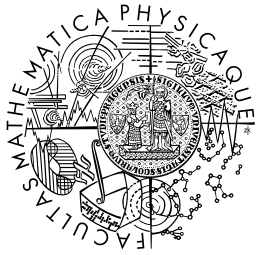


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Nikita Saydametov

Segmentation of scanned PDF documents

Department of Software and Computer Science Education

Supervisor of the bachelor thesis: doc. RNDr. Elena Šikudová, Ph.D.
Study programme: Computer Science
Study branch: IOI

Prague 2024

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication. I would like to express my sincere gratitude to doc. RNDr. Elena Šikudová, Ph.D., for her constant and highly appreciated help along with her endless enthusiasm. Also, I would like to thank my family for their support throughout the work on this thesis.

Title: Segmentation of scanned PDF documents

Author: Nikita Saydametov

Department: Department of Software and Computer Science Education

Supervisor: doc. RNDr. Elena Šikudová, Ph.D., Department of Software and Computer Science Education

Abstract: Abstract.

Keywords: PDF OCR Tesseract segmentation

Contents

Introduction

Every new day that we live becomes part of history. For many artifacts of history are the remains of dinosaurs and mammoths that lived on earth more than 200 million years ago.

One of these artifacts we can consider and old preserved documents, clippings of newspapers that were published over 300 years ago, which unfortunately in normal conditions can not keep their original form due to the fact that the ink fades over time , and the paper deteriorates. But the good news is that humanity is not standing still. And people were able to preserve these ancient materials through digitization. But also due to the development of the Internet a huge number of databases are created, most of the data you need can be easily found in large online libraries or through search engines.

But our digitized old newspapers clippings, books and booklets still remain either as photos or as unindexable pdf documents. And because of that a huge amount of information stored there remains outside the indexed Internet.

My goal is to change this problem by creating a program which segments a pdf document into the text and pictures it contains. You might have a reasonable question: "But how can a program recognize text, when there are so many different diacritical marks and nuances of spelling in the world? And you would be right, because it's a big enough problem that an ordinary program would be unable to solve it, but we live in the 21st century! Technology is still going strong, so at the heart of the program is the principle of machine learning, which allows it to work with more than 100 languages.

1. Background and theory

To fully grasp the significance of this work, one must engage with the theoretical data comprehensively to ensure an understanding of all critical details.

1.1 PDF

In today's digital environment, Portable Document Format (PDF) stands out as a prevalent file type, primarily due to its distinctive features that ensure consistent visual representation of documents across different platforms. This section delves deeper into the essence, history, varieties, and the range of software tools designed for creating and editing PDFs. A thorough grasp of these elements not only enriches our fundamental knowledge of PDF as a file format but also enhances our understanding of the subsequent sections of this document. Here, we will explore the specialized software that we have engineered for extracting text from PDF files. This exploration will include a discussion on how PDFs maintain layout and formatting consistency, making them indispensable in a multi-device, platform-diverse digital landscape. Moreover, we will look at the evolution of PDF standards and how they have adapted to the demands of digital documentation, further cementing their role in document preservation.

1.1.1 About PDF

In the early 1990s, Adobe Systems Inc. replaced its PostScript language with the PDF file format. Adding capabilities like interactive navigation and organized document layering, the format expands upon this framework. Its primary benefit is that documents display uniformly on all devices due to its independence from both hardware and software.

As the name implies, PDFs fall within the Page Description Language (PDL) group, which is concerned with a document's graphical representation. It places more emphasis on the visual presentation of a page than its informative content. In contrast to Data Description Languages (DDLs) like HTML and RTF, markup languages prioritize the semantic value and meaning of the data they include.

The visual arrangement of text and other components, such photos, is crucial when it comes to PDFs. The significance or informative value of these components, on the other hand, is incidental. For this reason, obtaining text from a PDF document might be difficult. Some PDFs may even be created in such a way that it is difficult to programmatically recognize the text using any other technique than optical character recognition (OCR).

