is defined as the number of blanks between columns and gap is defined as blanks between lines. The CAG data-structure is a histogram that stores the number of characters that appear in a particular location in a line. Gaps and holes will be identified within the CAG histogram.

The second part of the preprocessing module is component tagger, which distinguishes between the elements that are the results of the table extractor part such as column headings, table captions, and legends.

Component tagger uses different syntactic heuristics to tag the extracted information. Gap Structure Heuristic identifies table lines by large empty gaps in the middle of the line. Alignment Heuristic identifies table entries by analyzing the gaps and characters of two or more aligned lines in the middle. Pattern Regularity Heuristic tries to find a recurring pattern such that when a row is starting after an identified caption, then it is most probably part of the table.

Differential Column Count Heuristics determine the tables caption by analyzing the first few lines of the table, in case they have fewer columns than average columns in the table then these lines are probably table captions. The Differential Gap Structure Heuristic identifies table captions by analyzing the table lines at the beginning or end of the table. Cases where the alignment does not match the average gap alignment will be marked as the caption. For indexing and storing the extracted information, TINTIN uses the "inbuild" indexing program to index table fields and captions.

TINTIN uses the probabilistic retrieval engine INQUERY to retrieve tables from the database [31]. INQUERY can weigh the query based on fields allowing for more flexibility in the search.

The TINTIN table extractor and component tagging were run on the Wall Street Journal database between 1987 and 1992 and a total of 6509 tables were extracted. The table extractor method missed around 18 lines and 54 incorrect line extraction out of 6205

lines in the 100 documents. The component tagger mistagged 25 lines as table lines and 25 as captions out of 265 lines.

## 4.10 T-Recs

The table recognizer T-Recs [76] is a system that deals with the identification of tables within arbitrary documents, the isolation of individual table cells, and the analysis of the layout to determine a correct row/column mapping. The T-Rec has applied to document images of mixed text/table content and recognition of business letters.

The recognition process works differently from other approaches by using bottom-up clustering of word segments. The T-Recs document model was designed based on three organized structure types and a non-hierarchically embedded structure for text lines.

The first organized structure type is ***Word*** which is the primary object of the system. Formally a word is described as a triple W = (T,G,A), whereas T consists of textual content, G is the bounding box geometry and A holds the captured font attributes.

The second organized structure type is called ***Blocks***. Blocks are a dynamic aggregation of words and made by the central clustering algorithm.

The third and last step of the organized structure type is ***Document***. A document as shown in Figure 4.9 consists of a list of sorted blocks.

***Lines*** are the non-organized structure type and initially built from the aggregation of words. Lines are specified as quadruples L = (W0,succ,G,A), where W0 is a sorted list of words and *succ* is the successor function, G is the bounding box, and A holds the unique line number and logical row number.

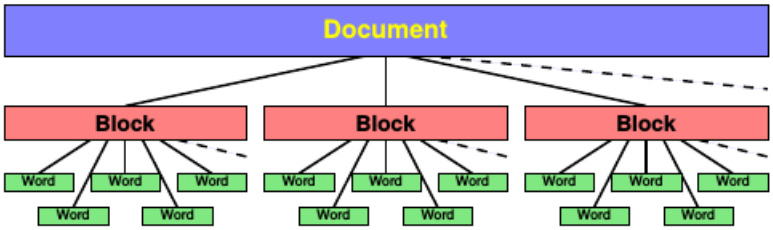



Figure 4.9: Document model of the T-Recs system [76].

Segmentation is characterized in two approaches: the top-down or bottom-up approach. The top-down approach is to detect separators. The idea behind the bottom-up approach is, instead of searching for separators the system tries to find words that belong to the same logical unit. The motivation of this strategy is based on two goals.
First, the top-down systems relies on detection of separators to somehow separate layout regions from each other and this leads to limited applicability.

Second, is the human way of recognizing a tabular structure based on word segmentation and not identifying the text lines then to decompose the layouts and regions.

T-Recs limits its search to nearby words in the lines before and after bounding-boxes which horizontally overlap with the inspected word and uses the so-called segmentation graph to visualize the clustering as shown in Figure 4.10. The nodes in the graph are the center points of the bounding-boxes.


Figure 4.10: Neighbors of the word "consists" [76].

T-Recs is capable of identifying and clustering regular blocks as we can see in 4.11.


Figure 4.11: Segmentation of a table [76]

The first advantage of this method compared to top-down is that it does not dependent on identifying the white spaces that allow accepting any document regardless of table format. Another advantage of this method is that it is not limited to rectangular shapes.

The output of the system is an HTML document that creates the recognized table as HTML table tag "<table>" as shown in Figure 4.12.

| | |
|---|---|
| <TABLE> | Defines the start of a new table |
| <TR> | Defines begin of a new row of a table |
| <TH> | Defines begin of a header cell |
| <TD> | Defines begin of a data cell |
| COLSPAN=$x$ | Optional Parameter for <TD...> and <TH...> that defines a cell to span $x$ columns |
| ROWSPAN=$y$ | Optional Parameter for <TD...> and <TH...> that defines a cell to span $y$ rows |

Figure 4.12: T-Recs HTML output [76].

## 4.11 TARTAR

TARTAR [77] (Transforming ARbitrary TAbles into fRames) is a system that performs the transformation of arbitrary tables (HTML, PDF, EXCEL, etc.) into logical structures, which can be used for automated query answering on 2D-tables.

Tables are usually in unstructured or semi-structured documents, and they can be presented in different formats even when they provide the same content. Tartar tries to handle this problem by automatic transformation of such data into explicit semantics or structured data.

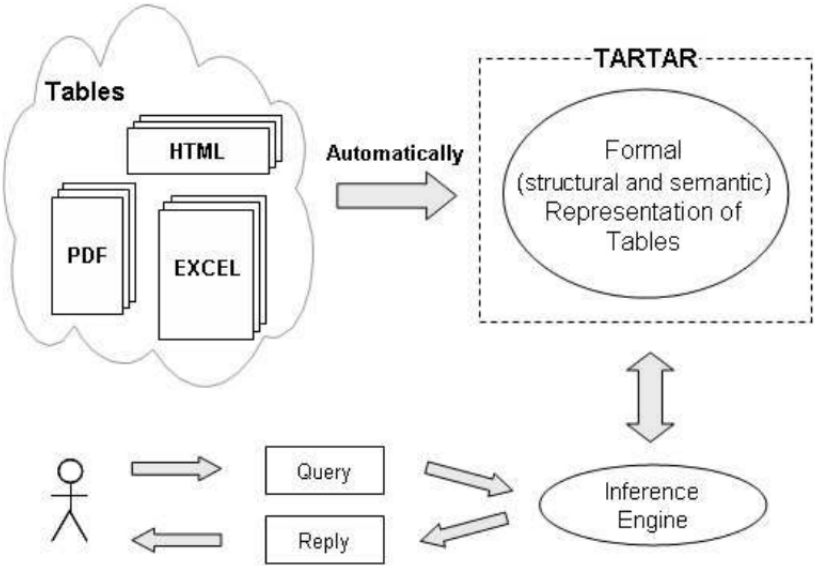



Figure 4.13: Tartar framework and functionality [77].

In Figure 4.13 we see the approach of the TARTAR framework and the problem that it is trying to solve. There is the unstructured or semi-structured information in the form of arbitrary domain related tables that the tartar framework accepts as input. The task of the system is to automatically transform semi-structured information into structured information and then to derive the logical description of input structures.

The goal is to automatically annotate input structures and prepare such data for user queries.

TARTAR transforms arbitrary tables into explicit semantics based on F-Logic frames. F-Logic or Frame Logic [78] integrates features from object-oriented programming, frame-based knowledge representation language and first-order logic.

The overall table transformation is represented in Figure 4.14. This Figure consists of three parts. Part a) represents a table in an arbitrary domain, part b) represents the applying of the TARTAR method especially the F-Logic to the table and part c) shows the two possible queries that could be applied to the result of the part b.

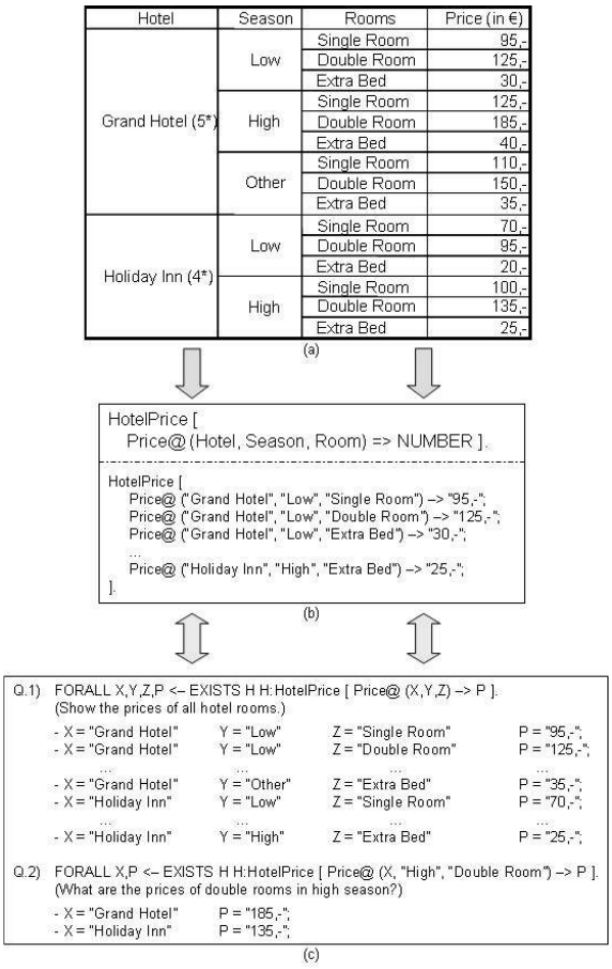



Figure 4.14: Tartar transformation logic in three steps [77].

