: N° d'ordre
: N° de série

PEOPLE DEMOCRACIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID HAMMA LAKHDAR - EL OUED

FACULTY OF EXACT SCIENCES

Computer Science Department

End of Study Thesis
Presented for obtaining the Diploma of

ACADEMIC MASTER

Field : **Mathematics and Computer Science**
Sector : **Computer Science**
Specialty : **Distributed Systems and Artificial Intelligence**

Presented by :

- LAOUID Ait Allah Zeineb
- LABBI Radhia

Theme

Document Errors Verification System

Sustained on : - - 2024 in front of the jury:

Dr.		MAA	President
Dr.		MAA	Protractor
Dr.	ZAIZ FAOUZI	MAA	Supervisor

College Year : 2023/2024

Dedication

First, we dedicate our dissertation work to the sake of ***Allah****, our Creator and our Master. Our great teacher and messenger, Mohammed (May* ***Allah*** *bless and grant him), who taught us the purpose of life.*

we dedicate our dissertation to all my family. A special feeling of gratitude to our loving parents, whose words of encouragement and push for tenacity ring in our ears. Our sisters and our brothers have never left our sides and are very special.

We also dedicate this dissertation to all friends who have supported us throughout the process. We will always appreciate all they have done.

May ***Allah*** *grant them Jannah Firdaus. Ameen*

LABBI Radhia , LAOUID Zeineb

Acknowledgement

Prima facie, we are grateful to ***Allah*** *for the guidance, good health, wellness and willpower that were necessary to complete this thesis.*

We would like to thank ***Our Parents****, who always believed in us.*

It is thanks to their support and prayers that we accomplished this work; they already know how much we owe them.

We also would like to thank our supervisor, Professor ***ZAIZ Faouzi****, whose expertise was invaluable in formulating the research methodology.*

Your insightful feedback pushed us to sharpen our thinking and brought our work to a higher level.

We also place on record our sense of gratitude to one and all who directly or indirectly have lent their hand in this venture.

Finally, we also want to thank the jurors for agreeing to review and judge our work.

Abstract

The majority of documents are made out according to predetermined rules, which regulate requirements to design of the given class of documents. However, the process of paper checking is quite time-consuming and requires from an expert person of high concentration and appropriate qualification.

The work provides the main methods that reduce the effort required to control the logical and physical structure of the graduation document (thesis). Therefore, we will work to ensure that there are no organization and formatting errors in the texts, paragraphs, and titles. We will also process images and formalities by checking their quality, as algorithms can lead to different categories of errors.

To automate the verification of formal correctness of the document, the system designed for the compliance assessment of documents in accordance with the parameters set by the user was developed. The functionality of the system was described and the tools used in its development were presented

Keywords: Document, Logical structure, Physical stucture, Verification.

Résumé

La majorité des documents sont élaborés selon des règles prédéterminées, qui régissent les exigences en matière de conception de la classe de documents donnée. Cependant, le processus de vérification des documents est assez chronophage et nécessite l'intervention d'une personne experte ayant une grande concentration et une qualification appropriée.

L'ouvrage fournit les principales méthodes qui réduisent l'effort requis pour contrôler la structure logique et physique du document de fin d'études (thèse). Par conséquent, nous veillerons à ce qu'il n'y ait aucune erreur d'organisation et de formatage dans les textes, paragraphes et titres. Nous traiterons également les images et les formalités en vérifiant leur qualité, car les algorithmes peuvent conduire à différentes catégories d'erreurs.

Pour automatiser la vérification de la correction formelle du document, un système conçu pour l'évaluation de la conformité des documents selon les paramètres définis par l'utilisateur a été développé. La fonctionnalité du système a été décrite et les outils utilisés dans son développement ont été présentés.

Mots Clés: Document, Structure logique, Structure physique, Vérification.

الملخص

تعتمد معظم الوثائق على قواعد محددة مسبقاً، تنظم متطلبات تصميم الفئة المعطاة من الوثائق. ومع ذلك، فإن عملية فحص الأوراق تستغرق وقتاً طويلاً وتتطلب من الشخص الخبير تركيزاً عالياً ومؤهلات مناسبة.

يوفر العمل الطرق الرئيسية التي تقلل الجهد المطلوب للتحكم في البنية المنطقية والمادية لوثيقة التخرج (الأطروحة). لذا سنعمل على ضمان عدم وجود أخطاء تنظيم و تنسيق في النصوص والفقرات والعناوين. كما سنقوم بمعالجة الصور والشكليات بالتحقق من جودتها، حيث يمكن أن تؤدي الخوارزميات إلى فئات مختلفة من الأخطاء.

لتحقيق التحقق التلقائي من الصحة الشكلية للوثيقة، تم تطوير النظام المصمم لتقييم الامتثال للوثائق وفقاً للمعايير التي حددها المستخدم. تم وصف وظائف النظام وعرض الأدوات المستخدمة في تطويره.

كلمات مفتاحية: مستند، البنية المادية ، البنية الفيزيائية، فحص .

Contents

List of Figures

List of Tables

Acronyms

ML:	Machine Learning.
PDF:	Portable Document Format.
DOC:	Document.
DOCX:	Document XML.
HTML:	HyperText Markup Language.
HTM:	HyperText Markup.
XLS:	Excel Spreadsheet.
XLSX:	Excel Spreadsheet (Open XML).
TXT:	Text.
AI:	Artificial Intelligence.
WPF:	Windows Presentation Foundation.
IT:	Information Technology.
RIQA:	Reference Image Quality Assessment.
AUROC:	Area Under the Receiver Operating Characteristic curve.
WT:	Wavelet Transform.
OCR:	Optical Character Recognition.
NLP:	Natural Language Processing.
HVS:	Human Visual System.
IDE:	Integrated Development Environment.

Introduction

Conformity of the document to the formatting rules is one of the priority requirements for documents reflecting the results of literary, scientific and technical activities. Uniformity of appearance and structure of text documents is mandatory for any field of knowledge since such standardization allows easier understanding of information provided. In addition to that, strict adherence to standards simplifies the process of storing and processing documents in databases [19].
Due to the fact that the key role in manual check of formatting rules is played by a specialist for whom increased attentiveness and special qualifications are compulsory, this process can be extremely time consuming and inefficient in terms of time expenses. For this reason, automation of the documents check seems to be a highly relevant issue. The first and fundamental step in document recognition tasks is to detect specific types of regions on the page (such as tables, figures, photographs, and text blocks).
In this work, we will consider the prevalent errors found in the title and citations which constitute a level of errors that are very common but rarely addressed. This type of errors includes organizational and formatting errors.
Poor-quality document images often lead to decreased readability. Therefore, we have developed an artificial intelligence model that evaluates the quality of images and classifies them as either good or poor. Also it evaluates images position then deciding if it is in right position.

Considering all the previously existing works, this work is comprehensive because it works on different areas within documents.

Problematic

Universities face challenges in managing large quantities of student theses on a daily basis, leading to the need for an efficient system to examine and audit the formatting and organization of these thesis.
The system we have developed aims to detect a wide range of common errors made by students when writing their theses, improving their quality and organizing them

uniformly. This facilitates the academic evaluation and review process.

Purpose of work

The purpose of this work is the development of a system of text document check according to specified formatting rules. Such system will significantly reduce the amount of time and effort required for document analysis in comparison with the manual check.
In addition, the system provides specific guidance and detailed reports to students about detected errors, which enhances the development of their writing and coordination skills.

Thesis Organization

Our study is divided into three sections:
The first chapter attempts to introduce document definition and its purpose, also, describes the different types and structures of documents and some concepts connected to work. Finnaly, It presents some previous works related to the study.

The second chapter presents various common mistakes made by students while writing their theses. We have categorized these mistakes into two categories: logical and physical. Then, we identified a list of errors that the system will address. Additionally, we have displayed all the diagrams of this system and explained them in detail.

The third chapter present tools, environments and techniques used to build the model among them we find Support Vector Machines (SVM), Random Forest(RF), Logistic Regression(LR) and CatBoost. This chapter it also presents the result of the system after its development, with an explanation of its important functions.

We conclude the work with a conclusion about the results obtained through the methods used, the limitations of this system and some future work.

Chapter 1

Document Analysis

Introduction

Nowadays, digital documents have become an integral part of our daily lives, whether in work, study, or personal activities. With the increasing volume of data and information flow, it has become necessary to have an effective system for verifying the accuracy integrity of these documents.

This chapter aims to provide an overview of documents, their types, and structures. It will also cover the fundamental concepts of error verification systems, including the linguistic processing techniques and artificial intelligence used in this field. Additionally, some previous works similar to this study will be mentioned.

1.1 Document Definition

When talking about the term document, many meanings and definitions come to mind such as :

- Document is a collection of data regardless of a medium on which is it recorded [24].
- Document is a written or printed paper furnishing information [6].
- Document include both paper and electronic [24].
- Bachimont considers that the document is inseparable from a material support. In fact, "a document is a physical object that expresses information." The inscription medium—through which content is expressed—is the tangible object. The body of knowledge and information to be expressed is called content [15].

1.2 Electronic Document

A document is considered digital (electronic) when its physical support is digital, according to Bachimont and Hadjar's definitions. An electronic document is any document that may be communicated between computers that is stored in memory or on a computer medium as a data structure or sequence of bytes [15].

1.3 Purpose of Document

A document's main goal is to make it easier for readers to receive information from its creator. It is the author's responsibility to ensure that the information in the document can be understood clearly and effectively [13].

1.4 Document Types

We may use text files when sending documents to our teachers or classmates. Here are some of the most popular text file formats you might use:

1. **Portable document format (PDF):** In many workplaces, PDF files are a standard file type. This file is useful for sharing plans and signing documents because it preserves the original document layout. This file type can be used for scanning, printing, and emailing. This option is widely used for sending or uploading resumes because it preserves the formatting and original layout of the document [25].
2. **Word document (DOC and DOCX):** Microsoft established this file format, which is the default file type for documents stored in word processing programs [25].
3. **Hypertext markup language (HTML and HTM):** HTML files are used by developers and other experts who create websites or web content. This kind of file transforms text into different website elements while operating online [25].
4. **Microsoft excel spreadsheet file (XLS and XLSX):** These are typical file formats that you may encounter at work. You can store and share files in this format if you exchange databases, graphs, or spreadsheets. Spreadsheets can be used to measure and monitor a wide range of data [25].
5. **Text file (TXT):** Simple text documents can be opened in TXT files. You can write ordinary text, instructions, or notes in a TXT. This file is compatible with a variety of systems and processing applications [25].

1.5 Concept of structure

Broadly speaking, According to the "LE PETIT ROBERT dictionary" : "Structure describes the way a building is constructed; in architecture, for example, it refers to

the arrangement of a building's components." When it comes to documents, structure refers to how the content is arranged into blocks, levels, and other sections as well as how those elements relate to one another. But in most cases, this arrangement results in two levels of structures: logical structure and physical structure. [15].

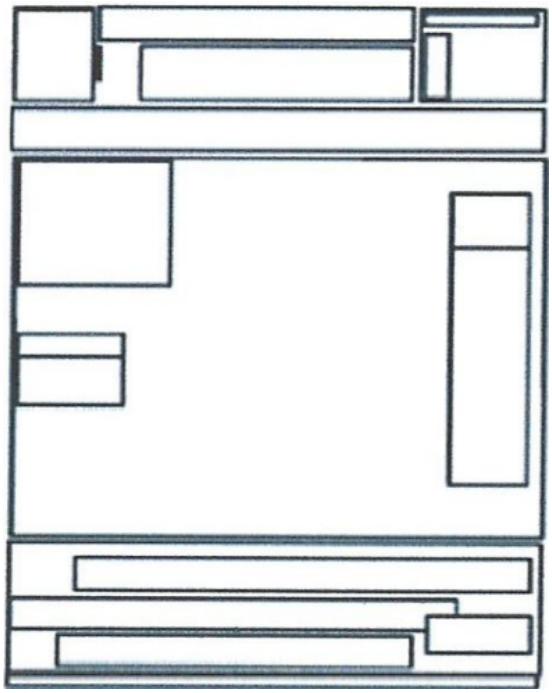
1.5.1 Physical structure

Defines the layout of the document on paper., such as its formatting, and visual elements, page size, margins, fonts, colors, spacing, and alignment [15].
The physical structure deals with the actual presentation of content on the page or screen, focusing on aesthetics and readability. Elements of the physical structure include fonts, colors, images, borders, indentation, columns, and other design elements.