Because PDF files are so widely used in both the public and private sectors, Adobe Systems Inc. decided on January 29, 2007, to offer a full specification in PDF format for standardization and publishing in ISO (International Organization for Standardization) format. January 2008, a year later, saw the paperwork recognized as an ISO 32000-1 worldwide standard. As a result, PDF has become a standard format. You may obtain the specification for free straight from the Adobe website.

1.1.2 PDF document types

In the specification - globally, we can distinguish two types of PDF documents:

- Searchable PDF
- Non-searchable PDF

Let us go over them in detail.

The main task of our program is to extract and convert text. You can extract text manually as follows:

- Open your document in any PDF editor
- Select and copy the text

This method works efficiently for most documents, which are often referred to as "searchable PDFs". In such documents, the text is generated using different PDF operators. Moreover, the linked font objects within these documents possess accurate information regarding the correlation between the glyphs and Unicode values, making them accessible for extraction.

Numerous PDF libraries have been developed with the capability to extract text from these searchable PDFs. However, challenges arise with documents that fall outside the category of searchable PDFs. These documents, called "non-searchable PDFs", typically present their text as bitmap images. A perfect example of this would be a scanned PDF document. In these scenarios, text can be represented in vector paths, bypassing specific PDF operators and fonts.

Optical text recognition is performed to extract text from inaccessible PDFs. Optical OCR does not guarantee proper text extraction 100% of the time. The result depends on the quality of your document and the recognition algorithm. Also, OCR is significantly slower than text extraction from searchable PDF files.

1.1.3 PDF software

The majority of PDF viewing programs are free, and there are several versions accessible from different sources.

Software options for creating PDF files are numerous and include the pdfTeX typesetting system, applications developed from Ghostscript and Adobe Acrobat itself, Adobe InDesign, Adobe FrameMaker, Adobe Illustrator, and Adobe Photoshop; software options for creating PDF files include Scribus; LibreOffice; Microsoft Office 2007 and up; WordPerfect 9; and many more. The online office suite from Google, Google Docs, allows you to download and store documents in PDF format. A few websites provide free tools for editing annotations and PDF files [1].

1.1.4 Security PDF

Numerous incidents throughout PDF's history have seen attackers taking advantage of the format's widespread use.

The attachment contained within the document was the most frequent vulnerability in these files. Because the virus was contained in the file and only became

active when the document was opened, inexperienced users need assistance in identifying it.

The first reports of virus-containing PDF attachments date back to 2001. The virus known as OUTLOOK.PDFWorm, also known as Peachy, sent itself as an Adobe PDF attachment via Microsoft Outlook [2]. This virus was unique in that it required Adobe Acrobat to initiate its activation, even if Acrobat Reader was not able to activate it.

Occasionally, fresh vulnerabilities in different versions of Adobe Reader are found, leading the business to provide security updates [3]. Other PDF viewers are vulnerable as well. An further aggravating element is that if a web page has an embedded PDF file, a PDF reader can be configured to launch automatically, opening up a potential attack channel. Even with a secured browser, the system can still be penetrated if a malicious web page has an infected PDF file that takes advantage of a flaw in the PDF reader. The PDF standard, which permits JavaScript to be used in the creation of PDF documents, is partially to blame for these vulnerabilities. While disabling JavaScript execution in the PDF reader does not shield against attacks in other areas of the PDF viewer software, it can aid in preventing similar future exploits. JavaScript is not necessary to read PDFs, according to security experts, and removing JavaScript has more security advantages than compatibility problems. Using a local or web service to convert files to a different format before viewing them is one method to prevent PDF vulnerabilities.

Researchers from Hackmanit GmbH and Ruhr University Bochum released information on attacks on PDFs that include digital signatures in November 2019. By taking advantage of implementation defects, they demonstrated how to alter visible information in a signed PDF file without nullifying the signature in 21 out of 22 desktop PDF viewers and 6 out of 8 online validation services. They also demonstrated how to filter out encrypted plaintext material from PDF files at the same conference. They presented further shadow attacks on PDFs in 2021, taking use of the features' inherent flexibility. An summary of security vulnerabilities in PDF files pertaining to arbitrary code execution attacks, data manipulation, information exposure, and denial of service was provided by Jens Mueller [4].