The implementation approach handles mostly web tables, but can also be expanded to support tables in arbitrary domains. The table recognition is analyzed based on the graphical, physical, structural, functional and semantic aspects. The graphical aspect analyses the image level description of the pixels, lines or other content areas. The physical aspect analyzes the similarity between cells. The structural aspect analyzes the relationship and organization of table cells. The functional aspect studies the purpose of areas regarding data access. The semantic aspect tries to analyze the whole table in the meaning of understanding the relationship between the interpretation of cell content, the meaning of structure in the table and meaning of its reading.

The tartar approach classifies tables for identification in three major layout classes: 1-Dimensional, 2-Dimensional, and complex tables.
The 1-Dimensional table layout usually has a box head (table header) and at least one line body.

The 2-Dimensional table layout has a header, a column description and at least one line of body and the content data is usually rectangular. The complex tables show a variety of layout structures. They are categorized into partition data labels or over-spanning data labels between the data and header labels and over-expanded labels such that entries might be expanded over multiple cells. Another option would be a combination of both variations.
The TARTAR approach has been applied to 158 web tables, which successfully transformed and extracted 135 tables with an 85.44% success rate. TARTAR could successfully process all one-dimensional and two-dimensional tables but had problems with complex tables.

## 4.12 PDF-TREX

Table recognition and extraction (TREX) [51] is a heuristic (experimental) approach for table recognition and extraction from PDF documents. The heuristic works in a bottom-up by applying the algorithm to understand the relationships among the elements of the tables. The benefits of this approach are that it does not require domain knowledge, graphical metadata, and ruling lines or predefined table layouts.

The scope of the approach is to recognize tables contained in a PDF document as 2-dimensional grids on a Cartesian plane and extract them as a set of cells with 2-dimensional coordinates. The heuristic algorithm is founded on eight steps.

Step 1 begins with harvesting the element that identifies the content from the PDF document, store and compute the horizontal and vertical distance threshold values that will be used in the next step.

Step 2, Lines building, which attempts to build lines from the stored content elements. The basic content elements are assigned to lines when their horizontal overlapping ratio is over a given threshold, or all content elements are in a line when their vertical coordinates are on the specified line.

Step 3, segments building and line tagging, which tries to combine the lines to segments. A segment can be of form paragraph or table. This method assigns each line to a cluster then an algorithm collects the clusters until the horizontal distance is lower than an amount hT, and the remaining clusters represents the segments. A line is tagged as a table line when it contains more than one segment and is tagged as text line. When a segment spans over half a horizontal line length, it can be marked as unknown.

Step 4, Table building, aims to analyze the sequence of lines to identify and build table areas.

Step 5, blocks and rows building, seeks to structure table rows. Recognition of the rows in the same algorithm as in step 3 is used; however, the segments are the initial clusters. The algorithm agglomerates these clusters until the vertical distance is lower than vT and the final clusters represent the blocks.

Step 6, column building, aims to create table columns by using the vertical overlapping ratio between the segments and the distance between columns in the table areas. Recognizing the column header spanning multiple columns is possible with this algorithm by assigning segments to the same column when they are overlapped.

Step 7, table building, generates the final tables from the information provided in the previous steps and builds 2-dimensional cells.

Step 8, the extraction step, produces serialized XML table cells from the information gathered from all of the previous steps as the output of this framework. The important part here is that the output can be processed or used for further requirements; for example, understanding table content, storing the data into database, allowing searching and querying.

PDF-trex is applied to 100 PDF documents from different domains containing 164 tables and the precision in table areas is 0.8626 and in table cells is 0.7532. The recall for table areas is 0.9849 and for table cells is 0.9652. It is noteworthy to mention that the recall and precision for table areas are evaluated for the whole document and for table cells the precision and recall is assessed for each table [51].

## 4.13 ABBYY FlexiCapture

Abbyy FlexiCapture, a commercial extraction tool, processes business documents and can extract data from forms [79]. This extraction is executed automatically after the form definition and configuration. After the recognition process, some automatic and manual checking of data is completed before the import into business databases, it uses positions of words and numbers extracted by OCR [80].

For detecting the structure of the document, it uses a method called FlexiLayout and consists of three parts. First, is to recognize elements in the document such as keywords, second, identify the relationship between the objects in the document and third, combine the elements into a group.

The FlexiCapture defines an algorithm to match the description with the document. In the process of matching the FlexiLayout with an image, the first step is to find the objects related to elements in the FlexiLayout, once the necessary information is gathered, the algorithm starts to look for blocks.

FlexiLayout has a tree structure with nodes as its elements. In the matching process, hypotheses generate and form a tree. A hypotheses in this regard is an assumption that the objects detected in the image belong to the elements of the FlexiLayout. After generating the hypotheses, the algorithm selects the best path in hypotheses for an element. In the final step, the hypotheses in the best path is used to locate the blocks described in FlexiLayout.

## 4.14 SmartFix

Smartfix [81], a document extraction tool consisting of two main modules, Document Manager, and Coordinator. The Document Manager is a graphical editor that enables users to select documents and to define document classes to configure the document definition for further processing.
The Coordinator consists of smaller modules such as Analyzer, Matching server and databases. The coordinator controls an importer process that transfers the document images from the documents selected and configured in the Document Manager into the database with all the information that is available to the documents.

Analyzer, with the worker's architecture, is in idle status will check out, process and check-in documents in the database. Finished documents are transferred out of the system by an exporter and with the help of the database, no data is lost during the process. The analyzer is based on Image-processing, Classification, Information extraction, and Improvements stages.

Image-preprocessing step verifies and prepares the image for further processing.

Classification runs on documents to search the following features to determine a class. Features are layout similarity, recognized tables, user-defined or machine-learned pattern, machine-learned semantic similarity, document size and barcode-like patterns.

Information Extraction is achieved according to scripts that are configured in the Document Manager based on the document class.

The Improver module relies on the symbolic AI technique of constraint solving. The main idea is that the fuzzy and unsafe results of scripts in the previous step can be step-by-step constrained to more precise and reliable results based on local auxiliary conditions and interdependency conditions.

SmartFix is a commercial system that is employed by insurance and other companies where it is embedded in their workflow successfully [81].

## 4.15 Conclusion

In this chapter, several extraction tools are analyzed and discussed. From the analyzed tools we can generally identify recognition based on heuristics, Machine learning methods and layout geometry which also work on tables that are scanned and recognized based on font, style, lexical and syntactical information extracted from the textual cell contents. In Table 4.1 we show briefly which methods the table extraction tools use for analyzing and extracting tables from the PDF documents.

Current methods for automatic table detection and extraction in PDF documents are useful but still have some restrictions. These limitations include; tables that span in multiple pages; columns in a table that span in multiple lines; detection relying on

| | Visual Table | Heuristic | Machine Learning |
|---|---|---|---|
| PDF2HTML(EX) | | X | |
| PDFGenie | | X | |
| PDFBOX | | X | |
| PDF2table | | X | |
| Tabula | X | | |
| Traprange | X | | |
| Tableseer | X | | |
| TAO | X | | X |
| TINTIN | | X | |
| T-Recs | X | | |
| TARTAR | X | X | |
| PDF-TREX | X | X | |
| ABBYY FlexiCapture | | X | |
| SmartFix | X | | X |

Table 4.1: Table identification and classification types of the table extraction tools.

patterns not present in all tables; and extraction-lacking elements (such as table lines) needed to reconstruct the information provided in a table accurately.

CHAPTER 5

# Framework Implementation

This work aims to evaluate the known table extraction tools and assess their appropriateness to the IR field. Furthermore, we analyzed whether these tools can recognize and extract tables from the documents correctly.

IR evaluation covers different areas such as information-searching behavior, interface usability, and the computation of IR efficiency. Measuring the effectiveness of an IR system is accomplished by analyzing the results received from the system and their relevance to a user's query and required information.

In order to analyze the performance of a table extraction tool we introduce a framework that provides services that take the output of the various table extraction tools and compares them with a ground-truth. For this purpose, the framework provides a ranking of the table extraction tools.