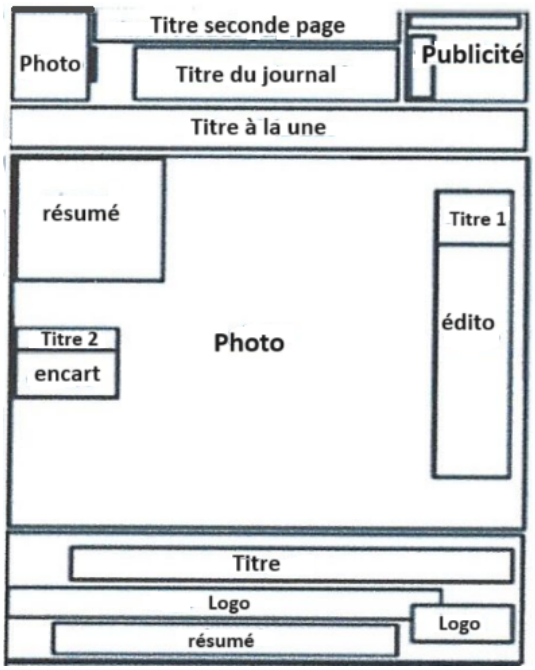
1.5.2 Logical structure

Lgical structure defines the hierarchical organization of the information contained and presented within a document, the flow of information [15], and the relationships between different sections or components of the document.
Elements that contribute to the logical structure include headings, subheadings, paragraphs, lists, tables of contents and indices.

Figure 1.1: Newspaper Page [15].



(a) Phisical Structure



(b) Logical Structure

Figure 1.2: Physical and Logical structure of Newspaper page [15].

1.6 Artificial Intelligence

The ability of a computer or a robot under computer control to perform tasks that are typically performed by humans because they call for human intelligence and judgment is known as artificial intelligence (AI). While no AI can accomplish the vast range of jobs that a typical human can, some AIs are comparable to humans in particular tasks [2]. Machine learning and deep learning are two closely related subfields of artificial intelligence that are frequently mentioned:

1.6.1 Machine Learning

Artificial intelligence (AI) in the form of machine learning (ML) makes it possible for software programs to improve their prediction accuracy without having to be specifically designed to do so. In order to forecast new output values, machine learning algorithms use historical data as input [10].

1.6.2 Deep Learning

A technique in artificial intelligence (AI) called deep learning trains machines to analyze information in a manner modeled after the human brain. To generate precise insights and forecasts, deep learning models are able to identify intricate patterns in images, text, sounds, and other types of data. Deep learning techniques can be used to automate processes that normally need human intellect, including text transcription from audio files or picture descriptions. [3].

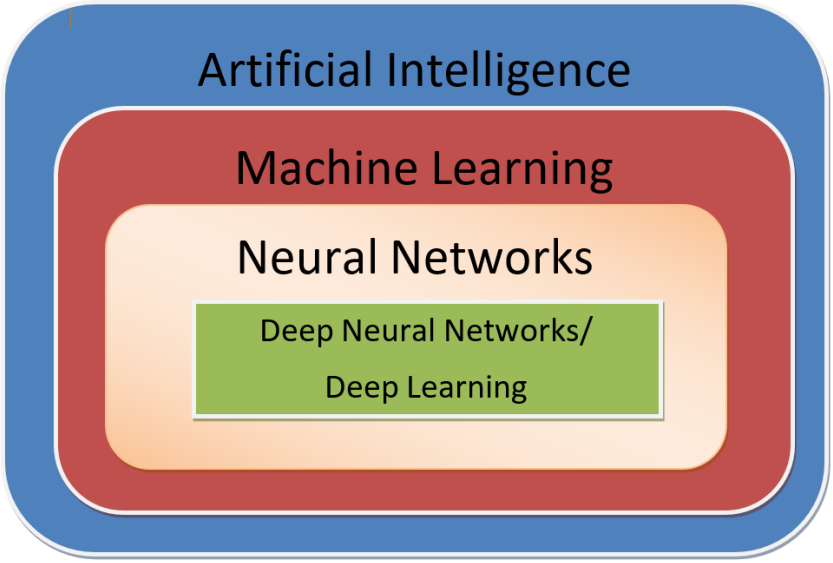


Figure 1.3: Artificial Intelligence Family

1.7 Errors detection approaches

There are many established approaches for detecting errors in documents. Among the most commonly used approaches, which have been previously implemented and proven effective in handling text errors, are:

1. **Spell checking :** is one of the most widely used tasks of NLP. It has broad range of uses like information retrieval, proofreading, email client etc. Today in many applications of NLP spell checker is being used. It is a language tool which breaks down the text for spelling errors. It flags when there exists any misspelled/unintended word in the given text. The typographic errors are categorized in 'non-word errors' and 'real-word errors'. There is enough work done in order to tackle the farmer error but it still remains the challenge for the researchers to tackle the later one. This paper focuses on later one and analyses the methods which are being used worldwide to detect and correct such errors. Paper also focuses on the challenges faced by researchers while processing the real-word errors [20].

2. **Dictionary :**

- Construction of Dictionaries: The process begins by assembling lists of words and other linguistic elements relevant to the language and domain of the OCR task. These lists include: Given names, surnames, and initials: These lists contain common personal names, both first names and last names, as well as initials (capital letters A through Z). Person titles: Titles such as "Mr" and "Jr" are included in the dictionaries. Days of the week and months: Lists of names and abbreviations for days of the week and months are compiled.

- Common English words: A list of common English words is selected from multiple sources.

- Testing and Evaluation: The performance of each dictionary is evaluated using an error-labeled training set. Performance metrics are assessed to determine which dictionary performs best in identifying OCR errors.

- Inclusion of Numeral Patterns and Punctuation: In addition to words, dictionaries are constructed for numeral patterns (1-, 2-, and 4-digit numerals) and punctuation characters (. , ; + "). The inclusion of these elements is based on their presence in the OCR data and their impact on performance when added to the dictionary.

- Case-Insensitive Matching: Matches between OCR output and dictionary entries are checked in a case-insensitive manner to account for variations in letter case.

- Performance Evaluation and Refinement: The performance of the dictionaries is continually evaluated, and entries that improve performance on the training data are

retained, while others are removed. This iterative process helps refine the dictionaries for optimal error detection accuracy [20] .

1.8 Previous Works

Previous works address some of the components included in our system. We have summarized the content of these works by extracting a summary and mentioning their methodology.

1.8.1 Textual and Stylistic Error Detection and Correction: Categorization, Annotation and Correction Strategies

In this work, they analyze common stylistic and structural errors encountered in texts authored by various authors. We then demonstrate the annotation of these errors and their corresponding corrections, leading to the induction of correction rules through generalization. As rectifying errors entails complexity with multiple potential solutions. we propose the application of an argumentation system, this system assists users in evaluating arguments supporting or opposing specific corrections, enhancing decision-making regarding error rectification [18].

To achieve these aims they need to produce a model of the cognitive strategies deployed by human experts (e.g. language teachers) when they detect and correct errors. Our observations show that it is not a simple and straightforward strategy, but that error diagnosis and corrections are often based on a complex analytical and decisional process. Since we want our system to have a didactic capacity, to help writers understand their text errors, we propose an analysis of error diagnosis based on argumentation theory, outlining arguments for or against a certain correction and their relative strength paired with a decision theory [18].

1.8.2 Algorithms and software for verification of scientific and technical text documents

This work proposes a solution for verifying the formatting of scientific and technical documents to ensure compliance with regulatory requirements, referred to as the document verification problem. It utilizes style analysis of the Word text editor to assess the document's elements such as headings, annotations, main text, figures, and references against predefined styles. The system allows for the creation, editing, and management of style sets by administrators. Algorithms were developed to analyze document structure including headers, footers, and main text paragraphs. Implementation was done using .Net, WPF, and DocumentFormat.OpenXml technologies, enabling analysis of .doc/.docx format files. The program outputs analysis results in .txt or .doc/.docx

formats, highlighting deviations from specified styles. It simplifies the document checking process, reducing time and resources required. The user interface was developed using .Net and WPF. The program was tested with real bachelor's and master's theses, demonstrating analysis times of under 3 seconds. It proves beneficial for automating document checking, ensuring compliance with design standards in scientific and technical publications, particularly in educational settings for verifying qualification works and reports [27].

To analyze the design of technical and scientific texts, algorithms for their analysis have been developed and implemented using .Net, WPF, DocumentFormat.OpenXml technology. Based on algorithms, software has been developed that automates the process of checking documents and detecting inconsistencies in them. Using this approach allows you to reduce the time and resources spent on document verification, as well as to ensure their quality. The use of an automated approach to document verification will help to ensure compliance by executors of the standards for the design of technical and scientific documentation, technical and scientific publications, reporting documentation of students in the educational process, reducing time spent on documenting design results, which will contribute to the further development of technical specialties, such as computer engineering [27].

1.8.3 System of aotomated checking of textual document design rules

Text documents often adhere to specific design rules, but manual checking can be time-consuming and demanding. This paper explores methods to streamline error checking, presenting software solutions. A system has been developed for automating document compliance checks based on user-defined parameters. It includes a visual language for describing document structures, accessible to both IT specialists and non-programmers. This system is useful for verifying compliance in projects, dissertations, scientific publications, and technical documents [19].

For developing an automated system for checking textual document design rules followed a structured approach. It commenced with identifying the challenges associated with manual checking and defining the project's scope. A thorough review of existing literature guided the system's design and architecture. Development proceeded iteratively, with a focus on selecting appropriate software tools and technologies. Formatting rules were meticulously defined, informed by established standards and user preferences, while domain analysis ensured alignment with specific requirements. Implementation prioritized functionality, accompanied by the design of a user-friendly interface. Rigorous testing and validation phases ensured the system's reliability before comprehensive

documentation facilitated deployment. Continuous improvement remained integral, driven by user feedback and aimed at enhancing functionality and addressing any encountered issues [19].

1.8.4 Retinal image quality assessment based on image clarity and content

A no-reference transform-based RIQA algorithm assesses retinal images for five quality issues: sharpness, illumination, homogeneity, field definition, and content. Leveraging wavelet-based features and a retinal saturation channel, it efficiently evaluates image quality, ensuring adequate field definition and excluding non-retinal images. Tested on diverse datasets, each feature set achieves AUROC scores >0.99. A collective feature classifier demonstrates superior performance over existing methods, making it ideal for automated screening systems [17].

The proposed RIQA algorithm exploits the multiresolution char- acteristics of WT for the computation of several quality features to assess the sharpness, illumination, homogeneity, and field definition of the image. Furthermore, the retinal saturation channel and color information are used to calculate image homogeneity and outlier features, respectively. Finally, the five quality feature sets are concatenated to create the final quality feature vector that forms the input to a classifier for the overall image quality assessment. Generally, the scale of the wavelet subbands depends on image resolution. In order to ensure res- olution consistency when computing wavelet-based features, images with different resolutions were resized to 432×496 pixels. All image resizing in this work was performed using bicubic interpolation. The results of the pro- posed RIQA algorithm exceed all the methods in comparison with an **AUC of 0.927** [17] .

1.8.5 Evaluating the performance of table processing algorithms

The paper introduces simple evaluation methods for table detection and structure recognition tasks. It addresses challenges such as insertion errors, deletion errors, splitting errors, and merging errors in table detection using an edit distance approach. For table structure recognition, it proposes a model called a table DAG and introduces a new evaluation paradigm called "graph probing." This method compares the results from the recognition system with the ground-truth representation. These approaches offer practical insights into evaluating higher-level document analysis tasks, which have traditionally been complex [16].