1.1.5 Licensing

Applications that can write to and read PDF files can be made by anybody without having to pay Adobe Systems royalties.

Although Adobe owns the PDF patents, it grants free licenses to developers of software that complies with the PDF standard.

1.1.6 Environmental Impact of Digital vs. Physical Documents

Introduction

The transition from physical to digital documentation, particularly in the form of PDFs, has significant environmental implications. This subsection analyzes the sustainability impact of this shift.

Reduction in Paper Usage The most apparent environmental benefit of digital documents is the reduction in paper consumption. The manufacturing of paper, primarily from wood pulp, has a considerable ecological footprint, involving deforestation, water consumption, and energy use.

Deforestation and Biodiversity Loss Paper production is a leading cause of deforestation, which leads to biodiversity loss and habitat destruction. By reducing reliance on paper, digital documents contribute to the preservation of forests.

Water and Energy Resources Paper manufacturing is water and energy-intensive. Transitioning to digital formats significantly reduces the consumption of these vital resources, contributing to environmental sustainability.

Carbon Footprint of Digital Documents While digital documents reduce paper use, they have their carbon footprint, primarily due to energy consumption for data storage, processing, and transmission.

Data Centers and Energy Consumption The operation of data centers, crucial for storing digital documents, requires substantial energy, often sourced from fossil fuels, contributing to greenhouse gas emissions.

Efforts in Green Computing Advancements in green computing and the increasing use of renewable energy sources for data centers are mitigating the carbon footprint of digital documents.

Waste Management and Recycling Physical document disposal and recycling processes have environmental impacts, including energy consumption and pollution from recycling processes.

The Role of PDFs in Sustainability PDFs, as a universal digital document format, play a vital role in promoting sustainability by enabling efficient digital documentation and reducing the need for physical paper.

Longevity and Accessibility The durability and compatibility of PDFs across platforms ensure document longevity, reducing the need for multiple copies and reprints.

Interactive Features and Paperless Operations PDFs support interactive features like forms and signatures, enabling completely paperless operations in various sectors, further reducing the environmental impact.

Comparative Analysis A comparative analysis of the environmental impact of physical and digital documents, considering factors like production, usage, and end-of-life management.

Future Outlook Looking ahead, the subsection will explore potential developments in digital documentation technology and practices that could further enhance environmental sustainability.

Innovations in Digital Storage Advancements in energy-efficient storage technologies and practices could significantly reduce the carbon footprint of digital documents.

Sustainable Practices in Document Management The adoption of sustainable practices in document management, both in digital and physical formats, is essential for a greener future.

Conclusion The shift to digital documentation, especially in the form of PDFs, offers substantial environmental benefits, particularly in reducing paper use and the associated ecological impacts. However, it is also essential to address the environmental challenges posed by digital documents, especially concerning energy consumption and e-waste.

1.2 OCR

Pages from a printed book or papers with handwritten notes occasionally need to be scanned. OCR, or optical character recognition, is useful in this situation. This useful technology creates editable digital files from handwritten or printed text [5].

Characters that are scanned, printed, or written can be digitally copied using OCR. Data stored on paper, such as invoices, passports, papers, business cards, letters, or printouts, are very commonly imported via this manner.

Text digitization allows for computerized text editing and searching. Furthermore, the technology makes it possible to save papers in a more compact format and view them online.

1.2.1 History of OCR

Technology for text recognition first emerged in the latter part of the 1800s.

Emanuel Goldberg created a machine in 1914 that scans characters and translates them into telegraph code. Edmund Fournier d'Albe created the Optophone concurrently [6], a portable scanner that generated auditory signals in response to certain letters or symbols as it passed over a printed page.

