

Automatic invoice Data identification with relations

By

Pettagam Tharindu Rukshan Ubewikkrama

2015/CS/136

This dissertation is submitted to the University of Colombo School of Computing

In partial fulfillment of the requirements for the

Degree of Bachelor of Science Honours in Computer Science

University of Colombo School of Computing

35, Reid Avenue, Colombo 07,

Sri Lanka

July 2020

Declaration

I, P.T.R Ubewikkrama of Student & Registration No.2015/cs/136 hereby certify that this dissertation entitled Automatic invoice Data identification with relations is entirely my own work and it has never been submitted nor is currently been submitted for any other degree.

.....

<Date>

.....

Signature of the Student

I, Dr. D.D. Karunaratne, certify that I supervised this dissertation entitled <Title> conducted by P.T.R Ubewikkrama in partial fulfillment of the requirements for the degree of Bachelor of Science Honours in Computer Science.

.....

<Date>

.....

Signature of the Supervisor

I, Mr. G P Seneviratne, certify that I supervised this dissertation entitled <Title> conducted by P.T.R Ubewikkrama in partial fulfillment of the requirements for the degree of Bachelor of Science Honours in Computer Science.

.....

<Date>

.....

Signature of the Co-Supervisor

Abstract

Mobile phones became a common device to everyone. There are 6 billion smartphone users in 2019. Because of that most of the time people tend to buy items through the mobile phone. Billions of online transactions happen each and every day. All generate e-invoices. People tend to use e-invoices rather than paper-based invoices. However the processing of those e-invoices is very hard.

Most of the time need to enter the e-invoice data into computers manually. Because no relationship between text elements makes automatic analysing PDF is very hard. To handle billions of invoices industry spend billions of money. By this research we introduce novel algorithm which was language independent and can be used to identify "Key-word , Value" pair data automatically

Preface

The results in this study rely upon the sample invoices created by me by using the real invoice data set. The analysis of the data is entirely my own work which I carried out with the help of apache pdfBox library.

Acknowledgement

I would like to express my sincere gratitude to my Supervisor Dr. D.D. Karunaratne and Co-supervisor Mr. G.P. Seneviratne for the continuous support on this project, for their patience, motivation and guidance throughout this project.

My parents for supporting me emotionally and financially as well as for being my strength.

The participants in the evaluation from University of Colombo School of Computing (UCSC) for their cooperation and enthusiasm.

Table of Contents

Declaration	2
Abstract	3
Preface	4
Acknowledgement	5
Table of Contents.	6
List of Figures	7
List of Tables	8
List of Acronyms.	9
Chapter 1 - Introduction	10
1.1 Background to the Research	1
1.2 Research Problem and Research Questions	4
1.3 Justification for the research	5
1.4 Methodology	6
1.5 Outline of the Dissertation	6
1.6 Definitions	7
1.7 Delimitations of Scope.....	7
1.8 Conclusion	8
Chapter 2 - Literature Review.	8
Chapter 3 - Design.	13
Chapter 4 - Implementation.	16
Chapter 5 - Results and Evaluation	22
Chapter 6 - Conclusions.	39

6.1 Introduction	39
6.2 Conclusions about research questions (aims/objectives)	39
6.3 Conclusions about research problem	39
6.4 Limitations	39
6.5 Implications for further research	39
References	41
Appendix A: Publications	42
Appendix B: Diagrams	43
Appendix C: Code Listings	44

List of Figures

1.1.1 X-Y tree generated from HTMM model.	3
1.1.2 combination of two segments.	4
2.1: The sparse lines in a PDF page	10
2.2: three features of an invoice image.	10
3.1: Overall design Architecture	13
4.1: Overall Architecture of libraries	16
4.2.2.1: Reading the pdf	18
4.2.2.2: Add relevant details to the list.	18
4.2.2.3: Unicode character stream with the coordinates.	18
4.2.2.3: How calculate the X,Y coordinate	18
4.2.2.4: Combined words	19
4.2.2.5: Sample input	19

List of Tables

4.2.2.1: After first iteration	20
4.2.2.2: After second iteration.	21
5.2.1.1 pre-Table special symbols and numbers	23
5.2.1.2 After table.	23
5.2.2.1 Pre-table No special symbols and have numbers	24
5.2.2.2 After table	24
5.2.3.1 pre-table No special symbols and numbers	25
5.2.3.2 After table	26
5.2.4.1 Contain special symbols and but no numbers	26
5.2.4.2 After table	27
5.2.5.1 pre-table No special symbols and with numbers	27
5.2.5.2 After table	28
5.2.6.1 pre-table No special symbols and one number	29
5.2.6.2 After table	29
5.2.7.1 pre-table Values missing horizontal	30
5.2.7.2 After table	30
5.2.8.1 pre-table Values missing vertical	31
5.2.8.2 After table	32
5.2.9.1 pre-table Complex scenario vertical	33
5.2.9.2 After table	34

Chapter 1 : Introduction

An invoice, bill is a commercial document issued by a seller to a buyer, relating to a sale transaction and indicating the products, quantities, and agreed prices for products or services the seller had provided the buyer. Payment terms are usually stated on the invoice. Although it has a 7000 year history, still most of the time they are processed manually. Most of the things done with the technology with higher accuracy. However some times that accuracy was not enough. To the invoice processing is the same. There are a lot of systems that are doing automatic invoice processing with not enough accuracy. There are some semi-automatic systems with higher accuracy, however user enrollment is very high in such systems. Therefore, it is our attempt to exploit this branch of Automatic processing of invoices and thereby contribute to the business world.

Thousands of thousands of research done to solve the automatic processing of documents. Also there are a large number of them that are tried on business documents. However most of them are focused on scanned or hand written documents. That is because it was very hard to process pdf documents which had no relationship between the components and only focused on final presentation. Therefore, through our research we aim to address this gap in the automatic processing of pdf documents.

1.1 Background to the Research

In the past there was a problem communicating visual material between different computer applications and systems. The specific problem is that most programs print to a wide range of printers, but there is no universal way to communicate and view this printed information electronically. PDF solve this problem which was written in PostScript language introduced by J. Warnock [1]. PostScript is a device independent page description language and this support for PostScript as a

standard makes the PostScript solution a candidate for this electronic document interchange.

Invoice can be grouped into two classes, paper-based invoices and E-invoices. In the past, people used only paper invoices. However, because of the revolution of technology people are more and more used to E-invoices. Billentis' report consists of the details about the invoices volumes used around the world each year. According to that nearly 6 billion mobile devices used around the world in 2019. Total volume of invoices estimate for 2019 is 550 billion and 2019 volume for e-bills/e-invoices will achieve at least 55 billion world-wide with annual growth rates of 10-20% in mid-term. Handling such amounts of invoices is a hard task and handling pdf invoices is a very hard task. Because when innovating the pdf, inventors only think about the presentation of the document. No relationship between words. Less number of tools available to manipulate pdf.

PDFMiner is a tool for extracting information from PDF documents. Unlike other PDF-related tools, it focuses entirely on getting and analyzing text data. PDFMiner allows one to obtain the exact location of texts in a page, as well as other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats. Apache Tika and Apache PDFBox are other well known pdf libraries which can be used to manipulate pdf [2][]. Those are still getting updated and there are many other good libraries which are stopped long years back because of the complexity of pdf documents. docsplit, Tabula, pdftohtml, pdftoxml etc.

There are two major solutions for extracting information from the pdf documents. First Hidden Tree Markov Models for Document Image Classification[3]. When a new document came, the system scanned it beginning from the upper left corner. When it found some section (section is one of the three sections of the pdf document, Text, image and vector graphics. Vector graphics are the things made with lines and curve like basic structures. Ex tables) it creates a new node. One text segment is one node likewise. The system scanned the whole document horizontally and vertically and created an X-Y tree from that [figure 01]. It assumes if the some keyword is some node then value should be child node. This system used Hidden Markov Model algorithm to find the primitives. (primitives are primitive keywords found in the document ex : total, description)

Second model is Field Extraction from Administrative Documents by Incremental Structural Templates[4]. In this model there are three components. User component, Database component and new invoice component. When a new invoice comes, the system first checks whether that invoice template is already available on the database or not. If it is available then it directly mapped the template with the invoice and extract data. If not, it segment the invoice and ask the user to labeled it. After the user labeled it it created a relative graph connecting each node to each one. Then it saves it as a new template. Figure 02 shows how the nodes connected. System keeps the angle between nodes and distance between nodes. There are a lot of commercial solutions available constructed based on the above two models. Some of the major products are IRIS , CLEARDATA , SmartSoft and Doc parser etc. All of those are used scanned invoices to extract the invoices. IRIS can scanned pdf documents and extract data, anatote data and many more. However all of these solutions are not fully automatic, they are semi automatic. Users need to annotate the documents.