The section outlines a method for evaluating table detection algorithms using the concept of edit distance. Assuming the input is in single-column format, tables are defined as contiguous text lines ranging from line i to line j. A document may contain multiple

non-overlapping tables listed consecutively. The recognized document (R) and the ground truth (G) are compared, where each consists of a sequence of tables. Errors in table detection include insertion (non-table regions labeled as tables), deletion (missed tables), splitting (larger tables broken into smaller ones), and merging (groups of smaller tables combined). An edit distance model is naturally suited to assess the performance of table detection algorithms considering these error types [16].

Conclusion

In this chapter, we provided a comprehensive overview of digital documents, methods of identifying errors in them, and processing them. In the next chapter, we will list the errors we have addressed and explain how the parts of the system we have developed work.

Chapter 2

SYSTEM DESIGN

Introduction

It's not easy for students to write a graduation thesis without errors due to their lack of experience in knowing the specific rules for the general format of the document. On the other hand, for an expert, reviewing students' theses and identifying errors takes a significant amount of time and effort. In this chapter, we will shed light on many of the recurring formatting and organizational errors that occur, in addition to addressing some of them. Finally, we will discuss the effectiveness of artificial intelligence in detecting these errors in the document.

2.1 Document errors types

In general, thesis document errors may belong to one of the following categories:

1. **Grammatical Errors**

   Grammatical errors are common in written documents and can significantly affect readability and professionalism. Subject-verb agreement errors occur when the subject and verb don't match in number, such as in "The students goes to the library every day" instead of the correct "The students go to the library every day." Incorrect verb tenses can change the meaning of a sentence, as seen in "I am going to the party tomorrow" when it should be "I will go to the party tomorrow." The incorrect use of articles (a, an, the) is another frequent issue. Articles are used before nouns, and the choice depends on the noun's specificity and whether it's singular or plural. For example, "I saw a elephant at the zoo yesterday" is incorrect, while "I saw an elephant at the zoo yesterday" is correct. Prepositions (in, on, at, etc.) are used to express relationships between words in a sentence, and using the wrong one can alter the meaning. "I will meet you in 6 o'clock" is incorrect; it should be "I will meet you at 6 o'clock." Lastly, word order is crucial in English, and incorrect

word order can make a sentence confusing or ungrammatical. "I to the park went yesterday" is wrong; it should be "I went to the park yesterday" or "Yesterday, I went to the park [23].

2. **Spelling Error**

   Spelling errors are another common issue in written documents. Misspelled words can make a document appear unprofessional or careless. For example, "He's a wierd person" is incorrect; the correct spelling is "He's a weird person." Another type of spelling error involves incorrectly doubled or omitted letters. Sometimes, a letter that should be doubled is not, as in "The comittee met yesterday," where the correct spelling is "The committee met yesterday," with a doubled 'm'. Conversely, letters can be omitted where they should be present. For instance, "The acount has been settled" is wrong because it's missing the 'c' in "account." The correct version is "The account has been settled." These types of errors, whether they involve misspellings, extra letters, or missing ones, can hinder comprehension and detract from the document's credibility [23].

3. **Organization and formatting errors**

   - Missing one of the needed elements, such as:
     - Forget to add an abstract page to the document.
     - Forget to add general introduction.
     - Missing introduction or conclusions in a chapter.
   - One of the elements in the wrong place, like:
     - Putting table of contents after general introduction.
   - One of the elements doesn't respect writing standards, such as:
     - Putting a website link reference without stating the date of visiting the website.
   - figure or table added in wrong place.

2.2 List of errors

The logical structure deals with the organization and flow of information conceptually,while physical structure pertains to its visual representation and appearance. Both aspects are essential for creating documents that are easy to understand, navigate, and engage with.
We have gathered a set of common errors in thesis and classified them in a table according to the physical and logical structure of the document.

Category of errors	Logical structure	Physical structure
Title page	- forgot to insert or add a title page - errors in students or committee names - wrong title	- error in page format
Abstract	- forgot to add an abstract page - no keywords (missing) - so many keywords - few keywords - wrong choice of keywords - key element of the abstract is missing - adding references errors	- abstract too long or too short
Table of contents	- element not inserted or added - element repeated - wrong title numbering	- wrong title format
List of figures	- figure not added - figure added in wrong place, or sequence	- wrong title format
List of tables	- table not added - table added in the wrong place, or sequence	- wrong title format
General introduction	- element not inserted - part of the introduction is missing - references not added	- introduction too long - intro too short (compared to document size)
Chapters	- introduction is missing - conclusion is missing	- section title with wrong format - chapters lengths imbalanced - section title too long
Text	- reference misplaced	- wrong text format
Figure	- caption without reference - caption format not correct	- figure not centred - figure too big - figure not clear (blurred, too small) - caption not centred - distance between caption and figure too big or too small - caption too long or too small
Table	- caption without reference - caption format not correct	- table not centred - table too big - table not clear (blurred, too small) - caption not centred - distance between caption and table too big or too small - caption too long or too small
General Conclusion	- element is missing - content not referenced	- conclusion too long - conclusion too short
Bibliography	- not correctly listed - reference format wrong - element is missing such as author or title or year . . . etc	

Table 2.1: Document errors according to logical and physical structure

2.3 Processing scheme of system

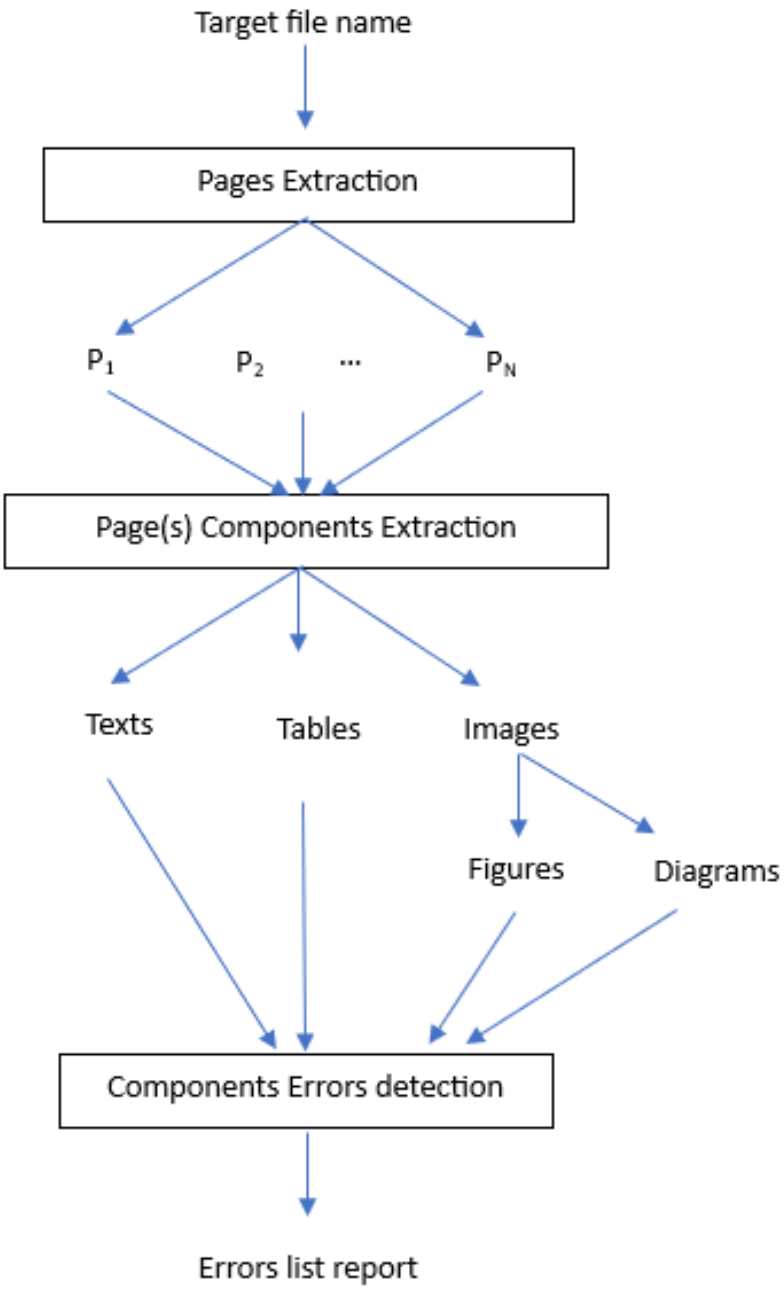


Figure 2.1: Processing Scheme

- System starts by taking a target file as input. This file format is specified in 'pdf' .
- Next, the system extracts pages from the file. There is no indication how many pages the file may contain .
- After the pages are extracted, the system then extracts components from each page, these components can include text, tables, images, figures, and diagrams.
- Once the components are extracted, the system checks them for errors.
- Finally, the system generates a report listing any errors that were found.

2.4 List of errors addressed

After collecting a large number of formatting errors, we have identified the following list that this system can detect:

1. Titles: Ckecking title format.
2. Images: Checking clarity and position.
3. Captions: checking captions existence.
4. References (Citation): Verifying citation location.

2.5 System Diagram

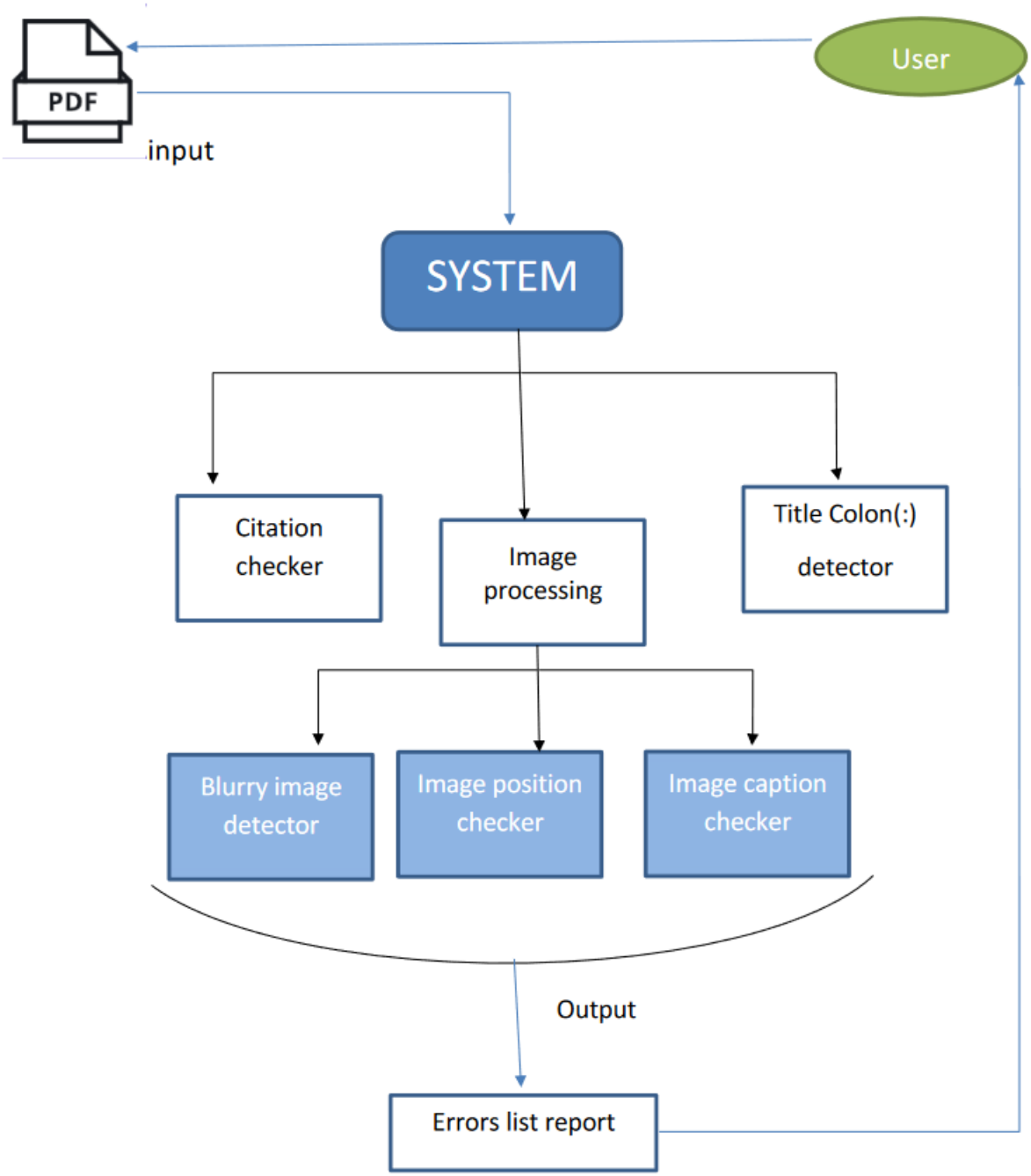


Figure 2.2: System architecture

The diagram depict a system that can detect different types of errors within a document or image. Here's a breakdown of the system's components:

- Blurry image detector: This component is responsible for detecting blurry images within a system.
- Image position checker: Designed to recognize the position of image, whether the are in center, right or left.
- Image caption checker: Designed to detect locations of captions then decide if it fits with image or not.
- Citation Ckecker: This detector might find citations within a document, citations are references to sources of information and can be found in scholarly works and research papers.
- Colon(:) detector: mentions titles that end with a colon(:).
- Errors list report: This section refers to a report that contains a list of errors detected by the system.