Emanuel Goldberg created a "Statistical Machine" in the late 1920s and early 1930s that used an optical code recognition method to search microfilm archives. 1931 saw him get U.S. 1,838,389 is the invention's patent number. IBM obtained the patent.

Systems in the contemporary meaning first appeared in the 1950s [7], many decades later.

The initial text recognition systems were machine-based and were only discovered to be utilized in specific establishments, such post offices, where they were employed for mail sorting. It was not until the first personal computers

were introduced that software programs with modern text recognition capabilities started to develop. The term "monofont" refers to the fact that the first text recognition technologies in use today could only recognize one font. OCR-A and OCR-B, the extraordinary typeface standards used in the US and Europe respectively, were developed for this reason. except since they were not being utilized, there was no alternative except to keep advancing technology and raising the sophistication of the systems. OCR systems of the following generation already knew how to read additional typefaces (multi-font). However, they were still in need of more precision and assistance in locating a more complete application due to their high purchasing cost. This has only altered with the release of the newest generation of systems, known as omnifont, which do not rely on type 16 fonts and are progressively finding their way into homes, workplaces, and small businesses owing to advancements in information technology.

1.2.2 Using OCR

OCR technology helps millions worldwide convert information from paper to electronic format.

Banking sector

The banking sector use OCR technology to efficiently and accurately process and authenticate checks for purposes such as deposits, loans, and other financial transactions. This verification has led to an enhancement in transaction security and the efficacy of anti-fraud measures. BlueVine, a financial technology company specializing in financing small and medium-sized enterprises, utilized Amazon Textract, a cloud-based optical character recognition (OCR) service, to develop a product that facilitates the process of obtaining loans under the COVID-19 stimulus package's Wage Protection Program (PPP) for small businesses in the US. Amazon's OCR automatically processed and analyzed tens of thousands of PPP forms daily, enabling BlueVine to help thousands of firms secure finance and save more than 400,000 jobs [8].

Health care

The healthcare system utilizes OCR to manage patient data, including information pertaining to hospital admissions, medical examinations, therapies, and insurance reimbursements. OCR streamlines hospital operations, reduces the need for human work, and ensures up-to-date record keeping. As an example, the Nib Group processes a large number of medical service claims on a daily basis and provides health insurance coverage to more than one million individuals in Australia. Customers of the company have the ability to utilize a mobile application to submit a photograph of their medical invoices. The identification system automatically processes these photographs, therefore expediting the company's claim processing.

Logistics

Logistics companies utilize optical character recognition (OCR) as a technique to more efficiently monitor documents including invoices, receipts, and package labels. Foresight Group, for example, uses OCR to automate the processing of SAP invoices. Foresight staff members had to manually enter data into many accounting systems, which took time and raised the possibility of errors. OCR increased business productivity by enabling Foresight software to read characters on a range of media with better accuracy [9].

Electronic archiving

The electronic preservation of information (such as books) is known as electronic archiving. Physical medium obsolescence, right owner insolvency, and out-of-print publications are all resolved by electronic storage. In fact, a lot of the classic books that were published in limited quantities or with basic editions are now very hard to find, and many can only be acquired by hand. A book's publisher is more likely to stop supporting its dissemination the longer it has been out of print than when it was first published, both in print and digital formats. In that instance, the book can only be preserved by libraries and aficionados.

Ideally, a scanned book should be stored in both an unrecognized (non-OCR) and a recognized (OCR) form. With the OCR version, you may search for keywords and copy extracts to the clipboard, and it often takes up less space. The unacknowledged version keeps a few small yet potentially significant information. Transparent recognized text can occasionally be superimposed over scanned pages to merge the two formats. In these documents, text can be highlighted and keywords may be searched, but the file is much more important than in non-OCR documents.