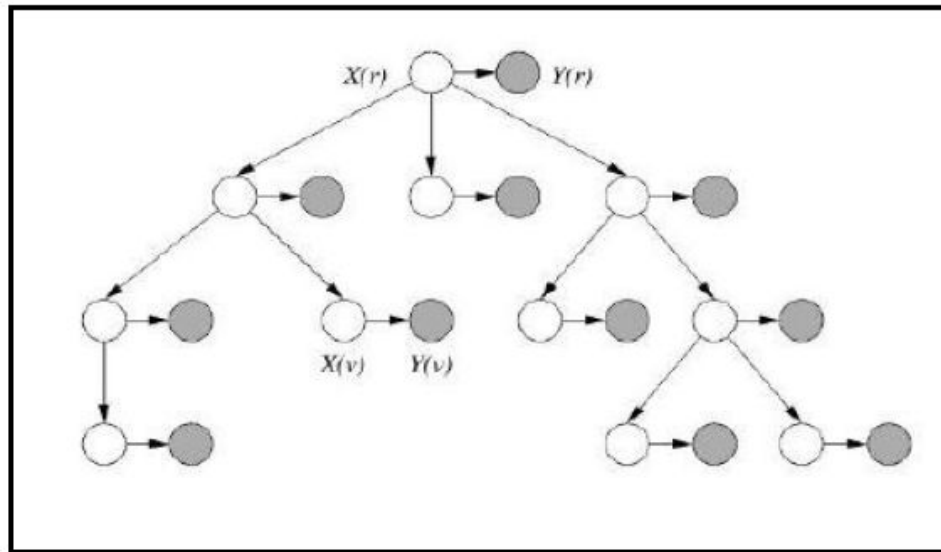


Figure 1.1.1: X-Y tree generated from HTMM model

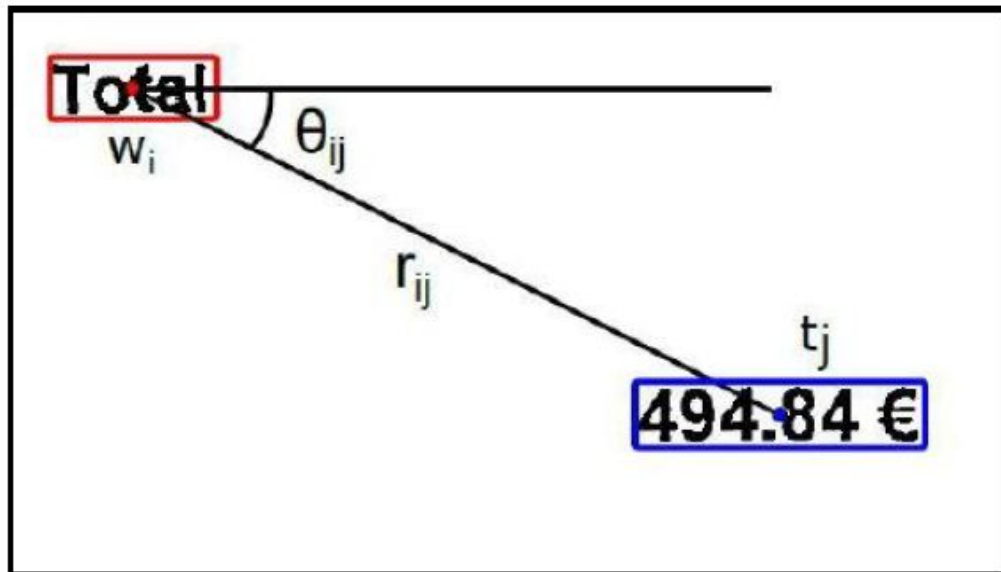


Figure 1.1.2: combination of two segments

1.2 Research Problem and Research Questions

1.2.1 How effectively can we write grammar to invoices and create a layout parse tree to classification?

As mentioned above each and every year there are billions of invoices moving around the world. Companies need to analyze their invoices to achieve their business goals. If there are online shopping portals they like to what items usually buy together (When customers buy one item, they can buy other items with buy together), what are the most selling items etc. In the past all invoices were paper based. Before analyzing them they need to enter each and every important details in the invoice manually to the computer. To do that they need a lot of manpower and need to spend a lot of money not only for the workers but also for buying papers to print the invoices. According to the Billentis report europeans average cost to process one manual invoice is \$15 and US government spends almost \$300 million just for paper invoicing in the goods. Using E-invoices we can reduce the paper cost, However the man power is the same. It is very important to automate this process.

Before the analysis we must classify invoices into groups. If we apply data extraction algorithms to each invoice it becomes a very time consuming task. So must

first classify them. To do that we used grammar. Grammars have been useful because they are intuitively simple to understand, and have very elegant representations. Their ability to model semantic interpretations of patterns, both spatial and temporal, have made them extremely popular in the research community. Also they are very fast when executing on the computer. After segmentation of the invoices we try to write a grammar to invoices, through that create parse tree for each and every invoices. Then when new invoice came compare parse trees and do the extraction steps. In here there are no predefined classes. When a new invoice comes which has a new layout then the system defines a new class for it. Using grammar introduces a new way to deal with invoices to computers.

1.2.2 Can we use Spaces between segments to create logical structure?

After the classification, we need to identify the logical relationships between segments. To do that we use spaces between the segments (As previously mentioned, segment can be a one keyword Ex: Total). In pdf documents there are no shifts between segments, all are well structured. If one keyword is theirs, the corresponding value of the keyword should be located near horizontal or vertical space around the keyword. Likewise use spaces to build relationships.

1.3 Justification for the research

As mentioned there are problems with automatic data extraction with their relationships. Through this research I am going to introduce new novel algorithm "Space algorithm" which solve those problem

1.4 Methodology

To manipulate the pdf files , we used mostly two very popular libraries. Apache PDFBox and PDFminer. Apache PDFBox is an open-source Java library that supports the development and conversion of PDF documents. Using this library, we can develop Java programs that create, convert and manipulate PDF documents. Using PDFBox, we extract Unicode text characters with the properis of them to construct the word locator. PDFMiner is another tool for extracting information from PDF documents. Unlike PDFbox, it focuses entirely on getting and analyzing text data. PDFMiner allows one to obtain the exact location of texts in a page, as well as other information such as fonts or lines same as the PDFbox. However both of two tools cannot be used to identify tables directly.

Invoice data set was given by the Creative Software company. In the dataset there are a lot of invoices with different languages and also different formats (image invoices and pdf invoices). First iteration only considers the pdf invoices in the english language.

The philosophical foundation of this research is based on the view that existing procedures can be verified through experiments, observations and mathematical logic. Therefore this research is an experimental research, which is designed to collect and interpret data, based on experiments and observations. Finally the validation will be conducted through experimentation and the findings of the research will be observable and quantifiable.

1.5 Outline of the Dissertation

Next chapter contains the Literature Review and the following chapters contain Design, Methodology ,implementation result and evaluation. Final chapter contains and conclusion.

1.7 Delimitations of Scope

1.7.1 In-Scope

- The project will cover the process writing grammar to invoices to classification
- Create logical structure of segments using the space algorithm
- Identify the invoice data automatically
- Deal if there are unexpected message in default invoice

1.7.2 Out-Scope

- If the invoice came rather than English language they will not be covered in the current phase of the project
- The initial phase will only cover pdf format invoices with single raw tables.
- Limited number of font Invoices may use a number of different font styles. However within the first face of the research focused on a limited number of fonts.

1.8 Conclusion

Above chapter explains how the handling of PDF invoices becomes a problem to society and what are the solutions they create to solve that. To handle those problems in this research introduce new algorithms which make layout relationships in a better manner.

Chapter 2: Literature Review

In the early 1990, the origin of the adobe acrobat and the PDF happened. At that time PostScript language was very popular all around the world. PDF builds on the PostScript page description language by underline the document structure. Also it has interactive navigation feature on PostScripts underlying imaging model, proving a convenient efficient mechanism enabling document to be reliably viewed and printed anywhere [5]. PDF documents preserve the look and feel of the original documents by describing the low-level structural objects such as a group of characters, lines, curves and images and associated style attributes such as stroke, color, font, fill, and shapes . Almost all PDF documents are untagged and don't have the basic high level logical structure information such as words, text lines, paragraphs, logos, and figure illustrations, which makes reusing, editing or modifying the layout or the content of the is very document difficult. When we consider a pdf document basically it consists of three components, Text, Images and Vector graphics. There is a lot of research going on making the pdf more than the viable format. Rather than logical structure, creating the layout of the pdf is also very difficult [6]. The logical structure is a tree of logical objects such as 'title' and 'author', while the layout structure is a tree of layout objects which can be represented as rectangles. Hui Chao and Jian Fan tried to extract the layout from the pdf documents by dividing one pdf into three separate pdf documents , Text only pdf, image only PDF and vector graphic pdf. Identify each object easily from related pdf and finally combine three pdfs and make a single xml file.

To extract text from the pdf there are a huge number of libraries. But to extract vector graphics there are a very limited number of tools. Identify the tables and text within them is very hard in pdfs. Extracting table metadata is a challenging problem for several reasons: diverse medium types, no formal table designing

rules/standards, different presentation schemes in different mediums, different table layout requirements of different publishers, diverse table cell types etc. In order to characterize tables spanning over a wide range of documents, a rich and flexible set of representation metadata is required. Ying Liu, Prasenjit Mitra propose a set of medium independent table metadata to facilitate the table indexing, searching, and exchanging. To extract the contents of tables and their metadata, they used automatic table metadata extraction algorithm[6] They done it by converting a PDF document into a formatted text file, detecting the table candidates based on location analysis and keyword matching (table environment/geography metadata is extracted in this phase), confirming/denying table candidates, and recognizing the table structure. Identifying table boundaries correctly helps to improve table data detection . Ying Liu introduced a method to improve the table boundary detection performance by considering the sparse-line property of table rows [7,8,9]. That method easily simplifies the table boundary detection problem into the sparse line analysis problem with much less noise. There are many sparse lines that can be found within the same document same as the invoice.