## 5.1 Framework Overview

The framework has three main sections: preprocessing, validation, and evaluation. The framework overview is shown in Figure 5.1.

In chapter 4 some extraction tools are analyzed and described. From the presented extraction tools, we decided to work with PDFGenie, PDF2table and PDFTOHTML extraction tools for this thesis, because they allow integration with this framework and can correctly extract simple tables from the PDF documents. There were as well other tools with the same attributes, however, in this thesis for demonstration purpose we chose these tools.

PDFGenie extracts the content of the PDF files and transforms it into an HTML file. It works especially well with PDF files that contain simple tables. The important aspect of PDFGenie is that it converts the PDF tables into the HTML table tags. The drawback

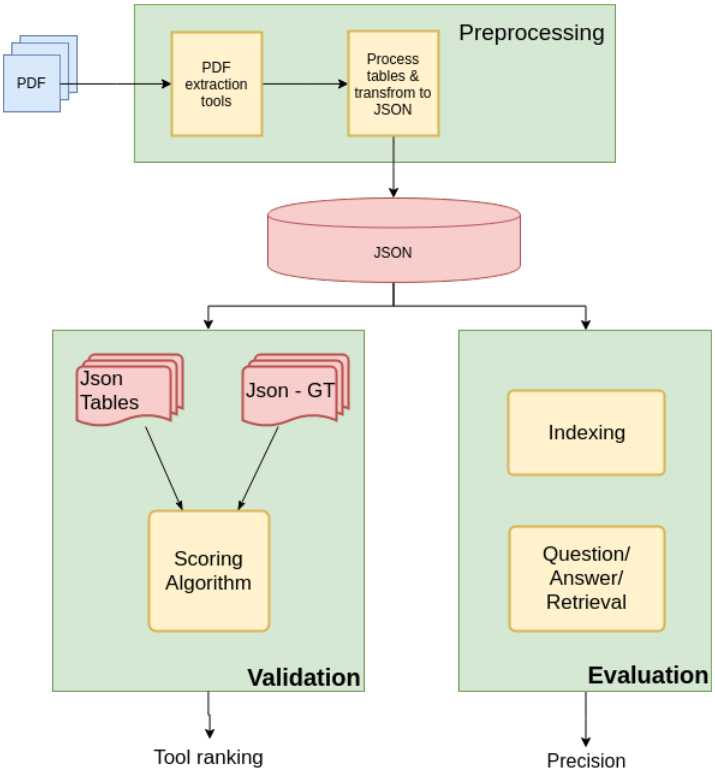



Figure 5.1: Framework overview

is that if the PDF texts are not clear, then it will transform the PDF into an image. PDFGenie also does not work well with older PDF files.

PDF2table transforms the PDF file into an XML document, it then will extract the tables from the XML file. The drawback of this tool is that it cannot extract the header, which is an important part of the table since it could contain valuable information about the table content.

PDFTOHTML extraction tool converts PDF files into an HTML document. The extracted tables in the HTML file do not follow the HTML table annotation (<table>).

The framework starts at the preprocessing step as shown in Figure 5.1, and processes the PDF files with table extraction tools and transforms the results into a common JSON structure and stores them in the database. The next part of the framework is to evaluate the table extraction tools by comparing the tables extracted by them with a previously created ground-truth and calculating a score as illustrated in Figure 5.1 part validation. The third part of the framework is to assess the score result, by indexing all tables with a search engine and perform a question-answering retrieval task to assess the results of the validation part as shown in Figure 5.1 part evaluation.

In the following sections, we will describe each framework part and explain how they

work. The source code of the framework is available in github [1].

## 5.2 Preprocessing and JSON Conversion

The first part of the framework is the preprocessing step. In this step, the PDF files are provided to the PDF table extraction tools to extract the tables from the PDF files. The outcome of these tools are different formats such as XML and HTML. For further processing a so-called standard format for representing the tables is vital. Hence the results of the extraction tools will be processed and transformed into a standard JSON format and stored in a database.

### 5.2.1 Tables in JSON format

JSON stands for "Javascript Object Notation" and is a simple data interchange format. JSON is designed to be a data exchange language, which is human readable and easy for computers to parse and use. Because of the readability and ease of use, JSON is now employed in many different contexts. JSON provides significant performance improvements over XML, since XML requires an extra library to retrieve data from the Document Object Model (DOM) articles.

JSON is built on two structures:

- A collection of name/value pairs realized as objects.
- An ordered list of values or objects realised as arrays.

For expressing the table information in JSON format, we propose the following JSON structure.

- "fileid": the filename of the PDF document containing the table.
- "tableCounter": the number of the table inside the PDF file.
- "header": the description or table header.
- "rows": represents the rows in the table. The column headers are the key of each row and the value representing the content of the table cell.

This pattern allows describing any table data to compare the tables extracted by the extraction tools with the ground-truth. Figure 5.2 shows a table example from one PDF file and following in Code 5.1 the same table in JSON format is presented.

[1] https://github.com/aminmt1362/TEFIR

Code 5.1: Table JSON structure.

```
{
   "fileid": "CLEF2013wn-CHiC-HallEt2013",
   "tablecounter": 2,
      "header": "Table 2. Number of users whose first
         action/first search or browse action were as column
         one",
      "rows": [
         {
            "Action": "Category select",
            "# Users first action": "15",
            "# Users first search/browse action": "20"
         },
         {
            "Action": "Display item",
            "# Users first action": "3",
            "# Users first search/browse action": "-"
         },
         {
            "Action": "Next search result page",
            "# Users first action": "1",
            "# Users first search/browse action": "-"
         },
         {
            "Action": "Add to book-bag",
            "# Users first action": "1",
            "# Users first search/browse action": "-"
         }
      ]
   }
```

Using this JSON structure, we have manually created 40 ground-truth tables that are available on Github for the validation. Furthermore, for IR evaluation, we applied the JSON structure to all tables from the test collection containing 5870 PDF documents by extracting them with the extraction tools and converting them to the JSON documents. PDFGenie extracted 44659 tables and PDF2table extracted 65624 tables. The test collection contains PDF documents from CLEF, NTCIR and TREC conferences from multiple years.

| Action | #Users first action | #Users first search/browse action |
|---|---|---|
| Category select | 15 | 20 |
| Display item | 3 | - |
| Next search result page | 1 | - |
| Add to book-bag | 1 | - |

**Table 2.** Number of users whose first action/first search or browse action were as column one.

Figure 5.2: Sample table, showing the structure of a table.

### 5.2.2 Services for preprocessing component

In this section, the services available for the preprocessing component are described and illustrated in Figure 5.3. The services are established on REST, a lightweight approach to the communication between different components, which perform the processing of the PDFGenie, HTML and PDF2TABLE XML files. These services defined as follow:

- ProcessPdf2Table: accepts a JSON String defining a source path, which is the path of files processed by the PDF2TABLE extraction tool. The process continues by walking through each file and extracting the tables from the XML file and converting those tables to JSON format. Each PDF2Table XML file can contain multiple tables that the processor iterates through the XML tags using XPath queries, and extract the tables.
  After conversion into a JSON document, these documents are introduced into the MongoDB database for further usage.

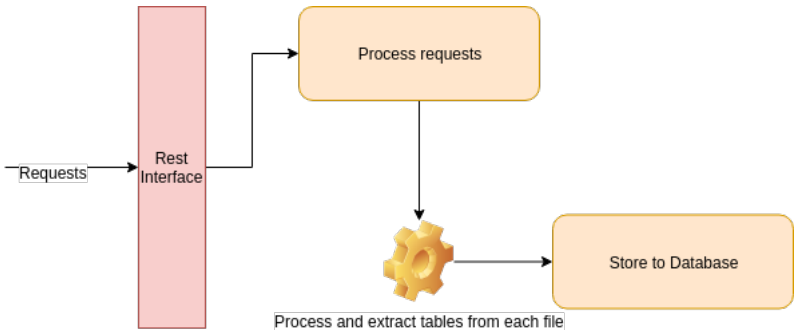



Figure 5.3: Process PDF2Table and PDFGenie table extractors.

- ProcessPdfGenie: accepts a JSON string, which contains the path of PDFGenie processed files. The service walks through each file and processes them. The processing step of the files continues by reading the HTML files to locate the tables inside the HTML file, extract and reconstruct them in JSON document format.
  The next step of the service is to save the created JSON documents in the database allowing further usage without reprocessing the files again.

As mentioned earlier we use three PDF table extraction tools for this thesis, however, we defined only two services, PDFGenie and PDF2TABLE and not PDFTOHTML because PDFTOHTML is only a PDF extraction tool and not a table extraction tool.

## 5.3 Tool Validation

In this section of the framework, we developed methods to receive the outcome of the table extraction tools, and compare those results with the ground-truth presented in Section 5.2.1 and calculate a score to represent a tool ranking result.

The scoring algorithm calculates a score value based on the provided JSON tables via services and the existing ground-truth. The score value is a number between zero and one and help users to assess the table extraction tool.

### 5.3.1 Framework Services for tool validation

In this section, we define the framework services for tool validation and provide a short description.