1.2.3 How OCR works

Pre-processing

OCR software often "preprocesses" images to increase the chances of successful recognition. Methods include:

- Skew Correction - If a document is not properly aligned when scanning, it may require a slight rotation either clockwise or counterclockwise to ensure that the lines of text are completely horizontal or vertical.
- Spot removal - involves eliminating both positive and negative spots while also refining the borders to get a smoother appearance.
- Binarization - the process of converting a grayscale or color image to black and white; the result is an image that has two colors, thus the name "binary image." The task of binarization involves separating the backdrop from the text, or any other desirable picture component.
- Removing strings - clears fields and strings that do not contain characters.

- ”Zoning” or layout analysis recognizes paragraphs, columns, captions, and other elements as discrete blocks. This is particularly crucial for multi-column tables and layouts.
- Line and word recognition is used to provide a reference point for words and character shapes, and it may also separate words if needed.
- Script recognition - in multilingual documents, a script can change at the word level. Therefore, script identification is required before invoking the correct text recognition to handle a particular script.
- Character extraction, often known as ”segmentation,” refers to the process of separating several characters that are connected owing to image artifacts. Similarly, it involves connecting individual characters that have been fragmented into many sections due to artifacts [10].
- Normalization of aspect ratio and scale [10].

Segmentation of fixed-pitch typefaces may be easily accomplished by aligning the picture to a uniform grid, focusing on the points where the vertical grid lines intersect the black regions with the least frequency. Proportional typefaces need more intricate techniques due to the potential presence of greater gaps between letters compared to gaps between words, as well as the possibility of vertical lines intersecting several characters.

Text recognition

Two fundamental recognition algorithms are capable of generating a prioritized list of potential characters.

- Matrix matching is the process of comparing a picture to a recorded glyph by examining each pixel individually. It is often referred to as ”pattern matching,” ”pattern recognition,” or ”image correlation.” This is dependent on accurately separating the input glyph from the surrounding picture and ensuring that the stored glyph is in a comparable font and scale. This method is most effective when used to typewritten material and is particularly successful when dealing with unfamiliar typefaces. This approach refers to the direct implementation of early physical photocell-based OCR.
- Feature extraction breaks down glyphs into ”features” such as lines, closed loops, line orientation, and line intersections. The extraction characteristics decrease the number of dimensions in the representation and enhance the computing efficiency of the recognition process. The characteristics are compared to an abstract vector representation of a character, which can be simplified to one or more glyph prototypes. Commonly utilized in ”intelligent” handwriting recognition and contemporary OCR software, general feature identification techniques in computer vision are applicable to this sort of OCR. Nearest-neighbor classifiers, like the k-nearest neighbors method, assess picture attributes by comparing them to stored glyph features and select the most similar match.

Cuneiform and Tesseract employ a two-pass methodology for character recognition. The second phase is referred to as "adaptive recognition" and utilizes the accurately identified letterforms from the first pass to enhance the recognition of the remaining letters in the second pass. This is advantageous for atypical typefaces or low-resolution scans when the font seems altered (such as being fuzzy or faded).

Some examples of contemporary OCR applications are Google Docs OCR, ABBYY FineReader, and Transym. OCRopus and Tesseract, among others, employ neural networks that have been trained to identify complete lines of text rather than individual letters.

The novel technique, referred to as iterative text recognition, autonomously partitions a document into distinct portions based on the arrangement of the page. The process of recognition is carried out separately for each part, utilizing varying criteria for character confidence levels in order to optimize the accuracy of identification at the page level. This method has been granted a patent by the U.S. Patent and Trademark Office.

Post-processing

Limiting the output to a predetermined vocabulary, which consists of a certain set of words that are permitted to appear in the text, can enhance the accuracy of character recognition. For instance, this can encompass all the terms in the English language or a more specialized lexicon specific to a certain field. This approach may encounter difficulties when the text includes words that are not listed in the dictionary, such as proper nouns. Tesseract utilizes its lexicon to manipulate the process of character segmentation in order to enhance precision.

The output stream can be either a simple text stream or a character file. Nevertheless, advanced OCR systems have the capability to preserve the original page format and generate an annotated PDF document that has both the original page image and a searchable text version.