2.5.1 Titles

Following a specific format in writing titles is important in the coordination and organizational aspect of the document (thesis).
Our system monitors all headings in the document to see if they have the same format
To detect this error, we have the following diagram:

2.5.1.1 Colon Detector (Part One)

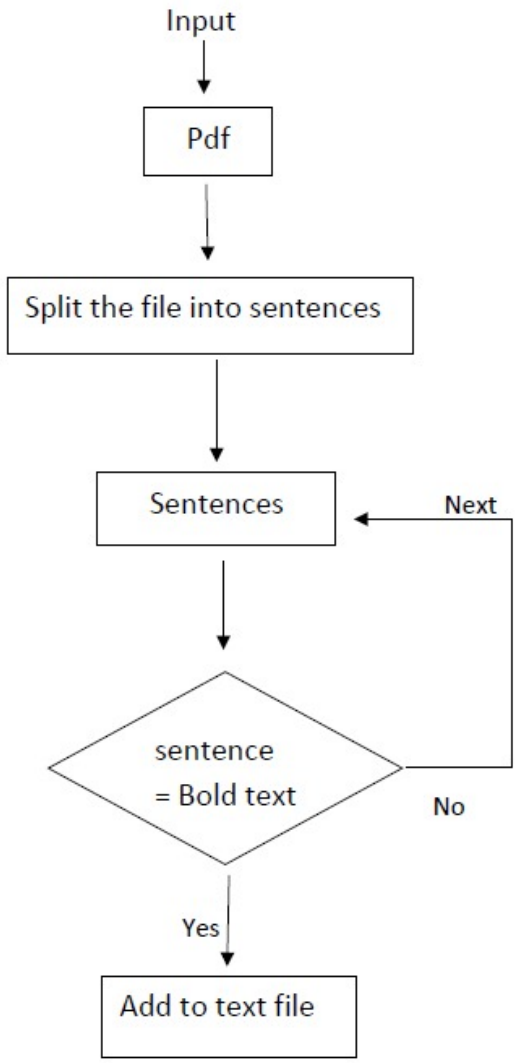


Figure 2.3: Gathering Titles Flowchart

The first part of the flowchart shows how to split a PDF file into sections based on bold text. Here's a step-by-step the way program it works (Figure 3.5):

- Input: The process starts with a PDF file being uploaded into the system.
- Split the file into sentences: The system splits the PDF file into individual sentences.
- Identify bold text: The system checks each sentence to see if it contains bold text.
- Add to text file (Yes): If the sentence contains bold text, it's added to a specific text file.
- Next sentence (No): If the sentence doesn't contain bold text, the system moves on to the next sentence.

This process continues until all the sentences in the PDF file have been processed. The output is a new text file containing all the sentences from the original PDF file that were formatted in bold.

2.5.1.2 Colon Detector (Part Two)

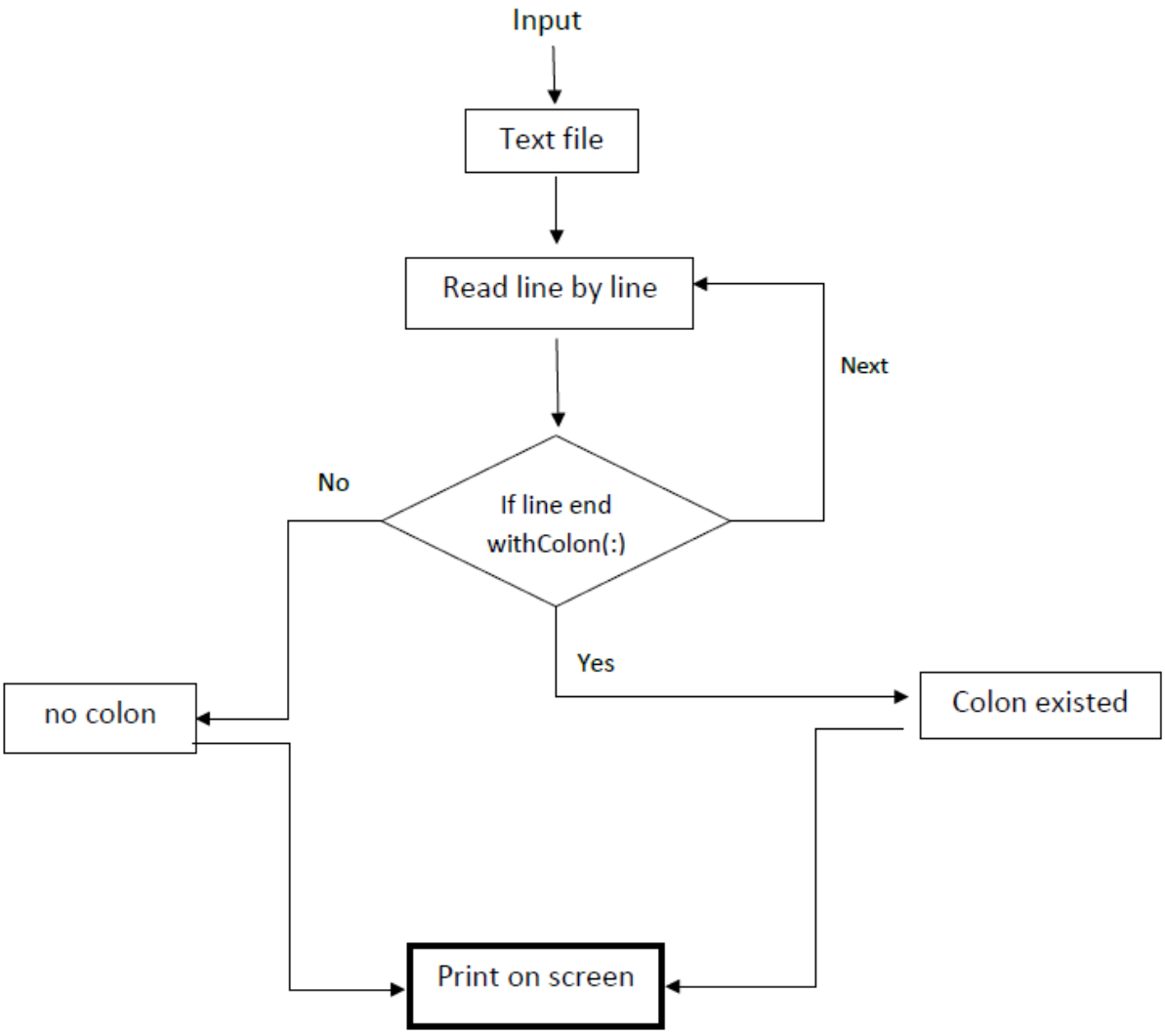


Figure 2.4: Verifying Titles Flowchart

Here's the second part of the flowchart illustrating how to read a line-by-line text file containing bolded text produced in the previous stage. This flowchart provides a step-by-step guide how to process the text file(corresponding code(Figure 3.6).

- Input: The process begins by inputting the text file.
- Read Line by Line: The program will read the file line by line.
- Check End of Line: The program checks to see if the line ends with colon or not.
- Yes (Colon (:) Exists): The sentence ends with colon.
- No (Colon (:) Doesn't Exist): The sentence doesn't end with colon.

  The "Next" arrow indicates that the system moves to the next sentence after checking the current sentence.
- Print on screen: In both cases, the system will display on the screen whether the line ends with colon or not (result of condition)

2.5.2 Images

Images are commonly described as having background and foreground information, where the foreground contains the content one is interested in analyzing. The Human Visual System (HVS) can typically differentiate these two, assuming some regions in a document image have greater relevance for human observers.
Document image readability, however, can be measured from a human or machine perspective. Human readability is estimated based on an individual‘s ability to acquire the content correctly, whereas machine readability is assessed using Optical Character Recognition (OCR) accuracy or another empirical measure.
In human and machine readability, poor-quality document images often result in low document image readability.
The next diagram shows how model is built (how to process images, and classify them whether they are blurry or not)

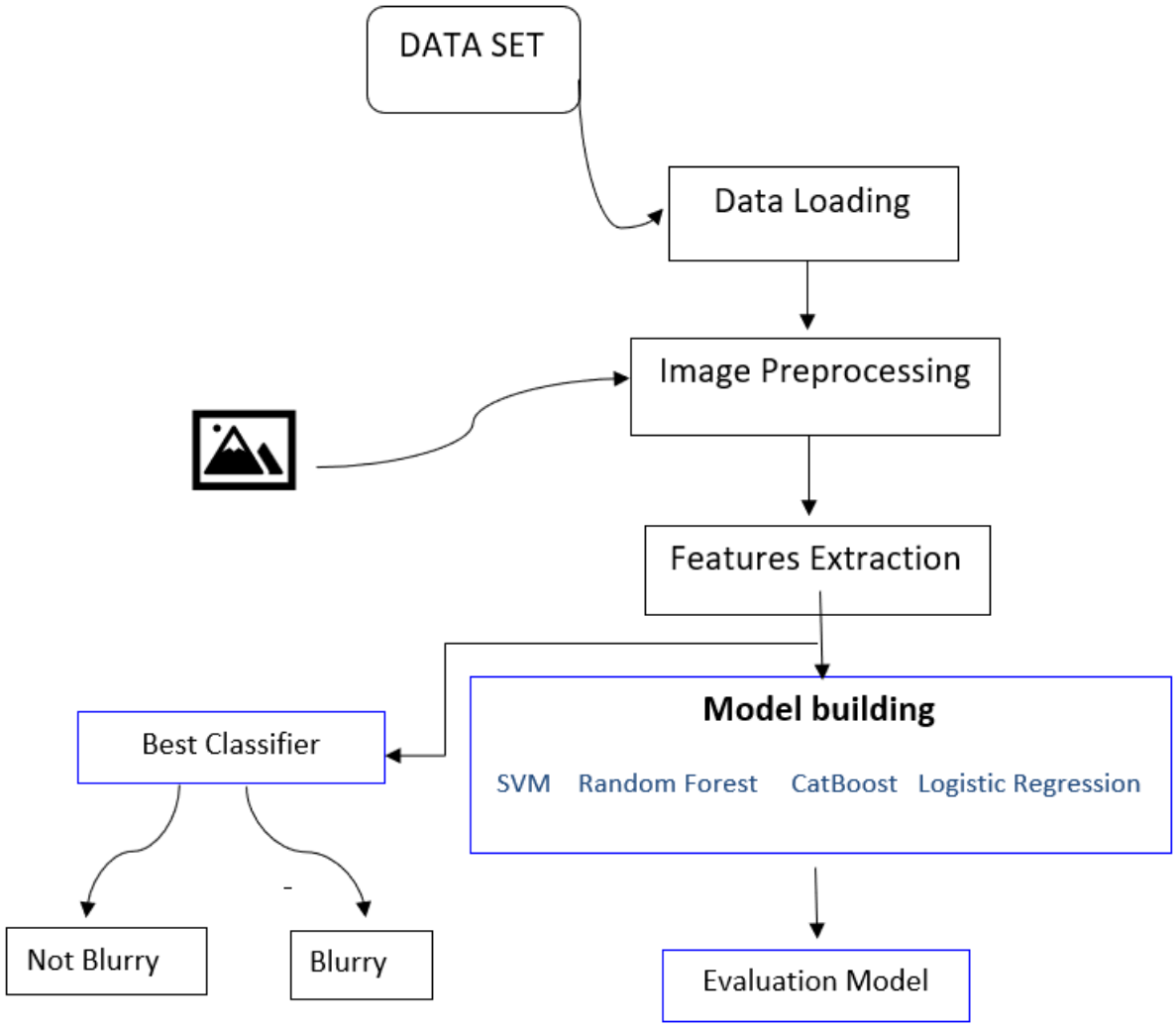


Figure 2.5: Training model using Catboost,SVM,Random Forest and Logistic Regression

1. DataSet: consists of 1664 image (864 blurred image,800 clear image)

   We extracted images from PDF collections of previous memoirs specializing in automated media and various other sources

2. Data preprocessing: This step involves preparing the data for use in the model. In the case of image data, preprocessing involves extracting the file type and reading the image using mahotas.imread.