Figure metadata extraction in invoices is also very important sometimes. There can exist two companies who are using the same invoice template. When we classify them into groups , in parse trees there is no difference. So we need to analyse company logo details which are inside the image object. Extraction of figures and text: We use a popular Java based PDF processing library PDFBox to extract text (text lines are extracted sequentially, as they appear in the original file) and raster graphics (image file for the graphic element, location, length and width) from PDF files. PDFBox or other common PDF processing libraries (Xpdf, PyPDF8) are not suitable for extraction of vector graphics. Also, they do not extract text from scanned articles, which need to be processed by OCR. Further we can divide image text identification in to three main parts. Color feature extraction, Text feature extraction and shape feature extraction [10]

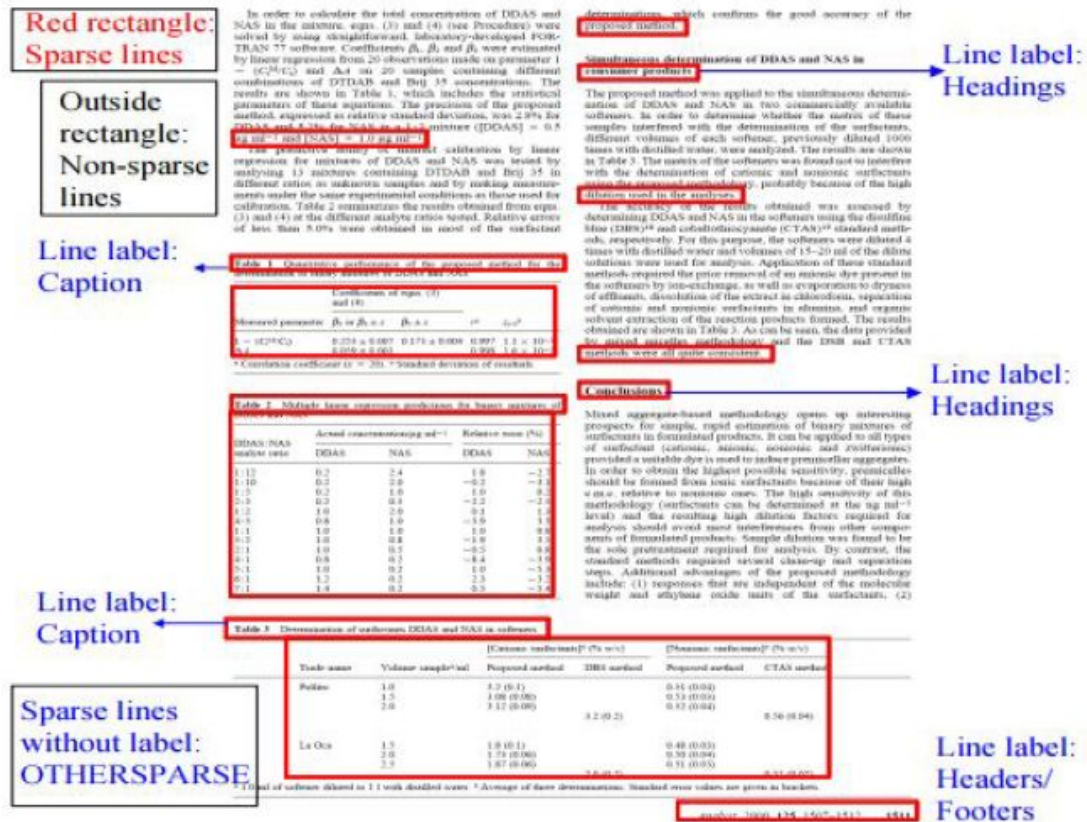


Figure 2.1: The sparse lines in a PDF page

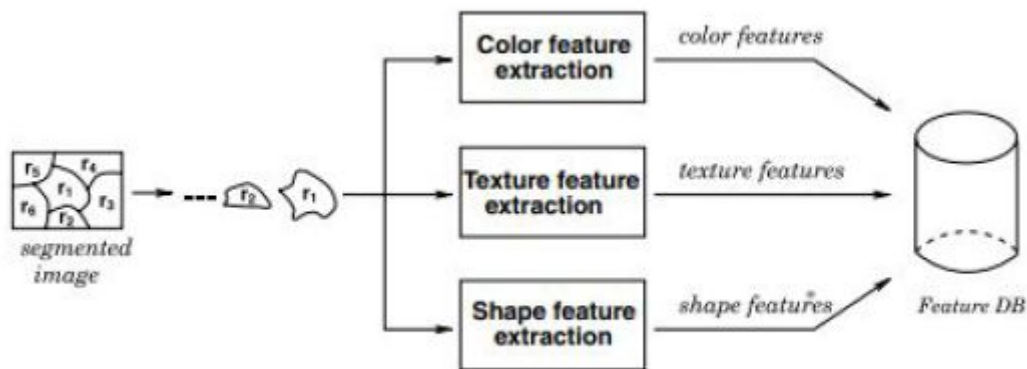


Figure 2.2: three features of an invoice image

From the information extraction point of view one problem is that the order of

textual objects in the file do not always correspond to the reading order. Several converters are available either open-source or commercial (a good survey can be found in [11]). More recently, in [12] text objects are extracted from the PDF and a word and line segmentation is produced based on heuristics using the distance between characters, their geometrical positions and ontologies can be used. Greenstone open source library software which can be used to information extraction. However because of geometrical usage they have some problems explained in the section below [13].

Identify the values in the invoice we use grammar. Grammars are very popular in the research community because of simplicity and very fast execution. Grammars have been useful for providing structural descriptions of a variety of objects like telephone numbers, addresses, English sentences etc., and associating semantic information with the resulting models. They have been most widely used by the natural language processing community. Grammars consist of a collection of basic primitives and a set of rules which compose patterns out of these primitives. Thus collections of objects which exhibit structural similarity can be designed using grammar. For example, the set { 00110, 0110, 01110...} describes the set of sequences over 01 which have a sequence of 1s bounded by two 0s. Much more complicated sets, which exhibit regularity in higher dimensions including time, can be modeled using grammar. Roughly speaking, they generate sub-patterns out of basic primitives, and then generate bigger or more complicated patterns from these subpatterns. English words can be constructed by combination of characters. Address can be constructed by a set of numbers followed by symbol followed by number or character sequence. We can define those grammar rules. There are several issues in the modeling of objects or events using grammar. The first issue is the choice of basic primitives and the appropriate grammar type. The choice of primitives is determined by how easily basic features can be extracted from the data, while the choice of grammar type is determined by the complexity of the desired structure. Ex : When identifying telephone numbers, primitives will be Symbol + / 0-9 numbers.

Logical connections can be identified by using spaces. Scanning all spaces between pdf objects we can find the two threshold values. threshold calculated using the four spaces around the world. Left, right, top and bottom. Least distance is the

distance between keywords. It is used as the one threshold. Other one is the second least common distance . that is the distance between the keyword and value. If there are symbols, the first threshold is the between symbols and the second one is between keywords.[14].

Use machine training techniques for identification vast popular method in now

days. However pre defined knowledge is the main disadvantage. Marc, al Rusi n ~ o propose an information extraction method that also relies on registering portions of layout elements. The proposed model is extremely simple and requires a minimum of human intervention. In addition, our main contribution is that although the template model is learned using a single annotated image, the subsequent processed images incrementally adjust this model by a reformulation of the tf-idf statistic. The main advantage of such a method is that the learned model can absorb variations of the invoice layout through time and learns in an unsupervised way which are the discriminative layout elements that help the most to extract a certain field[15]. Some use knowledge based methods to extract information.[16] here If the analysis system fails to analyze an example document, the knowledge descriptions are incrementally modified to cover the correct layout objects included in that document.