The services are based on Representational State Transfer.

Following is the list of services provided by the framework for validation purposes.

- ImportTable: this service accepts a JSON String defining the extracted tables in table JSON format as described in the Section 5.2.1. This service stores the tables in a database and allows for further processing. It uses the HTTP POST method to receive the data and responds with a simple text formatted message.
- DeleteUserTable: Removes all imported tables from the framework by deleting the tables from the database. It uses the HTTP DELETE method. The service responds with a simple JSON message containing information, whether all imported tables were removed, if not, it responds with an error in case of failure.
- CalculateScore: Calculates the score for imported tables using the existing ground-truth. The framework imports the ground-truth tables using a predefined path and not via a service and can be configured via parameters in the configuration file. It starts the scoring algorithm, then sends the imported tables into the algorithm for scoring calculation, and finally returns a value that represents the scoring value. This service uses the HTTP GET method to respond to the client with the scoring value.

### 5.3.2 Computing the tool score

Tool Scoring is a subcomponent of the tool validation component that calculates the table extraction tool score. The scoring algorithm calculates a value between zero and one based on the extracted table input and the ground-truth.

In [82] a method for table representation comparison, with the ground-truth presented, these table comparisons have three main characteristics:

- Table regions: a rectangular area in pages. For comparing table regions, the method uses the completeness and purity measurements as presented in [83].
- Cell Structure: the matrix of the cell in a table. For all cells in a region, adjacent relations between cells and their neighbors are generated, which allows comparing of cell structures with the ground-truth.
- Functional model: a set of access relations that allow easy comparison with the ground-truth and reflects the way humans would read a table to lookup information.

The methods described in [82] concentrate on comparing the structure of the extracted tables and can be applied in the implementation of the table extraction tools. This is however not feasible for this thesis since we are more interested in the output given by any table extraction tool.

We use a different approach for comparing the JSON table with the ground-truth. This approach can be applied to the outcome of the table extraction tool without the requirement of implementation inside the tools.

The score calculation of a table ($S_{table}$) consists of four main parts, Cell Scoring ($S_{cell}$), Column scoring ($S_{column}$), Row scoring ($S_{row}$) and Structure scoring ($S_{structure}$). Table score is calculated as

$$S_{table} = a^* \cdot S_{cell} + b^* \cdot S_{column} + c^* \cdot S_{row} + d^* \cdot S_{structure}$$

where $a^*$, $b^*$, $c^*$ and $d^*$ are variables to apply weight on the specific sub-score. These values are calculated based on configurable parameters a, b, c and d where $a^* = \frac{a}{a+b+c+d}$, $b^* = \frac{b}{a+b+c+d}$, $c^* = \frac{c}{a+b+c+d}$ and $d^* = \frac{d}{a+b+c+d}$. The values a, b, c and d can individually be defined by the users.

- Cell Scoring: The cell scoring as shown in Figure 5.4 is based on the comparison of JSON table cells between the extracted tables provided by table extraction tools and the ground-truth tables. The first step is to retrieve the cells from the JSON ground-truth and the cells from the JSON table given by the extraction tool.
  The calculation compares the JSON values (which are cells in the table) of the ground-truth with the values of the table extraction tool. For each mismatch, we subtract one point from the max points $S_{cell_g}$, assuming max points $S_{cell_g}$ are the count of all cells in the ground-truth. The final calculation is shown below where $S_{cell_d}$ is the sum of all differences between the JSON table provided by the table extraction tool and the ground-truth JSON and $S_{cell_g}$ is the count of all cells in ground-truth JSON:

$$S_{cell} = \frac{S_{cell_g} - S_{cell_d}}{S_{cell_g}} \tag{5.1}$$

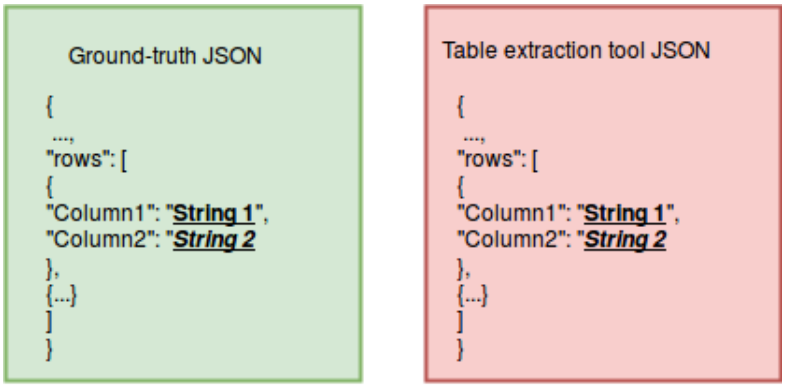



Figure 5.4: JSON Cell comparison.

- Column Scoring: The column scoring algorithm compares the JSON keys of the table extraction tool with the ground-truth as shown in Figure 5.5. The JSON keys are the columns in a table.
  For each column that exists in the ground-truth and not in the extraction tool, we subtract one point out of the sum of columns in the ground-truth $S_{column_g}$. The final calculation is shown below where $S_{column_d}$ is the sum of all differences between the JSON table provided by the table extraction tool and the ground-truth JSON and $S_{column_g}$ is the count of all columns in ground-truth JSON:

$$S_{column} = \frac{S_{column_g} - S_{column_d}}{S_{column_g}} \tag{5.2}$$

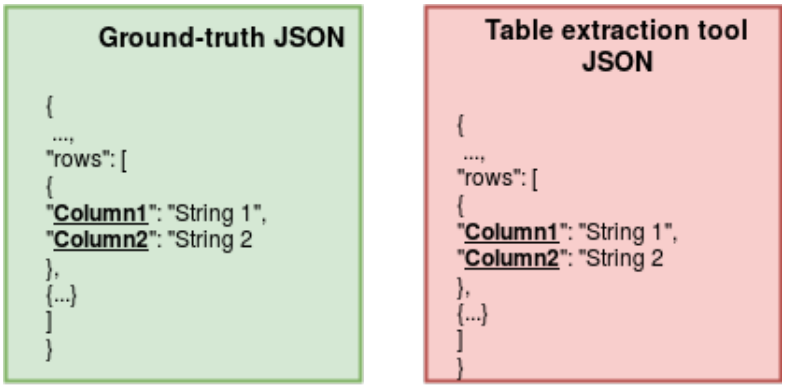



Figure 5.5: JSON Column comparison.

- Row Scoring: The row scoring algorithm compares the rows between the JSON ground-truth tables and the JSON tables provided by table extraction tool as

shown in Figure 5.6. The row scoring algorithm compares the extracted rows from the ground-truth JSON and the extraction tool JSON table by comparing the row based on both the column and cell values. The algorithm subtracts one point out of max points for any discrepancy between the ground-truth table and the table provided by the table extraction tool. The max point $S_{row_g}$ is calculated by counting all rows from the ground-truth. The final calculation is shown below where $S_{row_d}$ is the sum of discrepancies between the JSON table provided by the extraction tools and the ground-truth JSON and $S_{row_g}$ is the count of all rows in the ground-truth JSON.

$$S_{row} = \frac{S_{row_g} - S_{row_d}}{S_{row_g}} \tag{5.3}$$

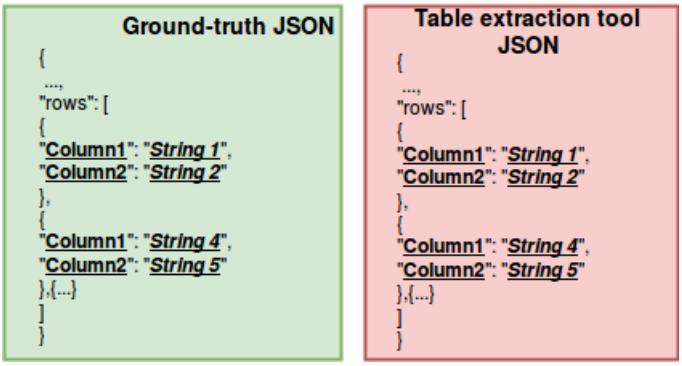



Figure 5.6: JSON Row comparison

- Structure Scoring: The structure scoring algorithm compares the structure of JSON tables between the ground truth and the tables by the extraction tools as shown in Figure 5.7.
  The table in JSON format can be seen as a tree structure so the comparison algorithm checks whether the nodes of the JSON table extraction tool and the ground-truth JSON have the same structure. The algorithm takes each node from the ground-truth and checks whether the same node exists in the provided JSON of the table extraction tool. This process iterates through all nodes of the ground-truth table and applies the same algorithm. This way, the algorithm compares the structures between the ground truth JSON and the extraction tool JSON table and subtracts points for any difference between them. The final score is shown below where $S_{structure_g}$ is the count of all nodes in the ground-truth and the, $S_{structure_d}$ is the sum of discrepancies between the ground-truth and the provided JSON by the table extraction tool.

$$S_{structure} = \frac{S_{structure_g} - S_{structure_d}}{S_{structure_g}} \tag{5.4}$$

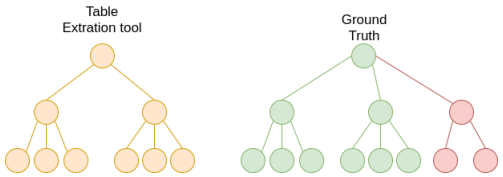



Figure 5.7: JSON Structure comparison

The final tool score value is the sum of all score values calculated for all tables provided by the table extraction tools, divided by the number of tables seen as following where $C_{gt}$ is the count of ground-truth tables and $S_{table}$ is the score value of the tables provided by the table extraction tool.

$$S_f = \frac{\sum_{t=1}^{C_{gt}} S_{table}}{C_{gt}} \tag{5.5}$$

## 5.4 Tool Evaluation

Tool Evaluation describes the process of confirming the ranking of the table extraction tools provided in the Section 5.3.