The "near-neighbor analysis" technique utilizes co-occurrence frequencies to rectify mistakes by identifying common word pairings. As an illustration, the term "Washington, D.C." is more often used in English than to "Washington DOC."

Scanning the grammar of a language can also aid in determining if a word is a verb or a noun, hence enhancing accuracy.

The Levenshtein distance technique has been utilized in OCR post-processing to enhance the outcomes of the OCR API [10].

1.2.4 Benefits of OCR

Searchable text

Users can convert existing and new documents into a fully searchable knowledge base. Automatic text-based processing software allows you to improve and expand your knowledge base.

Work efficiency

OCR software can improve efficiency by automatically integrating document management and digital workflows. Here are a few examples of what OCR software can do.

- Scan manually completed forms for automated verification, review, editing, and analysis. This approach reduces manual document processing and data entry time.
- Search for the documents you need with a quick-term search in a database instead of manually going through files in a drawer.
- Converting handwritten notes into editable texts and documents.

Artificial Intelligence Solutions

OCR is frequently a part of further AI systems that businesses might use. OCR, for instance, may be used to identify company logos in social media postings, identify product packaging in advertising photos, and scan and recognize license plates and road signs in self-driving automobiles. These artificial intelligence technologies assist companies in making more cost-effective and customer-focused marketing and operational decisions [9].

1.2.5 Challenges and Limitations of OCR Technology

Introduction

While Optical Character Recognition (OCR) has revolutionized the digitization of texts, it has challenges and limitations. These issues can affect the technology's efficiency, accuracy, and usability in various applications.

Complex Document Layouts

One of the primary challenges for OCR systems is processing documents with complex layouts. This includes handling multi-column texts, tables, graphs, and images mixed with text. The presence of sidebars, footnotes, and headers can further complicate text extraction.

Handwriting Recognition

Handwriting recognition remains a significant hurdle due to the immense variability in individual writing styles. This includes cursive writing, where letters are connected fluidly, making it hard for OCR to differentiate individual characters.

Language and Character Set Limitations

The effectiveness of OCR technology can vary significantly across different languages and scripts. Languages with extensive character sets, like Chinese, or those with complex scripts, like Arabic, pose significant challenges.

Image Quality and Scanning Limitations

The quality of the source material plays a critical role in OCR accuracy. Poorly scanned documents, low-resolution images, or text superimposed on complex backgrounds can significantly reduce accuracy.

Technological and Software Limitations

Despite advancements, OCR software is only sometimes efficient. Some systems need help with speed and performance, especially when processing large volumes of data or particularly complex documents.

Impact on Different Industries

The limitations of OCR technology can have varying impacts across industries. In sectors like healthcare and legal, where accuracy is paramount, these limitations can lead to critical errors.

Conclusion

Addressing these challenges is essential for the advancement of OCR technology. Ongoing research and development are crucial to enhance its accuracy, adaptability, and efficiency. Machine learning and artificial intelligence innovations are promising pathways to overcoming current limitations.

1.2.6 The Future of OCR and Its Role in Digitalization

Optical Character Recognition (OCR) technology, once confined to the realms of typed text and basic fonts, has undergone a remarkable evolution. Today's OCR systems are adept at deciphering complex handwritten notes, cursive scripts, and many languages, including those with intricate characters. This profound growth is mainly attributable to integrating sophisticated artificial intelligence (AI) and machine learning algorithms, which have significantly improved the accuracy and efficiency of OCR tools.

As we look toward the future, the potential for OCR technology seems boundless. With advancements in deep learning and neural networks, future OCR systems are poised to interpret texts and understand the nuances and subtleties of language. This could lead to groundbreaking applications in legal document analysis, where understanding the context and semantics is as crucial as recognizing the words themselves.

The ongoing integration of OCR with other cutting-edge technologies promises to revolutionize how we interact with digital content. One such promising integration is with natural language processing (NLP). By combining OCR's ability to convert images into text with NLP's ability to understand and interpret language, we could see the emergence of highly advanced automated customer service and data entry systems. This synergy could transform vast amounts of unstructured data into valuable, actionable insights.