Chapter 3 - Design

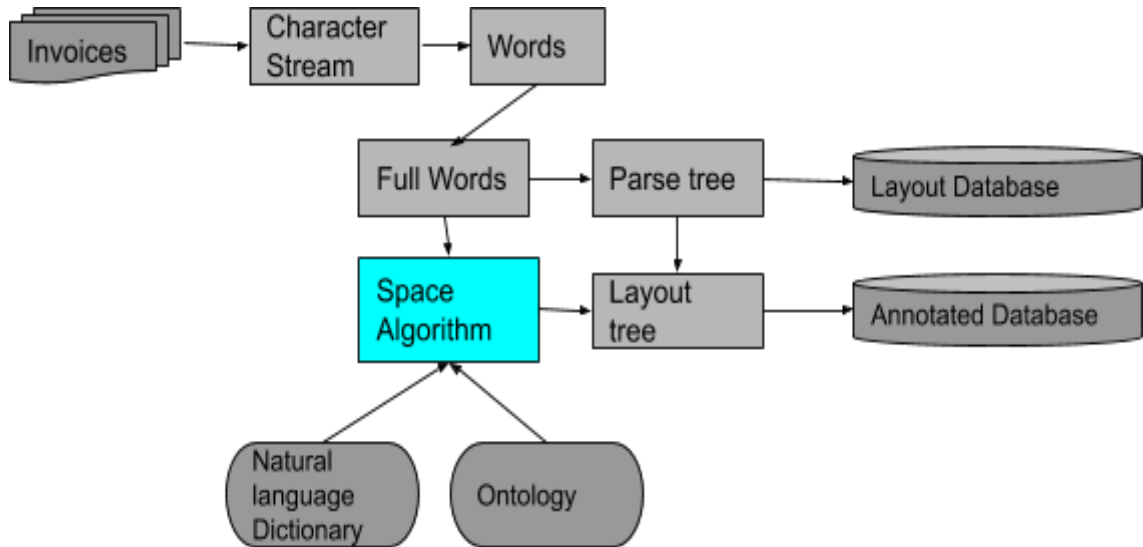


Figure 3.1: Overall design Architecture

This section explains the overall design of the system. This path used to create the parse tree for the invoice. PDF document content stream lists all the page objects such as text objects, image objects, path objects etc. First we define these objects as primitives and other direction primitives. Path objects are referred to as vector graphics objects or drawings and paintings composed of lines, curves and rectangles. Page objects in PDF documents don't reflect nor are related to the logical structure or logical components of the document. For example, a text object may only have part of the characters of a word; path objects which are the building blocks for the graphical illustrations, such as bar charts, pie charts and logos are often just a fraction of the whole figure illustrations e.g. one bar in a bar chart. To discover the logical structure of a document, first we need to analyze and segment the document and get the characteristics of each object (text object have text size, font, text width, x-axis and y-axis coordinates from the left upper corner which conder as "0,0" etc. image object may contain the size ,location etc, path (vector graphic) object may contain the width,size and location)

Using the different pdf manipulation libraries identify the three typed segments and characteristics .In this research if there are some text within an image, they are not considered. Just take that component as an image component. However tables are considered. But when considering a table, we did not assume it as a table. Just take as the combination of lines. Directive primitives used to create hold the relative positions of the objects and grammar word :

Ex : some invoice get the output as :

`"Text"->"west"->"line"->"text"->"right"->"text"->"down"->"line"->"text"->"left"->"text"`

Using grammar parser test the above word of the language with the already available words in the layout database library. If there is a matching word that means we already identified that layout. So then we can ask from the annotated invoice library to get the meaning of that layout.

In here basically do the creation of logical structure. First consider inputs. Three inputs needed to the algorithm.

- Spaces between object (can be calculated within the algorithm when the locations are given)
- Special characters (Double colon, dot , dash line etc)
- Numbers

Spaces are the primary input. When segmenting the image we get the coordinates of each object. Use them as input and calculate the spaces between the objects. In general space between the two key word sections are less than the space between the keyword and the value. In an invoice if there is a keyword which has a value (ex Total , address) then the related value must be in a horizontal way or in vertical way from the key word. Using that concept we can create logical structure to some extent without even considering the internal data of the object.figure

When analysing the data we found some invoices which are the spaces between the keywords and the spaces between the values are the same. To handle this kind of scenario we use special characters and font sizes as inputs. If the algorithm finds the spaces are the same then it uses other inputs to process further. If there's a special character and it is close to some side from the left and right sides ,

that means algorithms find the keyword section. The other section near to special character must be the value. After the same spaces if there are no special characters then algorithm checks for font sizes. In general, most of the time keyword sections get the higher font size. Addition to that algorithm consider the content of the text objects. Whether it is a numeric segment or alphabet segment. if near two segments are from two different types, that means alphabet one must be keyword.

Second, after the creation of the logical structure we use natural language dictionary and ontologies to validate the result. When we write the same keywords in different invoices we can write it in completely different ways but it must be the same. To solve that problem we use a natural language dictionary. Through that we map the key words to the general keyword. Also if only we construct logical trees it was not useful. There is a way to map the output to the real world general invoice layout. For that also this mapping is very important.

Then we use ontologies to identify the values of keywords. For example, there is a way to construct an address, write a date , discount etc. though that validate the keyword value matching are correct.

After getting the relationships combine the logical structure and the parse tree of the same algorithm and store in the Annotated invoices with semantics database.

Chapter 4 -Implementation

This project is accomplished through open source software. It is written in Java language using IntelliJ IDEA.

4.1 Discussion on technology used

This section discusses the technology that is used in this project and code implementations can be found in Appendix A. The overall architecture of the libraries and the connections among them is given in Figure 4.1.



Figure 4.1: Overall Architecture of libraries

4.1.1 Personal Computer(PC)

The PC used in this project has the following specifications, as it has to have the processing power to accommodate the requirements.

- Processor: Intel Core i5-8250U
- Ram: 16GB
- Operating System: Windows 10

4.1.2 intelliJ idea

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development. In this research I used the community version of the IDE.

4.1.3 Apache PDFBox® - A Java PDF Library

The Apache PDFBox® library is an open source Java tool for working with PDF documents. This library allows creation of new PDF documents, manipulation of existing documents and the ability to extract content from documents. Apache PDFBox also includes several command-line utilities. Apache PDFBox is published under the Apache License v2.0. This has built in functionality for text extraction with their characteristics. This research used PDFBox version 2.0.15 released in 2019/04/11 and used the above functionality to take the character stream with their related characteristics.

4.1.4 tabula-java

4.2 Implementation of the functionalities

This chapter discusses how the required functionalities were implemented using predefined libraries in java. Steps which were implemented separately are discussed here. In addition, the benefits and drawbacks of each stage will further be discussed.

4.2.1 Data Gathering

Invoice contains personal information about the buyer and seller. If the invoice is publically available then there is a privacy violation. So there are no dataset over the internet. Looking at the some sample invoices I have to create dummy invoices with dummy data which protect the original invoice structure. Most of the original invoices were given by the creative software company which contains the different invoices with different styles.

4.2.2 Read PDF and Character extraction

As the first step, first read the PDF file using the Java PDFbox *PDFTextStripper* class. To get the text position coordinates, font style characteristics, Unicode values of the characters and text size can be extracted using the *writeString* class. After

reading override the *writeString* class to get the relevant characteristics of each and every character including the space characters.

```
void ReadData() throws IOException {
    File file = new File( pathname: "C:\\Users\\Rukshan\\Documents\\Research\\RTest01\\Test.pdf");
    PDDocument document = PDDocument.load(file);
    PDFTextStripper stripper = new ReadPdf();
```

Figure 4.2.2.1: Reading the pdf

```
for (TextPosition text1 : textPositions) {
    positionList.add(new Position(text1.getUnicode(),text1.getXDirAdj(),text1.getYDirAdj()));
```

Figure 4.2.2.2: Add relevant details into list

In this research only used location details of each character. X direction, Y direction of each text and the unicode character of the text are added to the list named “positionList”. Directions are calculated considering the left top corner as 0,0 position and take the absolute value of the output value.

```
P 516.359 50.666016
a 523.02905 50.666016
g 528.58905 50.666016
e 534.69904 50.666016
: 540.25903 50.666016
 543.589 50.666016
l 546.369 50.666016
```

Figure 4.2.2.3: Unicode character stream with the coordinates

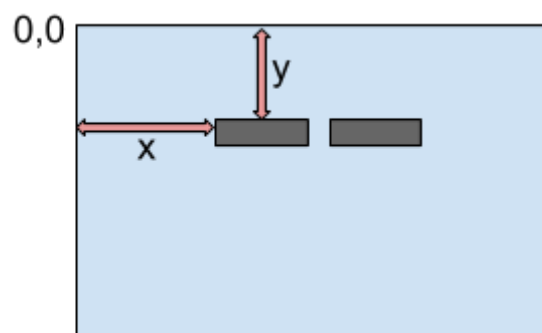


Figure 4.2.2.3: How calculate the X,Y coordinates

As the second step then combine those characters into groups to create meaningful words. Simple regular grammar expressions used for that. each and every character given as input and if the next input character is an escape character then it considers the previous input stream as a one word. If the grammar found the word, then it mark the first letter of the word as the starting point of a word.

But creating the words are not useful if they didn't give a meaningful definition. Spaces between the two characters within the word and between the word and the word and the text segment and text segment are totally different. Space between word and the word higher than the character and the character. Space between segment and segment is higher than the space between the two words. Algorithm used that characteristic to divide the segments. After creating the words used simple algorithm which take the words by words and compare them with the leftmost character position of each word and calculate the spaces (left most character coordinates represent the coordinates of the word Ex : In Figure 4.1 word 'page' coordinates equal to the character 'p' coordinates which was $X = 50.666$ and $Y = 516.359$). **Figure 4.1** shows sample segmented text output.

```
Taxable amount
1 437,95 SEK
Total incl. VAT
1 797,44 SEK
VAT summary
VAT code
VAT exemption
VAT rate
Taxable amount
```

Figure 4.2.2.4: Combined words

As shown in Figure 4.1: those words make a meaning segment. When we consider the value '1 437,95' and 'SEK' as two words they didn't make any sense. However combining those gave a meaningful segment. But important to mention this algorithm works on the horizontal words. But writing an address most often is written in a vertical fashion. To solve those scenarios used space algorithm which explains the later sub section.