In this section, we briefly discuss the search engines in general and then describe the configuration we used for this work. Search engines are the key to finding specific information on the vast expanse of the information, whether it is online such as World Wide Web or offline such as private libraries.

A search engine is a program that searches documents for specified keywords and returns a list of documents where the keywords are located. Nowadays, there are several search engines available on the internet, each with their individual abilities and functionalities.

We provide a solution to evaluate the ranking of the table extraction tools. For this, we first explain how we created the questions and relevant answers (qrel). Furthermore, we discuss the search engine and how it was applied to the evaluation. Finally, we introduced services that provided functionality for this process.
Using the provided services, JSON tables from the table extraction tools, will be imported into a search engine for indexing and preparing question-answering processes to evaluate results.

### 5.4.1 Questions and Answers

One way to assess the performance of an information retrieval system is to perform a QA tests in a controlled environment [84].

The general process is similar to the Cranfield-like approach in the adoption of a query topic set and corresponding answers (qrels). We have prepared 25 questions, which are

specified in the annex of this thesis, applied these questions to the specified index in the search engine and evaluated the results the search engine returned.

The process of creating the questions is to analyze the PDF documents and to find questions where answers can be found in different files.

We applied the questions to each index and analyzed the first 20 answers (which we think is an acceptable amount for this work) the search engine produced. For this, we verified whether the results provided by the search engine are relevant to the question by manually verifying the document. In case of relevance, we marked the answer as relevant. After analyzing the results of all questions, we performed the calculation of precision and recall, mean average precision (MAP) and mean repciprocal rank (MRR) using trec_eval tool.

### 5.4.2 Search Engine

The consideration for choosing the appropriate search engine are handling a massive amount of documents, high availability, and scalability. For this thesis, we decided to use Solr.

Solr is a portion of the implementation of the framework for which we provide the following short description: Solr [85] is a modern search engine that is optimized to handle data.

For this thesis we used three pdf table extraction tools, we created three cores (indexes) in Solr. In Solr, the term core is referring to a single index and the associated configuration files and logs. Solr can substantiate more cores at the same time.
Each index in Solr is used for an extraction tool such that we are able to evaluate each extraction tool separately. A document in Solr may contain multiple fields, which describes it.

The method to import the JSON tables extracted by PDFGenie is to convert the JSON documents containing table information into a Solr document as described in the following:

```
{
   "documentid": "CLEF2003wn-adhoc-vanderWeerd2003_1",
   "content": "...",
   "id": "unique_id",
   "_version_": 1568536242577997824
}
```

Where:

- id: is the identifier of the document and is unique for each the document.
- documentid: is the filename of the document specifying that a table belongs to a specific file.

- content: contain the JSON table representation.
- version: is the version of the document.

Importing the documents extracted by PDF2TABLE is similar to PDFGenie. The Solr index document structure of PDF2TABLE is the same as PDFGenie index structure.

For the PDFTOHTML extraction tool, since we want to index the whole HTML file into Solr, we use the following document structure for Solr index.

```
{
   "id":"CLEF2003wn-adhoc-vanderWeerd2003.html",
   "fileLastModified":"Wed May 1717:29:22UTC 2017",
   "link":"/home/amin/Documents/amin/classification/finalex...",
   "text":"...",
   "_version_":1568536242577997824
}
```

Where:

- id: is the identifier of the document and is set to the filename to be able to track the file this document belongs.
- fileLastModified: this is a self-explanatory field which shows the last time the file was modified before importing into Solr.
- link: shows the path from where the document was imported into Solr.
- text: contains the content of the file which is based on what we can search.
- version: is specifying the version of the document.

### 5.4.3 Services and processesses for tool evaluation

In this section, the services required for the evaluation component are described.

The services like other previously defined services are based on REST and are defined as follows:

- PdfGenieSolrImport: the PdfGenieSolrImport service point retrieves the JSON documents, which are already preprocessed with the ProcessPdfGenie, from the database and then using the Solr rest interface these JSON documents are imported into the Solr search engine for the indexing and evaluation process.
- Pdf2TableSolrImport: the Pdf2TableSolrImport service point works almost the same as the PdfGenieSolrImport, the difference is just in getting the PDF2TABLE type of data from the database.

The extracted HTML files by PDF2HTML is imported and indexed in Solr by configuring the path of the files in the Solr configuration.

## 5.5 Conclusion

In this chapter, we described the framework that provides a set of services for processing the tables extracted by the PDF extraction tools and calculate a score that provides a ranked list of the PDF extraction tools. The calculation is based on the comparison of the provided tables by table extraction tools with the ground-truth.

The framework also provides a set of services and a process for evaluating the ranking of the tools.

CHAPTER 6

# Experiments done with the framework

In this chapter, we define the steps of the experiments done with the different parts of the framework to calculate the score for the table extraction tool and the generating the ranked list of these tools. Furthermore, we evaluate the pdf extraction tools and assess their application to the IR field.

We divide the experiment into two parts. The first part is the validation step where we experiment with the Preprocessing and Validation component and show the results we achieved. The second part is the evaluation step where we experiment with the Preprocessing and Evaluation part of the framework and provide the results of the experiment.

## 6.1 Calculate the ranking of the tools

In this section, we experiment using the framework validation component presented in the previous chapter and provide the ranking of the PDF table extraction tools. We divide the process into preprocessing step and calculating the ranking list of the tools.

### 6.1.1 Preprocessing step

The Preprocessing component is the starting point of the process. We process the 40 tables contained in the PDF files that are selected for the ground-truth purpose with the PDF2TABLE and PDFGenie extraction tools and store the HTML and XML files.

The next step is to convert the tables inside the HTML and XML files into JSON using the service “ProcessPdf2Table” and import the tables into the MongoDB database.

The process for PDFGenie is similar to PDF2TABLE, by calling the "ProcessPdfGenie" service that extracts the tables from the HTML files, then converts the tables to JSON and imports them into the database.

### 6.1.2 Score calculation

For calculating the score of the extraction tools, we call the "CalculatePdf2TableScore" service. The service retrieves the PDF2TABLE JSON tables from database and then calls the "ImportTable" service from the validation component. After importing all JSON tables, the service continues by calling the "CalculateScore" service from the validation component to calculate the score as described in Section 5.3.2 and then returns the score value of PDF2TABLE. Our setting for parameters a,b,c and d were the value one for every parameter.

We continue the similar process for the PDFGenie extraction tool by calling the "CalculatePdfGenieScore" service that does the same procedure as for PDF2TABLE, but with the PDFGenie JSON tables.

The calculated score values of the PDFGenie and PDF2TABLE extraction tools provide a ranking of these tools as shown in the Table 6.1. The ranking list show that the PDF2TABLE has a much higher score in extracting the tables from the PDF files than the PDFGenie.

| Extraction tool | Score |
|---|---|
| PDFGenie | 0.498856 |
| PDF2TABLE | 0.907667 |

Table 6.1: The ranking of the PDFGenie and PDF2Table extraction tools.

We did not calculate the score of the PDF2HTML extraction tool because this tool cannot extract and convert the tables from the PDF files.

## 6.2 Assessing the ranking of the tools using IR evaluation

In this section, we perform an experiment using the evaluation component of the framework presented in the previous chapter and show the values obtained. The preprocessing step is similar to the preprocessing step in Validation part described in Section 6.1.1 with the difference that we process 5870 PDF files.

### 6.2.1 Evaluation process

In this step, we take the JSON tables from the 5870 PDF files processed by PDF2TABLE and PDFGenie extraction tools from database and index those in the search engine Solr using the "Importtosolr" service as described in Section 5.4.

The HTML files of PDFTOHTML extraction tools import information into Solr for indexing without using any service since it is configured to take the files in the specified path using the Solr configuration file.

We used the trec_eval tool to calculate scores for IR evaluation. For this, we created a ground-truth file formatted for the trec_eval. Furthermore, we created a script to generate an input file for the trec_eval containing the query number and the score we received from the Solr.

We further merge the results of the PDFTOHTML with PDFGenie and the results of PDFTOHTML with PDF2TABLE since PDFTOHTML extracts the whole PDF file and PDF2TABLE and PDFGenie extract only tables. This approach helps us to verify whether we could yield better overall results by having merged values.