   Convert the image to grayscale.

   Calculates Zernike moments using mahotas.features.zernike-moments.

   It writes the calculated moments to a CSV file, and classifies them as blurred (0) or clear (1).

3. Model building: Four different classifiers (SVM, Random Forest, CatBoost, Logistic Regression) are trained and evaluated. This step involves training each model on the pre-processed data and extracted features. The code we provided uses Zernike

moments which are image features. This allows us to compare the performance of different classifiers on the same data set and ensure that the best performing model is saved for use.

4. Evaluation model:

   - Testing the models on Testset

   The research was aimed at finding the highest accuracy for detecting the blurred images correctly. The module from the Scikit-learn library called 'Accuracy' helped analyse the correct classified as 'Blurry' and 'Not Blurry'. This can be measured by equation:

   $$Acc = \frac{(TN + TP)}{(TP + FN + FP + TN)}$$

   Evaluating the Dataset for training and testing data to provide better accuracy and showed improvement. This could vary on the dataset size and the information separated during the split. It should be noted that the higher the rate of training data than testing data, better the performance achieved. This is a good sign, since when considered as a real-world example, the models will have bigger weight for training data than testing. According to the experiments, have improved the accuracy of the model. CatBoost is the algorithm that has performed better than all the other algorithms.

5. Best classifier: This image is then classified as blurry or not blurry with the best classifier model, which is CatBoost.

The next diagram explains how the system extracts images from a document, how to process them, and classify them whether they are blurry or not.

2.5.2.1 Checking images position

In this section, we begin by considering Images position
After describing our methodology, we apply it to a system we designed to locate images, showing how well it captures it and explaining the performance of the detection algorithm. We next turn to the problem of images position recognition and evaluation.
The diagram shows the steps taken to discover the location of images, then evaluate its positions whether it's in the center.

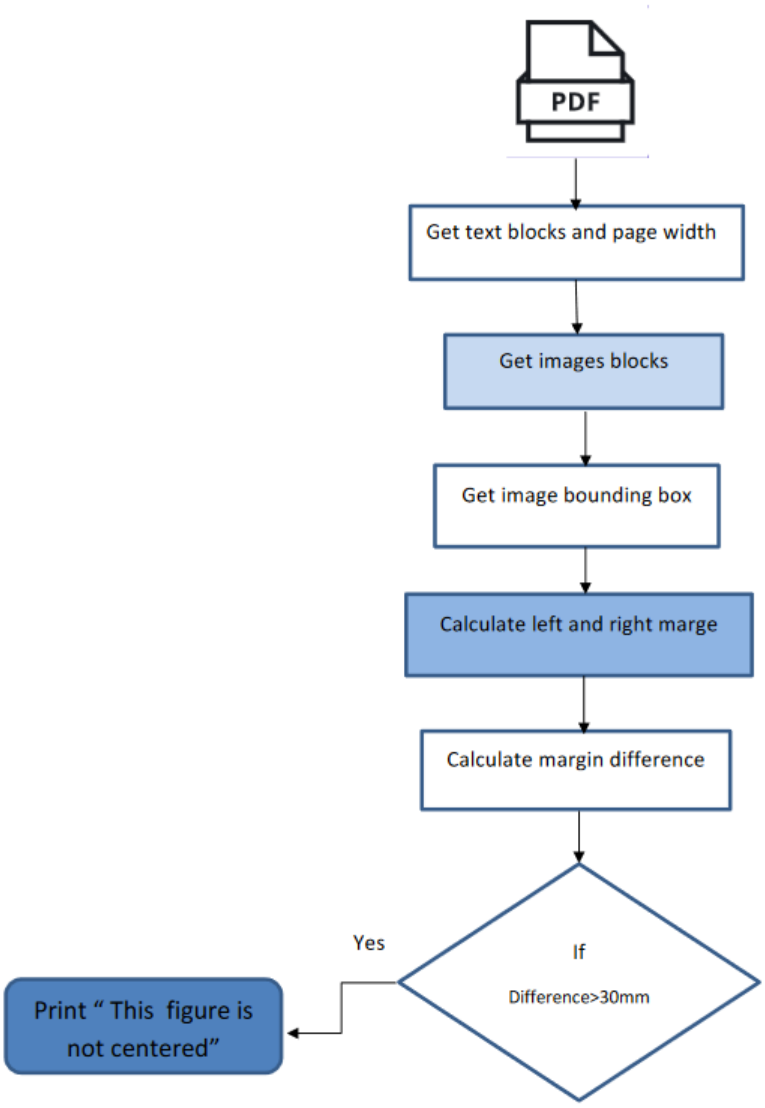


Figure 2.6: Checking image position diagram

- The content of the PDF document is loaded.
- Extract text blocks from the current page: Extracts text blocks, these blocks will likely contain images or text.
- Get Page Width: Calculates the width of the current page. This value will be used later to calculate the image margins relative to the page width.
- Check if the block contains an image: the code checks if the current block of text contains an image.
- Get image bounding box: If the block contains an image, the code will retrieve the image's bounding box coordinates within the page. This box determines the location and size of the image on the page.
- Calculate left and right margins: Calls the specified function get-image-margins to

calculate the left and right margins of the image. It uses the previously retrieved image bounding box coordinates and page width to perform this calculation.

- Margin Difference Calculation: For each entry in the results list, we calculate the difference between the left and right margins, effectively determining how centered the image is on the page.
- Check if the difference is greater than 30: The code compares the calculated margin difference with a threshold of 30mm.
- Print "Page is not centered" message (Yes): If the margin difference is greater than 30 mm, the code determines that the image is not centered and prints a message indicating this for the current page being analyzed.
- If the margin difference is less than or equal to 30 mm, we move to the next entry in the results list without printing anything specific about the centering of that image.

2.5.2.2 Ckecking images captions

In this section, we begin by considering images captions discovery problem
We will describe our methodology on the problem of detecting the presence of image captions.
Our diagram shows the steps taken to discover the location of images then, it shows whether it contains a caption:

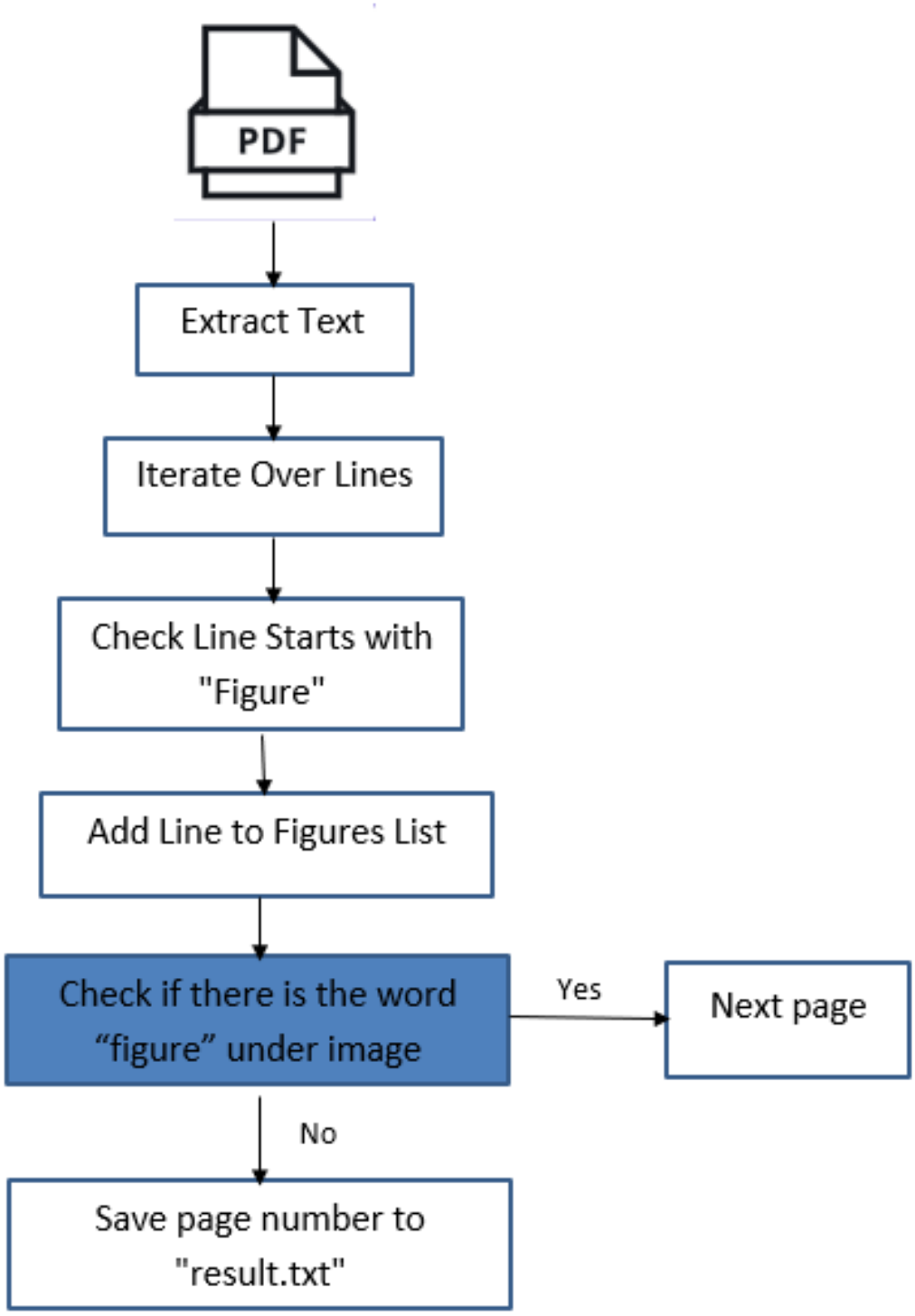


Figure 2.7: Checking image caption diagram

- Extract Text: The process begins by extracting text content from a PDF file. This corresponds to the part in the Python code where the fitz library is used to open the PDF and then extract text.
- Iterate Over Lines: The extracted text is then iterated over line by line. This matches the Python code where the text.split("") method splits the text into a list of lines,

which are then looped over using a for loop.