4.2.2 Space Algorithm

This was a novel algorithm which was very accurately working on text segments and creating logical structure between them. As input to the algorithm need to give the text segment with the coordinates. This algorithm works independently of the language so it can be used for any language invoices.

Name :	Thrindu Rukshan
Email :	Ptr.Ubwikkrama@gmail.com
Age	18
Invoice No	18ADS2122

Figure 4.2.2.5: Sample input

There are few predefined values for the algorithm.

- If space higher than below one then return **1** else **0**
- If the numeric characters found in the words then return **-2** else **+2**
- if Special symbols can be found then return **+2** else **0**

These values are assigned to give the higher value to the Key Word part of the KeyWord, Value combination. Looking at the values then algorithm mark the text segments as keywords and their related values. As example given the input figure 4.2.2.5 sample input then as shown in Figure 4.1 are the related values for each word calculated after the first iteration. When in the second iteration it changes the values according to some section process. First it checks the two values of related coordinates are less than its value. if not it continues to the next value. if not again continues. This process repeats until it finds less than values. If they are found then it selects its child from the two values and updates the related space values by adding one more to previous values. Doing so changes the previous unmatched cases to matched cases. Repeat this until it covers the whole text segments in the document. In the above scenario algorithm need to only run twice to map the details.

	Space	Number	Spec.Char	Total
Name:	1	+2	+2	5
Email:	1	+2	+2	5
Age	1	+2	0	3
Invoice No	0	+2	0	2
Tharindu Rukshan	1	+2	0	3
Ptr.Ubewikkrama@gmail.com	1	+2	0	3
18	1	-2	0	-1
18ADS2122	0	-2	0	-2

Table 4.2.2.1: After first iteration

	Space	Number	Spec.Char	Total
Name:	1	+2	+2	5
Email:	2	+2	+2	6
Age	2	+2	0	4
Invoice No	1	+2	0	3
Tharindu Rukshan	1	+2	0	3
Ptr.Ubewikkrama@gmail.com	1	+2	0	3
18	1	-2	0	-1
18ADS2122	0	-2	0	-2

Table 4.2.2.2: After second iteration

5. Results and Evaluation

This section is divided into three main subtopics. First introduce the detailed dataset (5.1) then the second research question results are evaluated (5.2) and finally the first research question results are evaluated(5.3).

5.1 Dataset

Dataset contains 56 invoices with different layout styles with dummy data (Same layout as original).Invoices are in english language. However there are 10 possible scenarios which can happen within the invoice. Applying those cases to each layout generates 840 invoices. All 560 invoices are used for result calculations.

5.2 Results with the algorithm

Following subsection describes how Space algorithms behave and generate results in each different possible cases.

5.2.1 Contain special symbols and numbers

Date :	2019-12-22
Invoice number :	5FS5233SD

In this case keywords contain the special symbols at the end “:” and values contain the numbers within it. Both “Date” and “Invoice number” get +2 for without number and +2 for special symbols and only “Date” gets +1 for space. because the algorithm cannot calculate space probability for “Invoice number” because it didn’t have a bottom member. Table 5.2.1.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
Date	+2	+2	+1	5
Invoice Number	+2	+2	0	4
2019-12-22	-2	0	0	-2
5FS5233SD	-2	0	0	-2

Table 5.2.1.1 pre-Table special symbols and numbers

In the second iteration algorithm changed the previous values when it found the clear positive number difference between two related text segments. Here “Date” has value +5 and its right member has value -2 and the bottom member has value +5. So the right member gives a higher difference than the bottom member ($-7 < 0$). Then the algorithm assigns the value “2019-12-22 ” to the “Date” and increment the “Invoice number” previous value by one and decrease the “5FS5233SD” by one (Table 5.2.1.2)because the neighbouring segments also have higher probability to found its related value in the right hand location.

	Previous	New
Date	5	5
Invoice Number	4	5
2019-12-22	-2	-2
5FS5233SD	-2	-3

Table 5.2.1.2 After table

5.2.2 No special symbols and have numbers

Name	Tharindu Rukshan
Invoice number	5FS5233SD

In this case keywords do not contain any special symbols at the end and one value contains the numbers within it. Both “Name” and “Invoice number” get +2 for

without number and 0 for special symbols and only “Name” gets +1 for space. Table 5.2.2.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
Name	+2	0	+1	3
Invoice Number	+2	0	0	2
Tharindu Rukshan	+2	0	0	2
5FS5233SD	-2	0	0	-2

Table 5.2.2.1 Pre-table No special symbols and have numbers

In the second iteration, “Name” has value +3 and both right and bottom members have value 2. So algorithm can't find the difference. Then it passes the implementation from “Name” to “Invoice number”. Here “Invoice number” has value +2 and the right member has value -2. There algorithm can find the difference. Then the algorithm assigns the value “5FS5233SD ” to the “Invoice Number” and increment the “Name” previous value by one and decrease the “Tharindu Rukshan” by one (Table 5.2.2.2)because the neighbouring segments which passed iteration to it also have higher probability to found its related value in the right hand location. Then there is a clear difference between “Name” and “Tharindu Rukshan” and it creates a relationship between those two.

	Previous	New
Name	3	4
Invoice Number	3	3
Tharindu Rukshan	2	1
5FS5233SD	-2	-2

Table 5.2.2.2 After table

5.2.3 No special symbols and numbers

Name	Tharindu Rukshan
City	Colombo

In this case both keywords and values do not contain any special symbols or numbers. All get +2 for without number and 0 for special symbols and only “Name” gets +1 for space. Table 5.2.3.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
Name	+2	0	+1	3
City	+2	0	0	2
Tharindu Rukshan	+2	0	0	2
Colombo	+2	0	0	2

Table 5.2.3.1 pre-table No special symbols and numbers

In the second iteration, Here “Name” has value +3 and both right and bottom members have value 2. So algorithms can't find the difference. Then it passes the implementation from “Name” to “City”. Here “City” has value +2 and the right member also has value +2. Their algorithm can find the difference. But no bottom member is available for the “City” . Because of that algorithm assigns the value “Colombo ” to the “City” and increment the “Name” previous value by one and decrease the “Tharindu Rukshan” by one (Table 5.2.3.2)because the neighbouring segments which passed iteration to it also have higher probability to found its related value in the right hand location. Then there is a clear difference between “Name” and “Tharindu Rukshan” and it creates a relationship between those two.

	Previous	New
Name	3	4
City	2	4
Tharindu Rukshan	2	1
Colombo	2	2

Table 5.2.3.2 After table

5.2.4 Contain special symbols and but no numbers

Name -	Tharindu Rukshan
City -	Colombo

In this case both keywords have a special symbol “-” and values do not contain any special symbols or numbers. All get +2 for without number and Keywords get “+2” for special symbols and only “Name” gets +1 for space. Table 5.2.4.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
Name	+2	+2	+1	5
City	+2	+2	0	4
Tharindu Rukshan	+2	0	0	2
Colombo	+2	0	0	2

Table 5.2.4.1 Contain special symbols and but no numbers

In the second iteration, Here “Name” has value +5 and its right member has value +2 and the bottom member has value +4. So the right member gives a higher difference than the bottom member ($1 < 3$). Then the algorithm assigns the value “Tharindu Rukshan ” to the “Name” and increment the “City” previous value by one

and decrease the “Colombo” by one (Table 5.2.4.2)because the neighbouring segments also have higher probability to found its related value in the right hand location.

	Previous	New
Name	5	5
City	4	5
Tharindu Rukshan	2	2
Colombo	2	1

Table 5.2.4.2 After table

5.2.5 No special symbols and with numbers

ID	Description	prize
01	04 Buns	Rs . 200 /=

In this case both keywords and values do not contain any special symbols and values contain numbers. In previous examples keyword value pairs are located in a horizontal direction and now they are located in Vertical direction. This is a situation like a single raw table. All keywords get +2 for without number and 0 for special symbols and “ID” and “Description” gets +1 for space. Table 5.2.5.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
ID	+2	0	+1	3
Description	+2	0	+1	3
Prize	+2	0	0	2
01	-2	0	0	-2
04 Buns	-2	0	0	-2
Rs . 200 /=	-2	0	0	-2

Table 5.2.5.1 pre-table No special symbols and with numbers

In the second iteration , Here “ID” has value +3 and its right member has value +3 and the bottom member has value -2. So the bottom member gives a higher difference than the bottom member ($-5 < 0$). Then the algorithm assigns the value “01” to the “ID” and increment the “Description” previous value by one and decrease the “04 Buns” by one (Table 5.2.5.2) ,because the neighbouring segments also have higher probability to found its related value in the right hand location. In “Description” it’s right member and bottom member both have less values, However the bottom value has the minimum value. So repeat the same procedure done by the “ID” segment.