There exist several merging algorithms such as raw-score merging, round-robin merging and normalized-score merging [86].
The raw-score or simple method sorts all results by their original similarity scores and then selects the top-ranked documents. This method assumes that the similarity scores across collections are comparable.
The round-robin method inserts the results based on the rank by assuming that each collection has about the same number of relevant documents and the distribution of relevant documents is similar among the result lists.
The normalized-score method suggests that for each topic, similarity scores of each document is divided by the maximum score in a topic after adjusting the scores, all results are placed into the same pool and sorted by the normalized score. This approach makes the results more comparable by mapping the similarity scores of different result lists into the same range, from zero to one.

We use the raw-score merging or simple merging method since the results are comparable because they are from the same test collection and we use the same engine and retrieval method behind all three indexes PDFTOHTML, PDFGenie and PDF2TABLE. We merge the results of the PDFTOHTML with the PDFGenie and name it merge-1, the results of the PDFTOHTML with the PDF2TABLE we designate merge-2.

In the Table 6.2 we show the calculated scores of the IR evaluation. We calculated the P@10, R@10, mean average precision (MAP) and mean reciprocal rank (MRR) for each tool and as well for merge-1 and merge-2.

As indicated, the overall score values for PDFGenie is higher than PDF2TABLE, and PDF2HTML has the highest values. This shows that the calculated ranking presented in Table 6.1 is not validated.

We see that the overall IR evaluation score for PDFGenie is higher than PDF2TABLE and this is opposite to the calculated ranking scores in the previous step. We can also observe that the PDF2HTML has the highest IR evaluation scores among other table extraction tools.

| | P@10 | R@10 | MAP | MRR |
|---|---|---|---|---|
| PDF2HTML | 0.2400 | 0.3582 | 0.2158 | 0.3522 |
| PDF2TABLE | 0.1640 | 0.3544 | 0.2717 | 0.3151 |
| PDFGenie | 0.1800 | 0.3674 | 0.2652 | 0.3255 |
| merge-1 | 0.2400 | 0.3582 | 0.2158 | 0.3522 |
| merge-2 | 0.2400 | 0.3582 | 0.2158 | 0.3522 |

Table 6.2: The Precision, recall, MAP and MRR calculation of PDF2HTML, PDF2TABLE, PDFGenie, merge-1 and merge-2 obtained using Solr and trec_eval tool.

Since the overall scores of PDF2HTML are better than the other table extraction tools, it shows that extracting only tables and putting it in an IR tool is not enough. While the tables contain rich data about a topic, the accuracy of IR tools to retrieve the correct table based on the asked question is low. One reason is that the table extraction tools could not extract the table's caption hence, the question keywords might not be in the data inside the table.
Even when the table extraction tools could extract the table's caption, the IR retrieval accuracy might still be low. This is because the user's queries to the IR tools are usually very broad and the query keywords might not contain the caption of the table.

## 6.3 Conclusion

In this chapter, we experimented with the implemented framework. We used the provided services of the framework to generate the JSON tables of the tables extracted by the extraction tools. Furthermore, we imported the JSON table documents into the framework to calculate a score and generate a ranking list. For evaluation, we employed the services of the framework to import the tables into the search engine, Solr, and then applied the already created topics verifying the outcome to obtain the precision, recall, MAP and MRR.

There are two important points to note; first, the scoring algorithm does not compare the table caption. This could be an important factor for future research, and in this regard, we assume that PDFGenie would yield better results since it can extract table captions correctly. The PDF2TABLE cannot extract the table headers from the PDF files in the latest version we used.

Second, as indicated from the results of the table extraction tools ranking 6.1, the margin between the tools is too large. The reason is that the PDFGenie extraction tool that we used for this thesis is the free trial version. The trial version has the drawback that cannot extract all words correctly by making deliberate mistakes such as replacing the actual words with "DEMO". The detection accuracy of PDFGenie is also much lower than PDF2Table in the older PDF file formats.

CHAPTER 7

# Summary

To evaluate the known table extraction tools and assess their appropriateness to the IR field, we developed a framework that consists of two parts, validation and evaluation. The validation part provides services for importing the tables into the framework and calculate score, then generate a ranking list of the table extraction tools. The evaluation part provides services for importing the tables into the search engine Solr. We further developed an algorithm to convert the extracted tables from the PDF documents by the table extraction tools into JSON format.

We further experimented with the framework using both parts. For the first part, we used the services to import the 40 tables that we used for ground-truth tables, into the framework and generated a ranking of these tools. For the second part, we experimented with the evaluation part and used the provided services to import the tables extracted by each tool into the search engine Solr. We further generated a ground-truth containing 25 questions and applied the questions to the search engine and evaluated the outcome. We calculated the precision, recall, MAP and MRR for each table extraction tool.

A question that arises after this research and experimentation is whether it is enough to extract only the tables from the PDF documents to support users search requests. Since user's questions can be very general and the tables usually have more numbers than text, hence finding the query keywords by IR tools in the tables can be low.

# Appendix: Evaluation Questions

Question 1: What are the MAP values for UNINE's runs in the CHIC 2013 Polish task?

Solr: MAP UNINE's runs CHIC AND 2013 Polish task.

Question 2: Which parameter settings were used by UNINE submissions at the CHIC 2013 Polish task?

Solr: parameter settings UNINE submissions CHIC AND 2013 Polish task.

Question 3: What are the precision values obtained by the Berkley team 2000 for their multilingual runs?

Solr: precision Berkley team 2000 multilingual runs.

Question 4: Which are the language pairs used in multilingual experiments in CLEF 2001?

Solr: language pairs multilingual experiments in CLEF AND 2001.

Question 5: Which size does the CLEF 2001 collection have per language in the collection?

Solr: size CLEF AND 2001 collection language.

Question 6: What precision values were obtained for the Dutch monolingual task in CLEF 2001?

Solr: precision values obtained Dutch monolingual task in CLEF AND 2001.

Question 7: Wich retrieval systems/methods were used in the Spanish monolingual tasks?

Solr: Which retrieval systems/methods were used in the Spanish monolingual tasks.

Question 8: Which groups of institutions submitted to the CLEF-IP lab in 2009?

Solr: Which groups of institutions submitted to the CLEF-IP AND 2009.

Question 9: What scores were obtained for the AUC metric in the Image Clef photo annotation task?

Solr: What scores were obtained AUC metric Image AND Clef photo annotation task.

Question 10: which AUC scores did the Xerox team obtain in the image classification task, 2011?

Solr: which AUC scores did the Xerox team obtain in the image AND classification task 2011.

Question 11: Which document sets were used for the Patent Mining task in NTCIR?

Solr: Which document sets were used for the Patent Mining task AND NTCIR.

Question 12: what are the comparison performance results of using Indri-word-query with and without wheighting and expansion IN NTCIR?

Solr: comparison performance results Indri-word-query with and without wheighting and expansion AND NTCIR.

Question 13: what are the precision and recall values for identifying opinionated sentences under different runs in NTCIR?

Solr: NTCIR precision recall values identifying opinionated sentences runs.

Question 14: What are the subtonic mining subtask run?

Solr: subtonic mining subtask run.

Question 15: What are the percentage values of source n-grams in the training set for NTCIR and Gale evaluation?

Solr: percentage source n-grams training set for NTCIR and Gale evaluation.

Question 16: What are the development tuning results and the tuning methods in NTCIR patent?

Solr: NTCIR AND patent development tuning results tuning.

Question 17: What are the performance results using data fusion in ad-hoc task?

Solr: data fusion ad-hoc performance.

Question 18: What are the precision and recall results of ruibsl and runit1/q1 sets?

Solr: precision recall performance ruibsl runit1/q1.

Question 19: What are the average precision of topics vs R5(FBIS) and R4?

Solr: average precision + topic + R5 + R4.

Question 20: What are the evaluation results of DEMIR group in ImageCLEF 2001 medical image context?

Solr: DEMIR ImageCLEF 2001 medical image context.

Question 21: What are the MAP values for the patent task retrieval track of CLEF-IP 2012?

Solr: MAP patent task retrieval track CLEF-IP 2012.

Question 22: What systems are used in the vertical selection of TREC 2014 FedWeb track?

Solr: systems vertical selection TREC AND 2014 FedWeb track.

Question 23: What are the comparison results of ICTNET run1 and ICTNET run2?

Solr: comparison results ICTNET run1 AND ICTNET run2.

Question 24: What are the 6 MRR measures sorted by WGT?

Solr: 6 MRR WGT + measures + sorted.

Question 25: What are the MAP values of BUAA AUDR group participation in photo annotation retrieval tasks at ImageCLEF 2012?

Solr: BUAA AUDR ImageCLEF AND 2012 photo annotation retrieval tasks.

# List of Figures

# List of Tables

# Bibliography


[1] M. Ware, "Stm report 2015 final 2015-02-20," http://www.stm-assoc.org/2015_02_20_STM_Report_2015.pdf, March 2015, (Accessed on 02/05/2018).