- Check Line Starts with "Figure": Inside the loop, each line is checked to see if it starts with the word "Figure". This aligns with the Python code's conditional statement if line.startswith("Figure"). o Yes: If the line starts with "Figure", the loop continues to the next line.

  o No: If the line doesn't start with "Figure", ", it's added to a list (figures list in the code). This matches the Python code where lines starting with "Figure" are appended to the figures list using figures.append(line).the loop continues to the next line.

- Save Page Number : where the page number is potentially saved if there's no "Figure" under image The process ends after iterating over all the lines in the extracted text.

2.5.3 References

Using references is crucial in writing articles or composing theses as it signifies credibility. Writing references also adheres to a specific structure, and if the writer deviates from it, it is considered an organizational error related to the logical structure of the document. Among the incorrect and common cases of writing references, we can mention the following:
- Writing the reference after period or colon (. or :) at the beginning of the text or paragraph,etc...
Meanwhile, among the cases considered correct, we have the following:
- If the reference is before period or colon (. or :).
Our diagram shows the steps taken to discover the location of references (citations) and then evaluate its position whether it is in correct place or not.

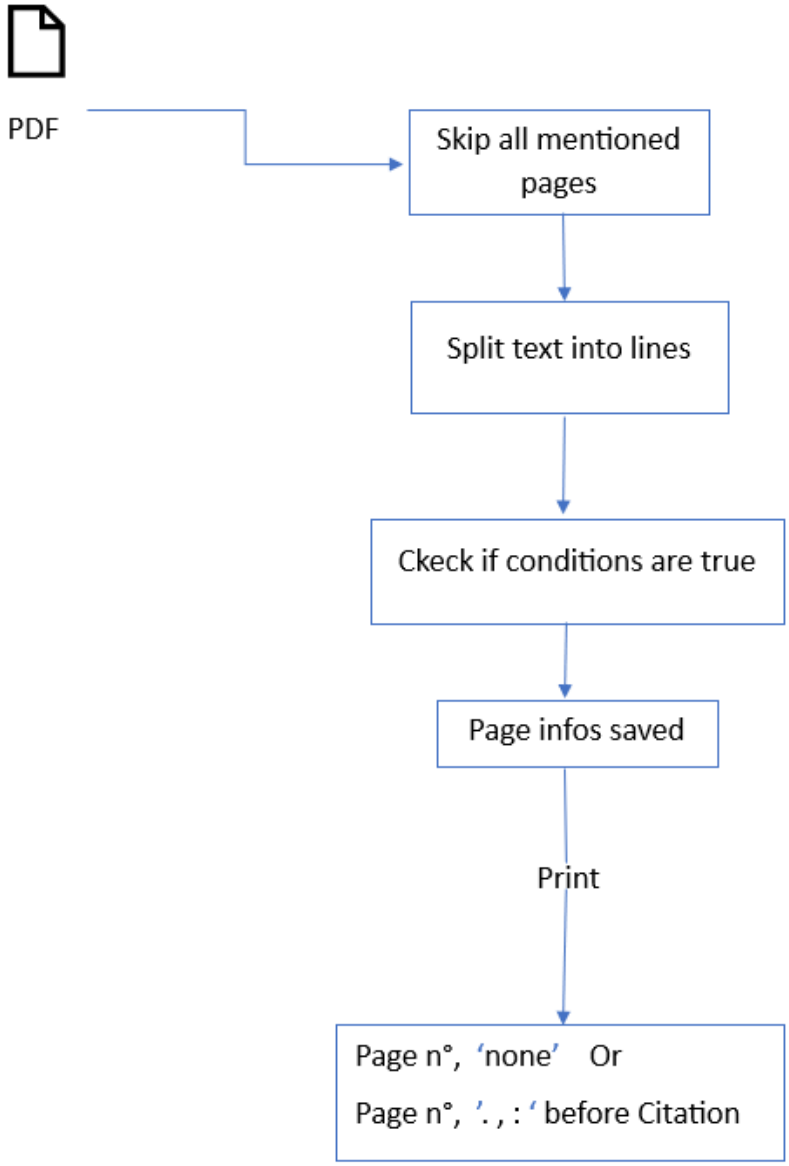


Figure 2.8: Flowchart of checking references

- Process each page: It iterates through all pages in the PDF then extracts text from the current page.
- Checks if the extracted text starts with any keyword from skipped list, if it does, it sets a flag skip-pages to True and moves to the next page, skipping pages containing the table of contents table of figures ...etc.
- If the page isn't skipped: will split the page text into lines.
- Loops through each line: Uses the regular expression pattern to find all citations in the line.
- For each matched citation: Extracts the matched text and checks the preceding

character for punctuation.

Printing the output file containing the page number, preceding punctuation (if any), and the extracted citation.

Conclusion

In this chapter, we first began by defining the types of errors and listing a comprehensive set of errors commonly found in students' thesis. Then, we narrowed down the list of errors that the system will address, designing the general structure of the proposed system. Finally, we provided detailed explanations of each part of the system and how it operates.

In the next chapter, we will focus on building the system itself. We will explain the tools, programming languages, and environment used to construct our system, as well as its implementation methodology.

Chapter 3

IMPLEMENTATIONS & RESULT

Introduction

This chapter will focus on the implementation of the system itself. Firstly, we will present tools and programming language used to construct our system. Following that, we will delve into the key components that make up our system

3.1 Development Tools

In this section, we present the different software tools that we used to carry out our models and experiments.

3.1.1 Material

We implemented our approach and carried out our experiments on a HP computer, under the operating system: Windows 10 Pro 64 bits, Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz and installed memory (RAM) : 8.00 GB.

3.1.2 Pycharm

PyCharm is a hybrid platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE! [21].

Figure 3.1: Pycharm Logo [1]

3.1.3 Python 3.10

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. Python interpreter and extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed [5].

Figure 3.2: Python Logo [8].

3.2 the used packages and algorithms

3.2.1 Mahotas

Mahotas is a computer vision library for Python. It contains traditional image processing functionality such as filtering and morphological operations as well as more modern computer vision functions for feature computation, including interest point detection and local descriptors [12].

3.2.2 Zernike

We use Zernike Moments to characterize and quantify the shape of an object. However, Zernike Moments are more powerful and generally more accurate image descriptors with very little additional computational cost [14].
Zernike Moments are not available in OpenCV. To utilize Zernike Moments we'll be using the **mahotas** package [14].

3.2.3 Joblib

Joblib is especially useful for machine learning models because it allows to save the state of computation and resume work later or on a different machine [22].
We use it for saving and loading model

3.2.4 Scikit-learn

Scikit-learn, also known as sklearn, is an open-source, machine learning and data modeling library for Python. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python libraries, NumPy. It implements numerous data modeling and machine learning algorithms, and provides consistent Python APIs [4].

Figure 3.3: Scikit-learn Logo[9].

3.2.5 CatBoost

3.2.5.1 Definition

CatBoost is a machine learning algorithm that is designed to work particularly well with data that contains categorical features (e.g. colors, genres, product categories). It belongs to the family of boosting algorithms, which are ensemble methods that combine many simple models (like decision trees) into one powerful predictive model. [7].

3.2.5.2 Why CatBoost ?

CatBoost is a powerful tool for gradient boosting tasks, especially when dealing with categorical data and when robustness to overfitting is a concern here's why we choose it :

1. Categorical Features Handling: CatBoost is particularly efficient at handling categorical features without requiring preprocessing or one-hot encoding. It internally deals with categorical variables, which can save time and effort in the feature engineering process [7].

2. Predictive Performance: CatBoost often delivers better predictive accuracy compared to other gradient boosting algorithms, making it a powerful choice for various machine learning tasks [7]. CatBoost is known for its high performance and scalability. It's optimized for speed and memory usage, making it suitable for large datasets and complex models [7].

3. Robust to Overfitting: CatBoost incorporates techniques like ordered boosting and effectively controls overfitting by managing the depth of trees and the learning rate, improving the model's generalization [7].

3.2.5.3 Implementation Binary classification using CatBoost

To get started with CatBoost, we typically need to define a problem classification, prepare the data, create a CatBoost model, train the model, and use it to make predictions [7].

Problem defining

The code provided is a binary classification problem. Specifically, it's aimed at predicting whether an image is clear (1) or the image is blurry (0) based on various features provided in the dataset. The problem can be summarised as follows:

Problem Type : Binary Classification

Target Variable

0: Blurry image

1: Clear image

3.2.6 Logistic regression

This is a classification algorithm used when a response variable is categorical. The idea of logistic regression is to find a relationship between properties and the probability of a given outcome. For example, when we have to predict whether a student passed or failed a test when the number of hours he spent studying is given as a characteristic, the response variable has two values, pass and fail [11].

3.2.7 Support Vector Machine (SVM)

Support Vector Machine (SVM) is also a binary classification algorithm. Just like logistic regression. If we take the image above, we have two classes (Let's imagine that these are emails, and that Spam emails are in red and non-spam emails are in blue). Logistics regression will be able to separate these two classes by defining the line in red. the SVM will opt to separate the two classes by the green line [26].

Without going into details, and for mathematical considerations, the SVM will choose the clearest possible separation between the two classes (like the green line). This is why it is also called Large Margins classifier [26].

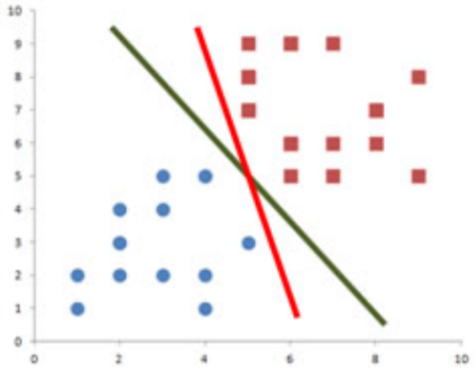
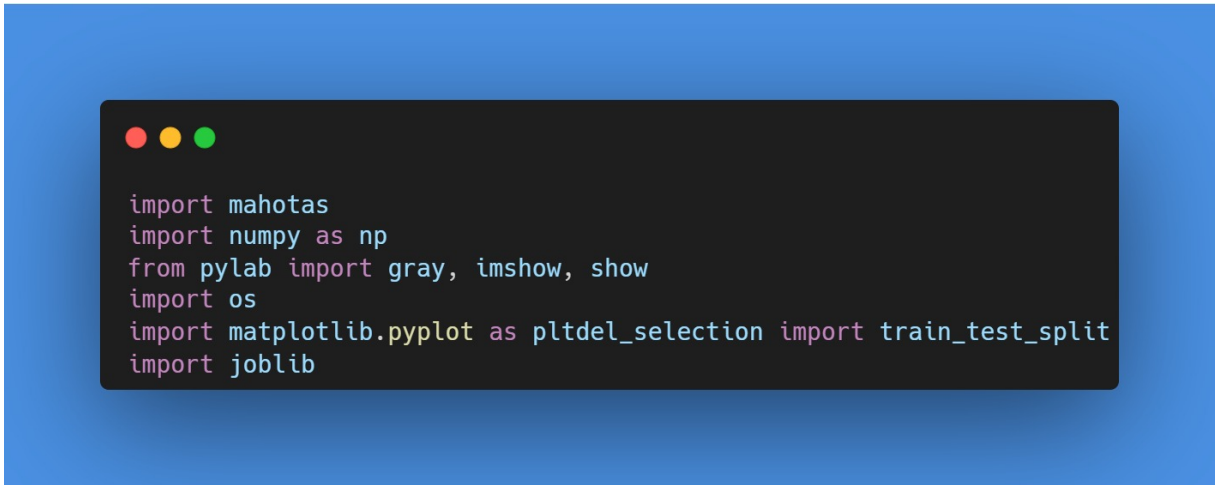


Figure 3.4: Support Vector Machine (SVM) [26].

3.3 Code explanation

This section of chapter provides a detailed description of our "Document errors verification system" code, supported with various interfaces.