	Previous	New
ID	+3	3
Description	+3	4
Prize	2	3
01	-2	-2
04 Buns	-2	-3
Rs . 200 /=	-2	-3

Table 5.2.5.2 After table

5.2.6 No special symbols and one number

Name	Description	prize
Buns	Cream Buns	Rs . 200 /=

In this case both keywords and values do not contain any special symbols and only one value contains a number. This is a special case where algorithm didn’t assign values to key words just looking at the space values. It iterates till it can find the best difference then recursively come back. In here also keyword value pairs are located in a Vertical direction. All keywords get +2 for without number and 0 for special symbols and “Name” and “Description” gets +1 for space. Table 5.2.6.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
Name	+2	0	+1	3
Description	+2	0	+1	3
Prize	+2	0	0	2
Buns	+2	0	0	2
Cream Buns	+2	0	0	2
Rs . 200 /=	-2	0	0	-2

Table 5.2.6.1 pre-table No special symbols and one number

In the second iteration , Here “Name” has value +3 and its right member has value +3 and the bottom member has value 2. So the bottom member gives a higher difference than the bottom member . However the only situation which can happen on 1 difference in second iteration is by space difference. Therefore the algorithm didn't create a relationship and continue to the next member. In “Description” both right and bottom members have the same value. So continue. Finally the “Prize” has no right member and the bottom member has a clear difference .Then it makes a relationship and updates the previous values. It increases the “Description” previous value by one and decreases the “Cream Buns” by one (Table 5.2.6.2) ,because the neighbouring segments also have higher probability to find its related value in the bottom hand location. Same happens to the “Name” member.

	Previous	New
Name	+3	+4
Description	+3	+4
Prize	2	2
Buns	2	1
Cream Buns	2	1
Rs . 200 /=	-2	-2

5.2.6.2 After table

5.2.7 Values missing horizontal

Name -	Tharindu Rukshan
City -	
Tp -	0711234567

In this case all keywords have a special symbol “-” and one values contains numbers. All get +2 for without number and Keywords get “+2” for special symbols and only “Name” gets +1 for space. Table 5.2.7.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
Name	+2	+2	+1	5
City	+2	+2	0	4
Tharindu Rukshan	+2	0	0	2
Tp	+2	+2	0	4
0711234567	-2	0	0	-2

Table 5.2.7.1 pre-table Values missing horizontal

In the second iteration, Here “Name” has value +5 and its right member has value +2 and the bottom member has value +4. So the right member gives a higher difference than the bottom member ($1 < 3$). Then the algorithm assigns the value “Tharindu Rukshan ” to the “Name” and increment the “City” previous value by one. “City” doesn’t have the right member , only the bottom member. Although it has a higher value than the bottom one, it did not take the bottom one as the value. Because he knows that his value is increased by the above one which means the value relent to it should have on the right side. but no value on the right side. then it takes an empty string as its value. But it increases the “Tp” by one. (Table 5.2.7.2)because the neighbouring segments also have a higher probability to find its related value in the right hand location.

	Previous	New
Name	5	5
City	4	5
Tharindu Rukshan	2	2
Tp	4	5
0711234567	-2	-2

Table 5.2.7.2 After table

5.2.8 Values missing vertical

ID	Description	prize
01		Rs . 200 /=

In this case both keywords and values do not contain any special symbols and values contain numbers. This is a situation like a single row table with one value missing. All keywords get +2 for without number and 0 for special symbols and "ID" gets +1 for space. Table 5.2.8.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
ID	+2	0	+1	3
Description	+2	0	0	2
Prize	+2	0	0	2
01	-2	0	0	-2
Rs . 200 /=	-2	0	0	-2

Table 5.2.8.1 pre-table Values missing vertical

In the second iteration , Here "ID" has value +3 and its right member has value +2 and the bottom member has value -2. So the bottom member gives a higher difference than the bottom member ($-5 < 1$). Then the algorithm assigns the value "01" to the "ID" and increment the "Description" previous value by one (Table 5.2.8.2

) ,“Description” doesn’t have the bottom member , only the right member. it did not take the right one as the value. Because he knows that his value is increased by the left one which means the value relent to it should have on the bottom side. but no value on the bottom side. then it takes an empty string as its value. But it increases the “Prize” by one.

	Previous	New
ID	3	3
Description	2	+3
Prize	2	+3
01	-2	-2
Rs . 200 /=	-2	-2

Table 5.2.8.2 After table

5.2.9 Complex scenario vertical

Name -	Tharindu Rukshan	Code -	SA546224
City -	Balapitiya	Date -	2019/10/22
Tp -	0711234567		

Space algorithm start with random locations . It may be the leftmost element or may not. Here consider if it starts with the leftmost one and the scenario if it starts with the next element.. In this case all keywords have a special symbol “-” and some values contain numbers. All get +2 for without number and Keywords get “+2” for special symbols and “Name”, “City”, “Tharindu Rukshan”, “Balapitiya” and “Code” gets +1 for space. Table 5.2.7.1 contains the pre assigned values.

	Number	Special Symbol	Space	Total
Name	+2	+2	+1	5
City	+2	+2	+1	5
Tharindu Rukshan	+2	0	+1	3
Tp	+2	+2	0	4

0711234567	-2	0	0	-2
Balapitiya	+2	0	+1	3
Code	+2	+2	+1	5
SA546224	-2	0	0	-2
Date	+2	+2	0	4
2019/10/22	-2	0	0	-2

Table 5.2.9.1 pre-table Complex scenario vertical

If it starts in the leftmost one it changes the values as below.

	Previous	New
Name	5	5
City	5	6
Tharindu Rukshan	3	3
Tp	4	4
0711234567	-2	-3
Balapitiya	3	2
Code	5	5
SA546224	-2	-2
Date	4	5
2019/10/22	-2	-3

Table 5.2.9.2 After table

If it started next to the left most one “Tharindu Rukshan” , It first checked it with two members “Code” and “Balapitiya”. However the right member has a higher value than the starting one. That means it starts with the wrong location. Then it changes to the left one of it “Name”.Then the algorithm behaves normally as above.

5.2.10 Complex scenario vertical and horizontal

Name - Tharindu Rukshan	
Tp	Code
0711234567	SA546224

In this case algorithm assign special symbols +2 for “Name” and “Tp” and “SA546224” get -2 for numbers.

	Number	Special Symbol	Space	Total
Name	+2	+2	+1	5
Tharindu Rukshan	+2	0	0	2
Tp	+2	0	1	3
0711234567	-2	0	0	-2
Code	+2	0	0	2
SA546224	-2	0	0	-2

Table 5.2.10.1 pre-table Complex scenario vertical and horizontal

In the second iteration, it checks the right and the left members of the “Name”. “Tharindu Rukshan” has a higher difference then it takes it as value and then it increases the “Tp” by one and “Code” by -1. However the “Code” is a keyword. When it comes to the “Tp” its two members “0711234567” has lower value. Then it assigns the value and decreases the value of “Code” by 1 again. So no change happened to “Code”.

	Previous	New
Name	5	5
Tharindu Rukshan	2	3
Tp	3	4
0711234567	-2	-2
Code	2	2
SA546224	-2	-1

Table 5.2.10.2 After table

5.3 Performance of the algorithm

Dataset contains 560 preposed invoices. Each invoice contains between 20 to 36 keyword-value relationship couples. All it contains 17600 couples.

Case	Total Couples	Correctly mapped	prasentage
Contain special symbols and numbers	2650	2650	100%
No special symbols and have numbers	2500	2500	100%
No special symbols and numbers	1250	1194	95.52%
Contain special symbols and but no numbers	2420	2420	100%
No special symbols and with numbers	1200	1200	100%
No special symbols and one number	1300	1300	100%
Values missing horizontal	1250	1250	100%
Values missing vertical	1250	1156	92.48%
Complex scenario vertical	2300	2300	100%
Complex scenario vertical and horizontal	1480	1480	100%

Two cases algorithm gave less accuracy than others. This happens if the user did not follow the layout of the invoice. In the first case there are no special characters or numbers. Also they didn't follow the order of the invoice. They are not correctly aligned. "Company Name", "Delivery address" and "No. persons" are correctly aligned but "Contact person" is not. They just create the invoice like a paragraph (Image 5.3.1). Which was an exceptional case found. And in the next case they follow the layout for a little bit. This also No symbol No number case and even they didn't have space differences. Algorithm map them assuming they are horizontal. But it can be vertical(Image 5.3.2).



Catering Invoice

Company Name ABC Company

Delivery Address Seenigoda wathugedara Suite/Rc

No. People Two Contact Person Tharindu

Image 5.3.1 Irrelevant layout

Name	Description	Quantity
Buns	Cream Buns	two

Image 5.3.2 no space change

In the second case "Values missing vertical" , some invoices contain the column values which expand to other columns (Image 5.3.3). This scenario algorithm fails to identify which column it was related to. Because of that it took 92.48% accuracy. Overall performance of the algorithm is approximately **98.8%** . This was a significant improvement than the previous methods but it cannot directly compare with those because there are not benchmark dataset available.

Name	Description	Quantity	Total
Buns	Cream Buns with cherries		Rs 100/=

Image 5.3.3 Irrelevant table filling

5.3 Parse tree results

Classifying invoices is very important for efficiency. However what happens if it consumes more time to classify the invoice rather than directly identify relationships. Keywords are the primary and only building block for a parse tree. But we cannot get the keywords in the first iteration. First need to identify the relationships using the "Space algorithm", then keywords can be identified. But primary goal is to create the logical structure and it can be achieved in a fast manner without a parse tree. Graphs show how the time behaves when the relationship size increases to create a parse tree (Image 5.4.1).