[2] S. Robertson, "On the history of evaluation in ir," *Journal of Information Science*, vol. 34, no. 4, pp. 439–456, 2008. [Online]. Available: https://doi.org/10.1177/0165551507086989

[3] S. Ceri, A. Bozzon, M. Brambilla, E. D. Valle, P. Fraternali, and S. Quarteroni, *Web Information Retrieval*. Springer Berlin Heidelberg, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-39314-3

[4] (2016) The text retrieval conference. [Online]. Available: https://dzone.com/articles/traprange-method-extract-table

[5] (2016) NII testbeds and community for information access research. [Online]. Available: http://ntcir.nii.ac.jp/about/

[6] (2016) The CLEF initiative - conference and labs of the evaluation forum. [Online]. Available: http://www.clef-initiative.eu/web/clef-initiative/home

[7] (2016) International conference on document analysis and recognition. [Online]. Available: http://2015.icdar.org/program/competitions/,urldate={2016-10-07}

[8] A. M. Cohen, C. E. Adams, J. M. Davis, C. Yu, P. S. Yu, W. Meng, L. Duggan, M. McDonagh, and N. R. Smalheiser, "Evidence-based medicine, the essential role of systematic reviews, and the need for automated text mining tools," in *Proceedings of the 1st ACM International Health Informatics Symposium*, ser. IHI '10. New York, NY, USA: ACM, 2010, pp. 376–380. [Online]. Available: http://doi.acm.org/10.1145/1882992.1883046

[9] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[10] S. H. Myaeng, D.-H. Jang, M.-S. Kim, and Z.-C. Zhoo, "A flexible model for retrieval of sgml documents," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser.

SIGIR '98. New York, NY, USA: ACM, 1998, pp. 138–145. [Online]. Available: http://doi.acm.org/10.1145/290941.290980

[11] S. Yin, Y. Qiu, J. Ge, and F. Wang, "A chinese text classification approach based on semantic web," in *2008 Fourth International Conference on Semantics, Knowledge and Grid*, Dec 2008, pp. 497–498.

[12] F. S. Gharehchopogh and Z. A. Khalifelu, "Analysis and evaluation of unstructured data: text mining versus natural language processing," in *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*, Oct 2011, pp. 1–4.

[13] D. Smith and M. Lopez, "Information extraction for semi-structured documents," in *In Proceedings of the Workshop on Management of Semistructured Data*, 1997.

[14] M.-F. Sy, S. Ranwez, J. Montmain, A. Regnault, M. Crampes, and V. Ranwez, "User centered and ontology based information retrieval system for life sciences," *BMC Bioinformatics*, vol. 13, no. 1, p. S4, 2012. [Online]. Available: http://dx.doi.org/10.1186/1471-2105-13-S1-S4

[15] R. R. Larson, "Information retrieval: Searching in the 21st century; human information retrieval," *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, no. 11, pp. 2370–2372, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1002/asi.v61:11

[16] H. Kaur and V. Gupta, "Indexing process insight and evaluation," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, Aug 2016, pp. 1–5.

[17] R. Sagayam, S. Srinivasan, and S. Roshni, "A survey of text mining: Retrieval, extraction and indexing techniques," *International Journal of Computational Engineering Research*, vol. 2, no. 5, 2012.

[18] S. McCartney, *ENIAC: The triumphs and tragedies of the world's first computer*. Walker & Company, 1999.

[19] M. Taube, C. D. Gull, and I. S. Wachtel, "Unit terms in coordinate indexing," *American Documentation*, vol. 3, no. 4, pp. 213–218, 1952. [Online]. Available: http://dx.doi.org/10.1002/asi.5090030404

[20] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM J. Res. Dev.*, vol. 1, no. 4, pp. 309–317, Oct. 1957. [Online]. Available: http://dx.doi.org/10.1147/rd.14.0309

[21] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc., 1986.

[22] K. Sparck Jones, "Document retrieval systems," P. Willett, Ed. London, UK, UK: Taylor Graham Publishing, 1988, ch. A Statistical Interpretation of Term Specificity and Its Application in Retrieval, pp. 132–142. [Online]. Available: http://dl.acm.org/citation.cfm?id=106765.106782

[23] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009. [Online]. Available: http://dx.doi.org/10.1561/1500000019

[24] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Trans. Inf. Syst.*, vol. 7, no. 3, pp. 183–204, Jul. 1989. [Online]. Available: http://doi.acm.org/10.1145/65943.65944

[25] M. Sanderson and W. B. Croft, "The history of information retrieval research," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1444–1451, May 2012.

[26] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins, "Stuff i've seen: A system for personal information retrieval and re-use," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03. New York, NY, USA: ACM, 2003, pp. 72–79. [Online]. Available: http://doi.acm.org/10.1145/860435.860451

[27] M. Sharma and R. Patel, "A survey on information retrieval models, techniques and applications," *International Journal of Emerging Technology and Advanced Engineering, ISSN*, pp. 2250–2459, 2013.

[28] D. L. Lee, H. Chuang, and K. Seamons, "Document ranking and the vector-space model," *IEEE Software*, vol. 14, no. 2, pp. 67–75, Mar 1997.

[29] D. Hiemstra and A. de Vries, *Relating the new language models of information retrieval to the traditional retrieval models*, ser. CTIT Technical report series, 6 2000, vol. 00, no. 00-09, imported from CTIT.

[30] H. Turtle and W. B. Croft, "Evaluation of an inference network-based retrieval model," *ACM Trans. Inf. Syst.*, vol. 9, no. 3, pp. 187–222, Jul. 1991. [Online]. Available: http://doi.acm.org/10.1145/125187.125188

[31] J. Callan, W. B. Croft, and S. M. Harding, "The inquery retrieval system," in *In Proceedings of the Third International Conference on Database and Expert Systems Applications*. Springer-Verlag, pp. 78–83.

[32] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft, "Indri: A language model-based search engine for complex queries," in *Proceedings of the International Conference on Intelligent Analysis*, vol. 2, no. 6. Amherst, MA, USA, 2005, pp. 2–6.

[33] E. Greengrass, "Information retrieval: A survey," 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.1855

[34] M. McCandless, E. Hatcher, and O. Gospodnetic, *Lucene in Action, Second Edition: Covers Apache Lucene 3.0.* Greenwich, CT, USA: Manning Publications Co., 2010.

[35] J. Piskorski and R. Yangarber, "Information extraction: Past, present and future," in *Multi-source, Multilingual Information Extraction and Summarization*, ser. Theory and Applications of Natural Language Processing, T. Poibeau, H. Saggion, J. Piskorski, and R. Yangarber, Eds. Springer Berlin Heidelberg, 2013, pp. 23–49. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28569-1_2

[36] J. Cowie and W. Lehnert, "Information extraction," *Commun. ACM*, vol. 39, no. 1, pp. 80–91, Jan. 1996. [Online]. Available: http://doi.acm.org/10.1145/234173.234209

[37] S. Büttcher, C. Clarke, and G. V. Cormack, *Information Retrieval: Implementing and Evaluating Search Engines.* The MIT Press, 2010.

[38] E. M. Voorhees and D. M. Tice, "Building a question answering test collection," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '00. New York, NY, USA: ACM, 2000, pp. 200–207. [Online]. Available: http://doi.acm.org/10.1145/345508.345577

[39] M. Sanderson, "Test collection based evaluation of information retrieval systems," *Foundations and Trends® in Information Retrieval*, vol. 4, no. 4, pp. 247–375, 2010. [Online]. Available: http://dx.doi.org/10.1561/1500000009

[40] C. Peters, *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, CLEF 2001, Darmstadt, Germany, September 3-4, 2001. Revised Papers.* Springer Science & Business Media, 2002, no. 2406.

[41] E. M. Voorhees, *The Philosophy of Information Retrieval Evaluation.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 355–370. [Online]. Available: http://dx.doi.org/10.1007/3-540-45691-0_34

[42] C. Cleverdon, "Readings in information retrieval," K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ch. The Cranfield Tests on Index Language Devices, pp. 47–59. [Online]. Available: http://dl.acm.org/citation.cfm?id=275537.275544

[43] C. CLEVERDON, "The cranfield tests on index language devices," *Aslib Proceedings*, vol. 19, no. 6, pp. 173–194, 1967. [Online]. Available: https://doi.org/10.1108/eb050097

[44] H. Chao and J. Fan, "Layout and content extraction for pdf documents," in *International Workshop on Document Analysis Systems.* Springer, 2004, pp. 213–224.

[45] "Acrobat dc sdk documentation," https://help.adobe.com/en_US/acrobat/acrobat_dc_sdk/2015/HTMLHelp/index.html#t=Acro12_MasterBook%2FJS_Dev_Overview%2FOverview.htm, (Accessed on 12/08/2017).