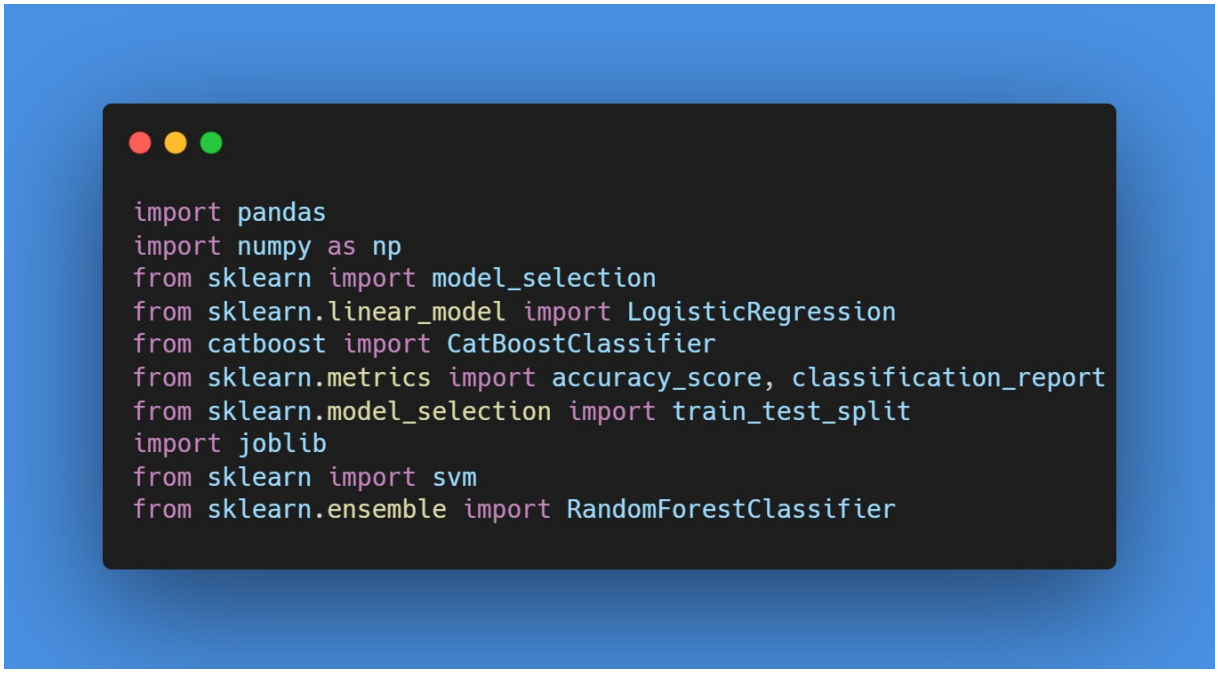
- Those are the packages used in code of extracting images features:



```
import mahotas
import numpy as np
from pylab import gray, imshow, show
import os
import matplotlib.pyplot as pltdel_selection import train_test_split
import joblib
```

Figure 3.5: Extracting images features packages.

- Here we have the list of packages that we used while training model features:



```
import pandas
import numpy as np
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from catboost import CatBoostClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
import joblib
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
```

Figure 3.6: Training model packages .

- These imports deal with processing text extracted from a PDF document.

  To reach to these libraries, we have tried some libraries before Such as **Spire.pdf** and **PySimpleGUI**.

  Which gave very good results in identifying and extracting text components

  But it was not free to use, its use was very limited.

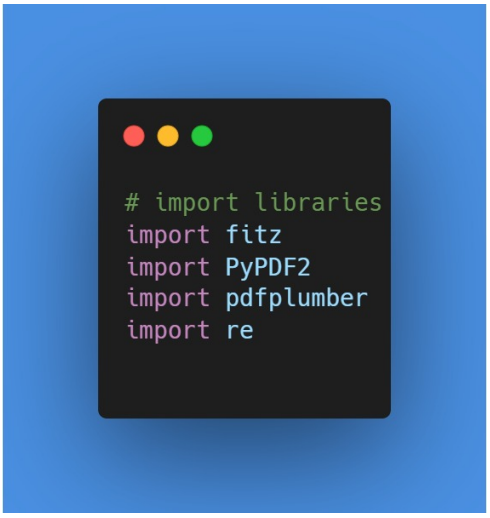


Figure 3.7: Processing text Libraries

- PyPDF2: used to extract text content from each page of a PDF.
- re: This library allows us to define regular expressions to search for specific patterns within the extracted text. This could be useful for tasks like finding keywords, references, or specific data points.
- pdfplumber: is a Python library used for extracting text, tables, and other content from PDF files.
- fitz: A module from PyMuPDF used for working with PDF files.
- Zernike moments for images:

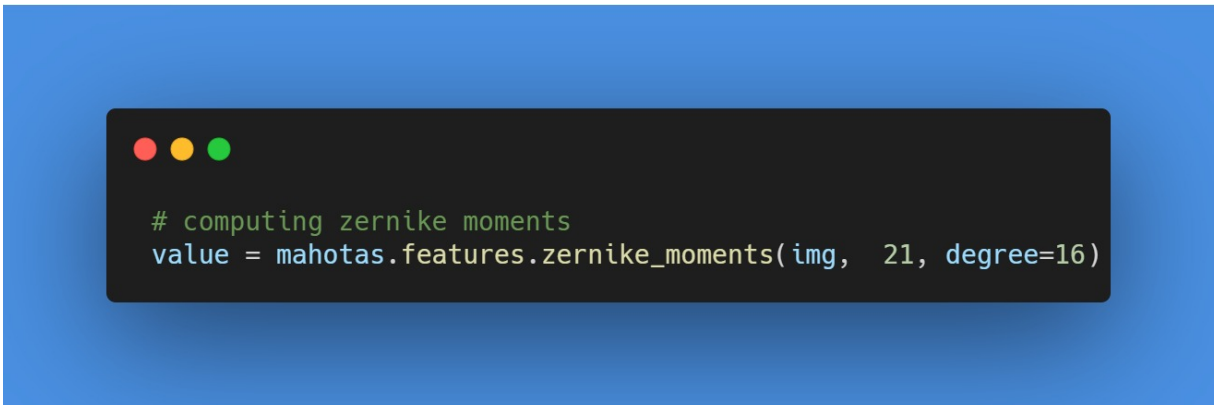


Figure 3.8: Zernike Moments .

- img: The grayscale image.
- "21" The radius used for computing the moments.
- "degree=16" The degree of Zernike polynomials used.
- Train and test Sptiting code:

Figure 3.9: Train Test spliting.

- X contains the features (Zernike moments).
- Y contains the labels (blurred or not blurred).
- model-selection.train-test-split: splits the data into training and testing sets with 20 reserved for testing.

- The next table shows training result of the four used algorithms (SVM,LR,RF and Catboost) as: Accuracy,Recall and Precision

Model	Accuracy	Recall	Precision
SVM	0.91	0.77	0.98
Logistic Regression	0.90	0.77	0.98
Random Forest	0.90	0.78	0.95
CatBoost	0.93	0.79	0.95

Table 3.1: Accuracy of the used algorithms

```python
# Function to extract bold text that starts from the beginning of the line
def extract_bold_text_from_pdf(pdf_path, output_file_path, start_page=0):
    with pdfplumber.open(pdf_path) as pdf:
        with open(output_file_path, "w", encoding="utf-8") as output_file:
            for page_number in range(start_page, len(pdf.pages)):
                page = pdf.pages[page_number]

                text_lines = page.extract_text().split('\n')
                bold_text = page.filter(lambda obj: obj["object_type"] == "char" and "Bold" in obj["fontname"])

                extracted_text = bold_text.extract_text()
                if extracted_text:

                    valid_bold_text = []
                    for line in extracted_text.split('\n'):
                        if any(line.startswith(text_line) for text_line in text_lines):
                            valid_bold_text.append(line)

                    if valid_bold_text:
                        output_file.write(f"Page {page_number + 1}\n" + "\n".join(valid_bold_text) + "\n")
                        output_file.write("-" * 40 + "\n")
                    else:
                        output_file.write(f"No valid bold text found on page {page_number + 1}.\n")
                        output_file.write("-" * 40 + "\n")
                else:
                    output_file.write(f"No bold text found on page {page_number + 1}.\n")
                    output_file.write("-" * 40 + "\n")
```

Figure 3.10: Extracting Titles .

- Extracting titles based on their bold structure
- using this function "def extract-bold-text-from-pdf" from library "pdfplumber"
- After extracting titles, this code checks if they have same formattig:

```python
def check_colon(input_filename, output_filename):
    with open(input_filename, 'r', encoding="utf-8") as file:
        lines = file.readlines()

    ends_with_colon = [line.strip().endswith(':') for line in lines]

    if all(ends_with_colon) or not any(ends_with_colon):
        consistency_check = "All lines are consistent (all end with colon or none end with colon)."
    else:
        inconsistent_lines = [line.strip() for line, ends in zip(lines, ends_with_colon) if ends !=
ends_with_colon[0]]
        consistency_check = "Error: Inconsistent ending detected. The following titles are
inconsistent:\n" + "\n".join(inconsistent_lines)

    with open(output_filename, 'a', encoding="utf-8") as output_file:
        output_file.write("Titles checker:\n\n")
        output_file.write(consistency_check + "\n")
```

Figure 3.11: Colon in titles .

- Function Definition: get-image-margins

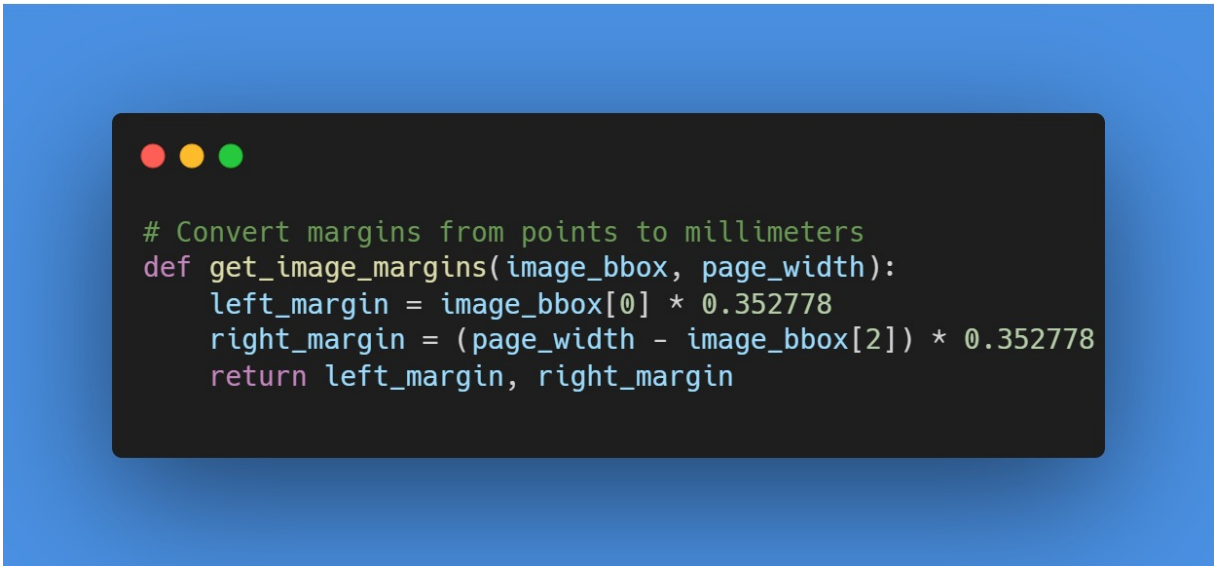


Figure 3.12: Image position .

- Input Parameters: image-bbox: A bounding box of the image, typically a tuple (x0, y0, x1, y1) representing the coordinates of the image in points.

  page-width: The width of the page in points.

  Conversion Factor: The conversion factor from points to millimeters is approximately 0.352778 (since 1 point = 1/72 inches and 1 inch = 25.4 mm).

- Margins Calculation:

  left-margin: The distance from the left edge of the page to the left edge of the image (x0) converted to millimeters.

  right-margin: The distance from the right edge of the image (x1) to the right edge of the page converted to millimeters.