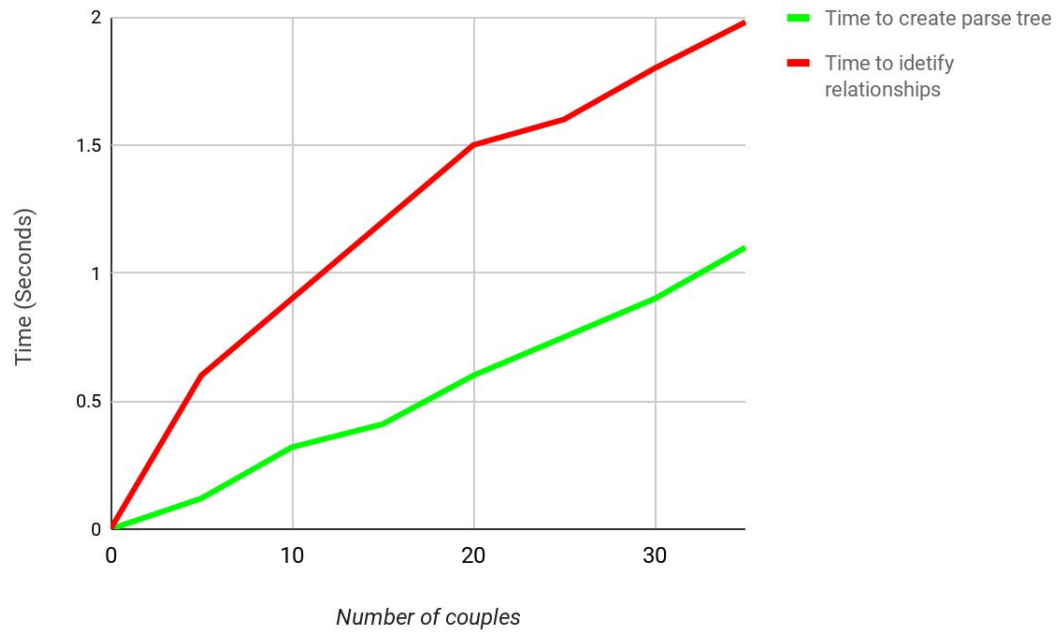


Image 5.4.1 Parse tree vs algorithm relational time comparison

6 . Conclusion

6.1 Introduction

This research introduces a new novel algorithm “Space algorithm” which can be used to build the relationships between segments in the invoices. This achieved 98.8% accuracy which was very good for automatic solution.

6.2 Conclusions about research questions (aims/objectives)

Two research questions are tested in this research and one of them is less profitable in practical scenarios. First , “ How effectively can we write grammar to invoices and create a layout parse tree to classification?” . As mentioned in the results section **“Time to create parse tree” > “Time to build relationships”**. It was more than one second higher. It was more effective rather than classifying , doing direct relationship building.

Second, “ Can we use Spaces between segments to create logical structure?” .Space algorithm used to build logical structure. Space algorithm uses spaces as primary input and most calculations use them (Even pre processing data) . This was achieved by higher accuracy with the algorithm.

6.3 Limitations

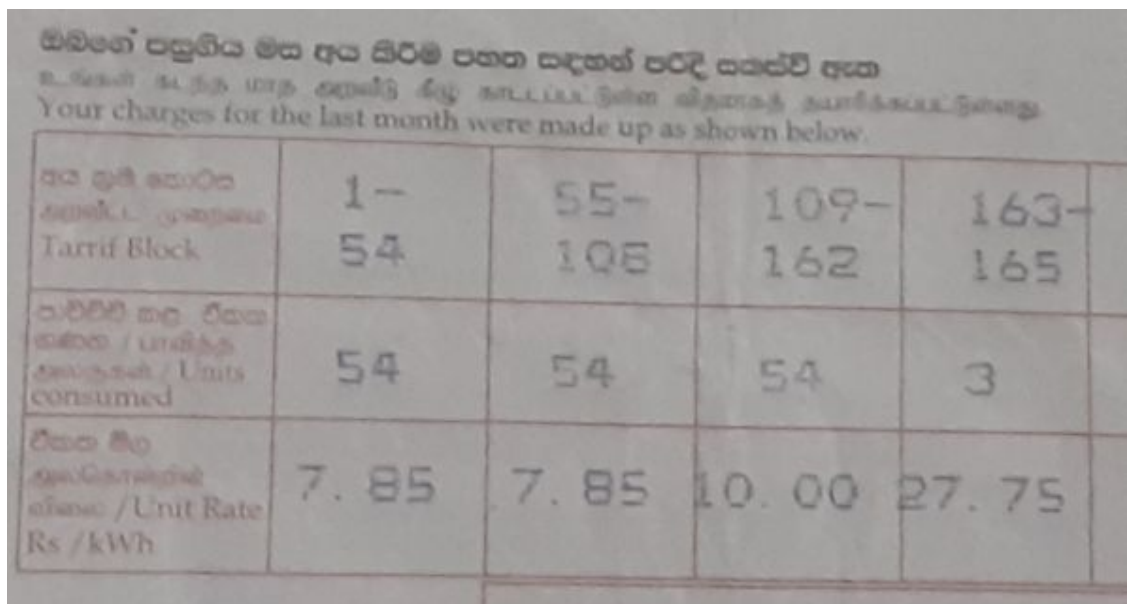
Some invoices contain the tables. They can be single row or multiple rows. This research only covers the PDF invoices without a table and with a single row table. Only consider “Spaces” “Numbers” and “Special Characters” as input to the algorithm and not used “Fonts” and “Font sizes”. Multi language invoices are not handled in this iteration.

6.4 Implications for further research

Three scenarios are introduced as future research improvements. First invoices are different from organization to organization. Some government invoices use multiple languages within the same invoice (Image 6.5.1). To handle those situations need to improve the methodology of the algorithm and need to be combined with the Natural Language Processing activities. Because the system needs to understand that the same word is written using different languages with the invoice.

Second, Invoices come with different fonts and different font sizes. Most of the time keywords have different font and big font size than the values. Same situation in the tables. Column names have different font or big font size. In this iteration of the algorithm , only consider Spaces , Numbers and special characters. To improve the accuracy can provide fonts and font sizes as input to the algorithm.

Third, some invoices contain very complex tables with very complex table structures. Those are not handling this iteration.



සිංහල බලශක්ති මධ්‍යස්ථානයේ මාසික බිල්පත සඳහා වූ පැහැදිලි කිරීම				
உலகம் உருவாகிய மாத அளவில் கிடைத்த மின்சாரத் தரவரிசைகள் பின்வருமாறு				
Your charges for the last month were made up as shown below.				
අය කළ මාරු කොටස அடைபட்ட முடிவுகள் Tariff Block	1- 54	55- 108	109- 162	163- 165
පාවිච්චි කළ විද්‍යුත් ශක්තිය / පාවිච්චි කළ ඒකක / Units consumed	54	54	54	3
විද්‍යුත් මිල අයදුම්කරුගේ මිල / Unit Rate Rs / kWh	7.85	7.85	10.00	27.75

Image 6.4.1 Multilingual invoice

References

01. Warnock, J. (n.d.). The Camelot Project. [online] Available at: http://www.eprg.org/G53DOC/pdfs/warnock_camelot.pdf.
02. Unixuser.org. (2014). PDFMiner. [online] Available at: <http://www.unixuser.org/~euske/python/pdfminer/> [Accessed 5 Jan. 2020].
03. Diligenti, M., Frasconi, P. and Gori, M. (2003). Hidden tree markov models for document image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, [online] 25(4), pp.520–524. Available at: <https://ieeexplore.ieee.org/abstract/document/1190578>.
04. Rusinol, M., Benkhelfallah, T. and dAndecy, V.P. (2013). Field Extraction from Administrative Documents by Incremental Structural Templates. 2013 12th International Conference on Document Analysis and Recognition. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/6628784>.
05. Adobe system Incorporated . Pdf reference 1.7 Available at : https://www.adobe.com/content/dam/acom/en/devnet/pdf/pdfs/PDF32000_2008.pdf
06. Chao, H. and Fan, J. (2004). Layout and Content Extraction for PDF Documents. Document Analysis Systems VI, pp.213–224.
07. Acm.org. (2020). Identifying table boundaries in digital documents via sparse line detection | Proceedings of the 17th ACM conference on Information and knowledge management. [online] Available at: <https://dl.acm.org/citation.cfm?id=1458255>.
08. Acm.org. (2015). Automatic extraction of table metadata from digital documents | Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries. [online] Available at: <https://dl.acm.org/citation.cfm?id=1141835>.
09. Acm.org. (2015). TableSeer | Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries. [online] Available at: <https://dl.acm.org/citation.cfm?id=1255193>.