[46] A. S. Incorporated, "PDF Reference - Adobe Portable Document Format, 5th edition, http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf," 2006. [Online]. Available: http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf

[47] R. Rastan, "Towards generic framework for tabular data extraction and management in documents," in *Proceedings of the sixth workshop on Ph. D. students in information and knowledge management*. ACM, 2013, pp. 3–10.

[48] D. P. Lopresti and G. Nagy, "A tabular survey of automated table processing," in *Selected Papers from the Third International Workshop on Graphics Recognition, Recent Advances*, ser. GREC '99. London, UK, UK: Springer-Verlag, 2000, pp. 93–120. [Online]. Available: http://dl.acm.org/citation.cfm?id=645438.652758

[49] S. Mandal, S. P. Chowdhury, A. K. Das, and B. Chanda, "A hierarchical method for automated identification and segmentation of forms," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ser. ICDAR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 705–709. [Online]. Available: http://dx.doi.org/10.1109/ICDAR.2005.17

[50] Y. Wang, M. Haralick, R. M. Haralick, and I. T. Phillips, "Document analysis: table structure understanding and zone content classification," 2002.

[51] E. Oro and M. Ruffolo, "Pdf-trex: An approach for recognizing and extracting tables from pdf documents," in *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ser. ICDAR '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 906–910. [Online]. Available: http://dx.doi.org/10.1109/ICDAR.2009.12

[52] Y. Liu, P. Mitra, and C. L. Giles, "Identifying table boundaries in digital documents via sparse line detection," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, ser. CIKM '08. New York, NY, USA: ACM, 2008, pp. 1311–1320. [Online]. Available: http://doi.acm.org/10.1145/1458082.1458255

[53] T. G. Kieninger, "Table structure recognition based on robust block segmentation," in *Document Recognition V*, vol. 3305. International Society for Optics and Photonics, 1998, pp. 22–33.

[54] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, "Tableseer: Automatic table metadata extraction and searching in digital libraries," in *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*, ser. JCDL '07. New York, NY, USA: ACM, 2007, pp. 91–100. [Online]. Available: http://doi.acm.org/10.1145/1255175.1255193

[55] S. Kim and Y. Liu, "Functional-based table category identification in digital library," *2011 International Conference on Document Analysis and Recognition*, 2011; doi: 10.1109/ICDAR.2011.274. [Online]. Available: https://doi.org/10.1109/icdar.2011.274

[56] M. A. Hearst, A. Divoli, H. Guturu, A. Ksikes, P. Nakov, M. A. Wooldridge, and J. Ye, "Biotext search engine: beyond abstract search," *Bioinformatics*, vol. 23, no. 16, pp. 2196–2197, 2007. [Online]. Available: +http://dx.doi.org/10.1093/bioinformatics/btm301

[57] L. Lin and S. Evans, "Structural patterns in empirical research articles: A cross-disciplinary study," *English for Specific Purposes*, vol. 31, no. 3, pp. 150 – 160, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0889490611000615

[58] J. Fang, P. Mitra, Z. Tang, and C. L. Giles, "Table header detection and classification," *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, vol. 26, 2012.

[59] A. Gilani and r. F. Rukh Qasim, Shah and Malik, "Table detection using deep learning," 09 2017.

[60] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[61] S. C. C. D. M. Kavasidis, Isaak and Palazzo, "A saliency-based convolutional neural network for table and chart detection in digitized documents," *arXiv preprint arXiv:1804.06236*, 2018.

[62] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from web tables," in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242583

[63] W. L. Lu Wang. (2016, Okt) Pdf2htmlex. [Online]. Available: http://coolwanglu.github.io/pdf2htmlEX/doc/tb108wang.html#pfc/

[64] T. Harvey. (2016) Fontforge. [Online]. Available: http://fontforge.github.io/

[65] "Table extraction and pdf to xml with pdfgenie | pdftron blog," https://blog.pdftron.com/2014/03/02/table-extraction-and-pdf-to-xml-with-pdfgenie/, (Accessed on 08/20/2017).

[66] "Iso 19444-1:2016(en), document management — xml forms data format — part 1: Use of iso 32000-2 (xfdf 3.0)," https://www.iso.org/obp/ui/#iso:std:64911:en, (Accessed on 08/10/2017).

[67] "Whitepaper – pdf/a – der standard für die langzeitarchivierung," https://www.pdf-tools.com/public/downloads/know-how/whitepaper-pdfa-standard-iso-19005-de.pdf, (Accessed on 10/21/2017).

[68] S. Tuarob, S. Bhatia, P. Mitra, and C. L. Giles, "Automatic detection of pseudocodes in scholarly documents using machine learning," in *2013 12th International Conference on Document Analysis and Recognition*, Aug 2013, pp. 738–742.

[69] B. Yildiz, K. Kaiser, and S. Miksch, "pdf2table: A Method to Extract Table Information from PDF files," in *2nd Indian International Conference on Artificial Intelligence*, 2005, Refereed Conference & Workshop Articles. [Online]. Available: http://ieg.ifs.tuwien.ac.at/pub/yildiz_iicai_2005.pdf

[70] J. B. M. Manuel Aristarán, Mike Tigas, "Tabula: Extract tables from pdfs," http://tabula.technology/, online: (Accessed on 04/06/2017).

[71] M. Atzmueller, D. Benz, A. Hotho, and G. Stumme, Eds., *LWA 2010 - Lernen, Wissen & Adaptivität, Workshop Proceedings, Kassel, 4.-6. Oktober 2010*, 2010. [Online]. Available: http://www.kde.cs.uni-kassel.de/conf/lwa10/proceedings/proceedings.pdf

[72] T. Q. Luong. (2015) TrapRange: a Method to Extract Table Content in PDF Files. [Online]. Available: https://dzone.com/articles/traprange-method-extract-table

[73] Y. Liu, K. Bai, P. Mitra, and C. L. Giles, "Tableseer: Automatic table metadata extraction and searching in digital libraries," in *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*, ser. JCDL '07. New York, NY, USA: ACM, 2007, pp. 91–100. [Online]. Available: http://doi.acm.org/10.1145/1255175.1255193

[74] M. O. Perez-Arriaga, T. Estrada, and S. Abad-Mota, "Tao: System for table detection and extraction from pdf documents," in *FLAIRS Conference*, Z. Markov and I. Russell, Eds. AAAI Press, 2016, pp. 591–596. [Online]. Available: http://www.aaai.org/Library/FLAIRS/flairs16contents.php

[75] P. Pyreddy and W. B. Croft, "Tintin: A system for retrieval in text tables," in *Proceedings of the Second ACM International Conference on Digital Libraries*, ser. DL '97. New York, NY, USA: ACM, 1997, pp. 193–200. [Online]. Available: http://doi.acm.org/10.1145/263690.263816

[76] T. Kieninger and A. Dengel, *The T-Recs Table Recognition and Analysis System*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 255–270. [Online]. Available: http://dx.doi.org/10.1007/3-540-48172-9_21

[77] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovič, and R. Studer, "Transforming arbitrary tables into logical form with tartar," *Data Knowl. Eng.*, vol. 60, no. 3, pp. 567–595, Mar. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.datak.2006.04.002

[78] M. Kifer, G. Lausen, and J. Wu, "Logical foundations of object-oriented and frame-based languages," *J. ACM*, vol. 42, no. 4, pp. 741–843, Jul. 1995. [Online]. Available: http://doi.acm.org/10.1145/210332.210335

[79] "Document processing automation: Workflow | abbyy flexicapture," https://www.abbyy.com/en-eu/flexicapture/how-it-works/, (Accessed on 10/21/2017).

[80] Octav Ivanescu and Ivan Babiy, "Abbyy recognition technologies - ideal alternative to manual data entry. automating processing of exam tests," in *Proceedings of the 5th International Conference on Virtual Learning (ICVL)*. Bucharest University Press, ISSN 1844-8933, 2010, pp. 263–269.

[81] B. Klein, A. R. Dengel, and A. Fordan, *smartFIX: An Adaptive System for Document Analysis and Understanding*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 166–186. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24642-8_11

[82] M. Göbel, T. Hassan, E. Oro, and G. Orsi, "A methodology for evaluating algorithms for table understanding in pdf documents," in *Proceedings of the 2012 ACM Symposium on Document Engineering*, ser. DocEng '12. New York, NY, USA: ACM, 2012, pp. 45–48. [Online]. Available: http://doi.acm.org/10.1145/2361354.2361365

[83] Ana Costa e. Silva, "Metrics for evaluating performance in document analysis: application to tables," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 14, no. 1, pp. 101–109, Mar 2011. [Online]. Available: https://doi.org/10.1007/s10032-010-0144-2

[84] A. Peñas, B. Magnini, P. Forner, R. Sutcliffe, Á. Rodrigo, and D. Giampiccolo, "Question answering at the cross-language evaluation forum 2003–2010," *Language resources and evaluation*, vol. 46, no. 2, pp. 177–217, 2012.

[85] T. Grainger and T. Potter, *Solr in Action*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2014.

[86] W.-C. Lin and H.-H. Chen, "Merging mechanisms in multilingual information retrieval," in *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer, 2002, pp. 175–186.