- This function specifies a list of keywords in "ContentPage" variable (list of figures, list of tables, etc...)
- In "pattern" variable, we set the wrong format for references that should be detected

```
# Define keywords that indicate pages to skip

    contentpage = ['Table des matières', 'content', 'Liste des figures', 'Liste des
tableaux', 'Liste des algorithmes']
    pattern = r"([.:]?\s*\[\d+\])"
# Process the page if not skipped
if check_start_keywords(text, contentpage):
                skip_pages = True
                continue
            lines = text.splitlines()
            for line in lines:
                matches = re.finditer(pattern, line)
                for match in matches:
                    matched_text = match.group(0)
                    preceding_punctuation = matched_text[0] if matched_text[0] in
'.,:' else 'None'

                    output.write(f"Page {page_number + 1}: Preceding
'{preceding_punctuation}', Citation: {matched_text.strip()}\n")
```

Figure 3.13: Verifying citations in pages not skipped .

- This function checks whether pages start with one of the pre-defined keywords to skip them

```
    # Check if the beginning of the text matches any of the
specified keywords
def check_start_keywords(text, keywords):
    for keyword in keywords:
        if text.strip().startswith(keyword):
            return True
    return False

def check_end_keywords(text, end_keywords):
    for keyword in end_keywords:
        if keyword in text:
            return True
    return False
```

Figure 3.14: Skipped keywords .

3.4 Result

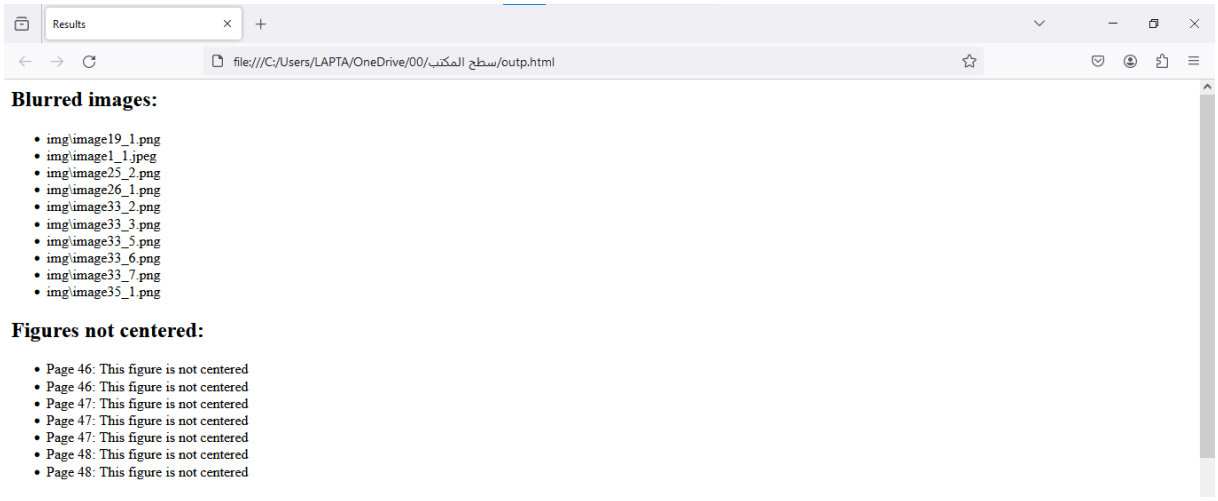


Figure 3.15: Errors list.

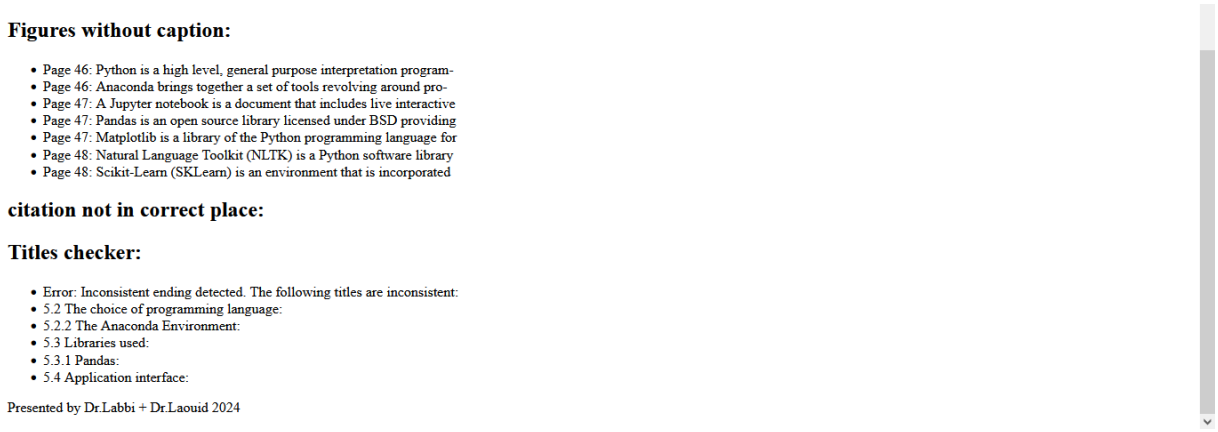


Figure 3.16: Errors list .

3.5 Discussion

A Document Errors Verification System is a tool designed to identify, classify, and help correct errors in digital documents. These systems analyze documents for structural, formatting, content and logical errors using AI and other techniques. They leverage a document's physical and logical structures to detect issues such as missing sections or inconsistent formatting. Key components include error detection algorithms, classification models, and user interfaces for reporting and correction. These systems are crucial in maintaining document quality across various domains. Challenges include handling complex formats and balancing automation with human judgment. The section should cover types of errors, methodologies, challenges, and future directions, supported by research and case studies.

We can notice that the proposed methods succeeded to help achieving the main goals,this

system can detect the errors with efficiently, and training the model on them was very helpful.

Conclusion

Finally, this chapter delves into the design and implementation of our Document errors verification system. We investigated the numerous development tools and technologies used throughout the system's development, including the programming languages, frameworks, and libraries that were critical to its success. In addition, We provided a detailed explanation of all parts of the code.
The system we built to detect formatting and organization errors in documents is considered a solution that helps in detecting errors in thesis, serving both the teacher and the student.

Conclusion and perspectives

Students continue to make many mistakes while editing their theses.
The goal of this thesis is to create a Python application that uses a data source (dataset, pdf, csv) to extract a set of formatting errors that are likely to be frequently found in students' theses.
We started by defining some concepts and then talked about five similar works. After that, we studied the effectiveness of supervised training algorithms "SVM, LR, CatBoost, and RF." We created a prediction model to classify types of images whether they are clear or not with the help of the Zernike feature extraction function, where SVM gave very good results, but CatBoost saved Better results, such as better accuracy (93 percent) were obtained among all the algorithms tested.
We explained in detail all the other parts of this program, including checking the format of titles and checking the placement of references and captions.

System limitations

A "Document errors verification" system typically aims to automate the process of identifying errors in documents, which is immensely beneficial in various domains . However, like any system, it comes with its own set of limitations:

1. False Positives: Automated systems might flag correct usage as errors, leading to false positives. For instance, Blurry images may be incorrectly identified as errors. whereas , the user may intentionally input it in that form or quality.

   Also,

   Addressing these limitations requires a combination of advanced technologies, human oversight, and continuous refinement of the system's algorithms and processes.

2. Unlike images, the system can't detect the position of shapes because they are not defined within a box.

3. About references, this system considers some reference methods are incorrect, while they are correct this is due to the multiplicity of reference methodologies

Future Work

Perspectives For future work, we can cite:
- Collect the largest possible number of frequent and possible formatting errors.
- Covering possible errors outside the chapters (cover page, table of images, list of tables, etc.)
- Applying it to different languages, such as French and Arabic, to include all languages.
- Adding the feature of processing texts and titles from a linguistic standpoint so that the system can fully detect all aspects of the memorandum.
- Experimenting with more and better classification and detection algorithms, if any, to obtain very accurate results.

Bibliography

[1] "Pycharm logo png vector" – https://seeklogo.com/free-vector-logos/pycharm, [Online; accessed 10-May-2024].

[2] "What is artificial intelligence?" – https://www.britannica.com/question/What-is-artificial-intelligence, [Online; accessed 01-Mars-2024].

[3] "What is deep learning?- deep learning explained - aws" – https://aws.amazon.com/what-is/deep-learning, [Online; accessed 01-Mars-2024].

[4] "What is sklearn?" – https://domino.ai/data-science-dictionary/sklearn.

[5] "What is python?" – https://www.python.org/doc/essays/blurb/, 2019, [Online; accessed 10-May-2024].

[6] "Meanings definitions of english words" – https://www.dictionary.com/browse/document, September 2020, [Online; accessed 01-Mars-2024].

[7] "Binary classification using catboost" – https://www.geeksforgeeks.org/binary-classification-using-catboost, November 2023.

[8] "Logos" – https://logos-world.net/python-logo/, April 2024, [Online; accessed 10-May-2024].

[9] "Scikit-learn" – https://en.wikipedia.org/wiki/Scikit-learn, April 2024.

[10] E. AI. – "What is machine learning and how does it work?", https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML, [Online; accessed 01-Mars-2024].

[11] Y. M. M. Benzaki – "9 algorithmes de machine learning que chaque data scientist doit connaitre.", https://mrmint.fr/9-algorithmes-de-machine-learning-que-chaque-data-scientist-doit-connaitre, Augest 2017, [Online; accessed 12-Mars-2024].

[12] L. P. Coelho – "Mahotas: Open source software for scriptable computer vision".

[13] A. R. David Doermann, Ehud Rivlin – "The function of document", (1996).

[14] R. Dayala – "10.5 zernike moments. computer vision.", https://cvexplained.wordpress.com/2020/07/21/10-5-zernike-moments/, July 2020, [Online; accessed 12-Mars-2024].

[15] I. F. Hanane BOUFERSAOUI – "Extraction de la structure logique des documents", Faculté des mathématique, d'informatique et de science de matière, June 2015.

[16] D. L. G. W. J. Hu, R.S. Kashi – "Evaluating the performance of table processing algorithms", 2002, p. 40–153.

[17] S. E.-R. G. M. J. H. Lamiaa Abdel-Hamid, Ahmed El-Rafei – "Retinal image quality assessment based on image clarity and content", 2016, p. 1–17.

[18] P. S.-D. Laurie Buscail – "Textual and stylistic error detection and correction: Categorization, annotation and correction strategies", (2009), p. 205–210.

[19] A. S. Maria Zhigalova – "System of automated checking of textual document design rules", *Information Content and Processing* **2** (2015), no. 4, p. 164–180.

[20] T. L. Packer – "Performing information extraction to improve ocr error detection in semi-structured historical documents", Thèse, Department of Computer Science Brigham Young University Provo, Utah, USA, 2013.

[21] A. Pushkar – "What is pycharm? intellipaat", https://intellipaat.com/blog/what-is-pycharm/, December.

[22] P. Sharma – "How to save and load machine learning models in python using joblib library?", https://www.analyticsvidhya.com/blog/2023/02/how-to-save-and-load-machine\protect\@normalcr\relaxlearning-models-in-python-using-joblib-library/, February 2023, [Online; accessed 12-Mars-2024].

[23] A. SOUQUE – "Modèle de vérification grammaticale automatique gauche-droite", Thèse, École Doctorale no 50 – Langues, Littérature et Sciences Humaines, 2014.

[24] A. Srinivasacharyulu – "Documentation", https://www.slideshare.net/ascharyulu/documentation-presentation-626328, September 2008, [Online; accessed 01-Mars-2024].

[25] I. E. Team – "21 different types of files and how to use them", https://www.indeed.com/career-advice/career-development/types-of-files, June 2024, [Online; accessed 01-Mars-2024].

[26] K. Teknomo – "Introduction to svm tutorial.", , note =.

[27] D. S. S. Valerii S. Hlukhov – "Algorithms and software for verification of scientific and technical text documents", *Applied Aspects of Information Technology*, ORCID, 2023, p. 304–317.