10. Lu, X., Kataria, S., Brouwer, W.J., Wang, J.Z., Mitra, P. and Giles, C.L. (2009). Automated analysis of images in documents for intelligent document search. *International Journal on Document Analysis and Recognition (IJ DAR)*, [online] 12(2), pp. 65–81. Available at: <https://clgiles.ist.psu.edu/pubs/IJCAR2009-image-analysis-search.pdf> .
11. H. D'éjean and J.-L. Meunier (2006) A system for converting PDF documents into structured XML format.' In *DAS '06. 7th IAPR Int'l Workshop on Document Analysis Systems*, pages 129–140, 2006.
12. K. Hadjar, M. Rigamonti, D. Lalanne, and R. Ingold. Xed:'a (2004) new tool for extracting hidden structures from electronic documents'. In *DIAL '04. First Int'l Conference on Document Image Analysis for Libraries*, pages 212–224
13. Marinai, S. (2009). Metadata Extraction from PDF Papers for Digital Library Ingest. 2009 10th International Conference on Document Analysis and Recognition. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/5277711> .
14. Chanda, G. and Dellaert, F. (2011). Grammatical Methods in Computer Vision: An Overview. *Gatech.edu*. [online] Available at: <https://smartech.gatech.edu/handle/1853/3738> .
15. Kise, K., Yajima, N., Babaguchi, N. and Fukunaga, K. (2020). Incremental acquisition of knowledge about layout structures from examples of documents. *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR '93)*. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/395649> .
16. Nagy, G., Seth, S. and Viswanathan, M. (1992). A prototype document image analysis system for technical journals. *Computer*, [online] 25(7), pp.10–22. Available at: <https://ieeexplore.ieee.org/abstract/document/144436> .

Appendix A: Publications

Appendix C: Code Listings

The `getXYValues` used to retrieve the related members of the one segment. This used the parameter word index which was the index of the main member.

```
private static String[][] getXYValues(int wordIndex) {
    String p[] = new String[6];
    int arrayIndex = 0, r = 1;
    int wordLocatorX;
    float xCoordinate = positionList.get(wordIndex).getXCoordinate();
    float yCoordinate = positionList.get(wordIndex).getYCoordinate();
    p[0] = positionList.get(wordIndex).getFullWord();
    for (wordLocatorX = wordIndex + 1; wordLocatorX < positionList.size() - 1;
wordLocatorX++) {
        if (positionList.get(wordLocatorX).getIsFullWord() == 1) {
            float xXCoordinate =
positionList.get(wordLocatorX).getXCoordinate();
            float xYCoordinate =
positionList.get(wordLocatorX).getYCoordinate();
            if (yCoordinate == xYCoordinate & (xXCoordinate - xCoordinate) <
pageWidth / 2.5) {
                p[r] = positionList.get(wordLocatorX).getFullWord();
                break;
            } else {
                break;
            }
        }
    }
    int counterX = 0;
    for (String aP : p)
        if (aP != null) {
            counterX++;
        }
    String values[][] = new String[4][10];
    if (counterX % 2 == 0) {
        for (int c = 0; c < counterX; c++) {
            values[arrayIndex][0] = p[c];
            values[arrayIndex][1] = p[c + 1];
        }
    }
}
```

```

values[arrayIndex][2] = "" +
positionList.get(wordIndex).getYCoordinate();
values[arrayIndex][3] = "" +
positionList.get(wordIndex).getXCoordinate();
values[arrayIndex][4] = "X";
values[arrayIndex][5] = "" +
positionList.get(wordLocatorX).getYCoordinate();
values[arrayIndex][6] = "" +
positionList.get(wordLocatorX).getXCoordinate();
c++;
arrayIndex++;
}
}
String py[] = new String[6];
int ry = 1;
int wordLocatorY;
py[0] = positionList.get(wordIndex).getFullWord();
for (wordLocatorY = wordIndex + 1; wordLocatorY < positionList.size() - 1;
wordLocatorY++) {
    if (positionList.get(wordLocatorY).getIsFullWord() == 1) {
        float yYCoordinate =
positionList.get(wordLocatorY).getYCoordinate();
        float yXCoordinate =
positionList.get(wordLocatorY).getXCoordinate();
        if (yXCoordinate == xCoordinate & (yYCoordinate - yCoordinate) <
40) {
            py[ry] = positionList.get(wordLocatorY).getFullWord();
            break;
        }
    }
}
int counterY = 0;
for (String aPy : py) {
    if (aPy != null) {
        counterY++;
    }
}
if (counterY % 2 == 0) {
    for (int c = 0; c < counterY; c++) {
        values[arrayIndex][0] = py[c];
        values[arrayIndex][1] = py[c + 1];
    }
}

```

```

values[arrayIndex][2] = "" +
positionList.get(wordIndex).getYCoordinate();
values[arrayIndex][3] = "" +
positionList.get(wordIndex).getXCoordinate();
values[arrayIndex][4] = "Y";
values[arrayIndex][5] = "" +
positionList.get(wordLocatorY).getYCoordinate();
values[arrayIndex][6] = "" +
positionList.get(wordLocatorY).getXCoordinate();
c++;
arrayIndex++;
}
}
String[][] subArray = new String[arrayIndex][10];
for (int i = 0; i < arrayIndex; i++) {
    subArray[i] = Arrays.copyOfRange(values[i], 0, 9);
}
return subArray;
}

```

Following code images show the code for the “Space algorithm”. Which changed the values pre assigned according to the differences.

```

private int valChanger(Float id) {
    Node node = hashMapWords.get(id);
    if (node.getRightId() != 0 & node.getBottomId() == 0) {
        if (node.getTopId() != 0) {
            hashMapWords.get(node.getTopId()).setTotalPro(hashMapWords.get(node.getTopId(
            )).getTotalPro() + 1);
            if (hashMapWords.get(node.getRightId()).getTopId() != 0) {
                node.setUsed();
                node.setKeyWord();
                hashMapWords.get(node.getRightId()).setUsed();
                hashMapWords.get(node.getRightId()).setNotKeyWord();
                hashMapRelations.put(node.getValue(),
            hashMapWords.get(node.getRightId()).getValue());

```

```

hashMapWords.get(hashMapWords.get(node.getRightId()).getTopId()).setTotalPro(
hashMapWords.get

(hashMapWords.get(node.getRightId()).getTopId()).getTotalPro() - 1);

        return 0;
    }
} else {
    node.setUsed();
    node.setKeyWord();
    hashMapRelations.put(node.getValue(), "");

hashMapWords.get(node.getRightId()).setTotalPro(hashMapWords.get(node.getRigh
tId()).getTotalPro() + 1);

        return valChanger(node.getRightId());
    }
    return 0;
} else if (node.getRightId() == 0 & node.getBottomId() != 0) {
    node.setUsed();
    node.setKeyWord();
    hashMapWords.get(node.getBottomId()).setUsed();
    hashMapWords.get(node.getBottomId()).setNotKeyWord();
        hashMapRelations.put(node.getValue(),
hashMapWords.get(node.getBottomId()).getValue());
    return 0;
}
    if ((!node.getUsed()) & node.getRightId() != 0 & node.getBottomId() != 0)
    {
        if (node.getTotalPro() <
hashMapWords.get(node.getRightId()).getTotalPro()) {
            return 0;
        }
        if (hashMapWords.get(node.getRightId()).getTotalPro() <
hashMapWords.get(node.getBottomId()).getTotalPro()) {
            //System.out.println("Keyword : " + node.val+ "\tValue :"+
node.totalPro);
            if (node.getTotalPro() >
hashMapWords.get(node.getRightId()).getTotalPro()) {
                node.setUsed();
                node.setKeyWord();
                hashMapWords.get(node.getRightId()).setUsed();
                hashMapWords.get(node.getRightId()).setNotKeyWord();

```



```

                                hashMapRelations.put (node.getValue(),
hashMapWords.get (node.getRightId()).getValue());

hashMapWords.get (node.getBottomId()).setTotalPro (hashMapWords.get (node.getBottomId()).getTotalPro() + 1);
        if (hashMapWords.get (node.getRightId()).getBottomId() != 0) {

hashMapWords.get (hashMapWords.get (node.getRightId()).getBottomId()).setTotalPro (hashMapWords.get
ro (hashMapWords.get

(hashMapWords.get (node.getRightId()).getBottomId()).getTotalPro() - 1);
        }
        return valChanger (node.getBottomId());
    } else {
        return 0;
    }
    } else if (hashMapWords.get (node.getRightId()).getTotalPro() ==
hashMapWords.get (node.getBottomId()).getTotalPro()) {
        return valChanger (node.getBottomId());
    } else {
                                if      (node.getTotalPro()      >
hashMapWords.get (node.getBottomId()).getTotalPro()) {
        node.setUsed();
        node.setKeyWord();
        hashMapWords.get (node.getBottomId()).setUsed();
        hashMapWords.get (node.getBottomId()).setNotKeyWord();
                                hashMapRelations.put (node.getValue(),
hashMapWords.get (node.getBottomId()).getValue());

hashMapWords.get (node.getRightId()).setTotalPro (hashMapWords.get (node.getRightId()).getTotalPro() + 1);
        if (hashMapWords.get (node.getBottomId()).getRightId() != 0) {

hashMapWords.get (hashMapWords.get (node.getBottomId()).getRightId()).setTotalPro (hashMapWords.get
ro (hashMapWords.get

(hashMapWords.get (node.getBottomId()).getRightId()).getTotalPro() - 1);
        }
        return valChanger (node.getRightId());
    } else {
        return 0;
    }
}

```

```
    }  
}  
return 0;  
}
```