Another critical area where OCR is set to impact significantly is the digitalization of documents and records. As businesses and institutions move towards paperless operations, the demand for digitizing paper documents increases. OCR

stands at the forefront of this transition, enabling the conversion of physical documents into editable and searchable digital formats. This transformation is crucial for accessibility, efficiency, and environmental sustainability.

Looking further ahead, we can envision the convergence of OCR with blockchain technology, bringing about a new era of secure and verifiable digital documents. This integration could be particularly transformative in sectors like banking, legal, and governmental services, where the authenticity and integrity of documents are paramount.

In summary, the future trajectory of OCR is inextricably linked with the broader path of digital transformation. Its evolving capabilities, powered by AI and machine learning, and its integration with other technological advancements are set to expand its applications and utility. OCR is transitioning from a tool of convenience to a critical component in the digital ecosystem, playing a pivotal role in accessing, processing, and utilizing information in the digital age.

1.3 Machine learning

A computer may learn without explicit instructions by using mathematical data models, which are the basis of machine learning (ML). In terms of artificial intelligence (AI), it is regarded as such. Algorithms are used in machine learning to find patterns in data. A data model is developed for prediction [11] based on these patterns. The accuracy of the findings increases with the amount of data processed by the model and its duration of usage. This is quite similar to how humans practice honing their talents.

Because machine learning is adaptable, it works well in situations where the data is dynamic, the characteristics of queries or issues are unstable, or the problem is nearly impossible to handle using code [12].

1.3.1 History of Machine Learning Development

Arthur Samuel, an IBM employee who pioneered computer games and artificial intelligence, developed machine learning in 1959. During this time, computers were also linked with self-learning.

By the early 1960s, Raytheon had created the Cybertron, an experimental perforated-memory ”learning machine” that uses basic reinforcement learning to evaluate speech patterns, ECGs, and sonar data. It was repeatedly ”trained” to identify patterns by a human operator/teacher and given a ”goof” button to force him to reconsider his incorrect choices [13]. Nielson's book on learning machines, which focused mostly on machine learning for pattern classification [14], was a typical book on machine learning research in the 1960s. As noted by Duda and Hart in 1973, pattern recognition interest persisted into the 1970s. A study on employing learning algorithms to teach a neural network to identify 40 characters (ten digits, four special characters, and 26 letters) from a computer terminal was given in 1981 [15].

The two main objectives of modern machine learning are to categorize data using created models and forecast outcomes using these models. A hypothetical data classification method might be trained to identify malignant moles using

supervised learning and computer vision on moles. A stock trading machine learning algorithm might provide the trader with future prediction possibilities.

1.3.2 Types of machine learning

In a world saturated with artificial intelligence, machine learning, and too much talk about them, it is interesting to learn to understand and identify the types of machine learning that one might encounter. For the average computer user, this means understanding how machine learning manifests itself in their applications. For practitioners who create these applications, it is essential to know the types of machine learning to create the right learning environment for any given task [16].

Supervised machine learning

Learning with a teacher uses a training set to teach the models to produce the desired result. This training set includes input and correct output data that allow the model to learn over time. The algorithm measures its accuracy using a loss function, adjusting until the error is sufficiently minimized.

Unsupervised machine learning

Unsupervised learning employs machine learning algorithms to examine and group unlabelled data sets. These algorithms autonomously identify concealed patterns or clusters within data, eliminating the necessity for human involvement. The capacity to identify parallels and dissimilarities in data makes it well-suited for exploratory data analysis, cross-selling tactics, consumer segmentation, and image recognition [17].

Deep learning

Deep learning technology is built on artificial neural networks. They are given both the algorithm and the data to perform this learning, which is constantly increasing in volume. The more information the neural networks receive, the more influential the learning process will be.

1.3.3 Common machine learning algorithms

Various machine learning algorithms are commonly used. These include the following.

Neural Networks

Neural networks, which have several interconnected processing nodes, resemble the human brain. Because of their superior ability to recognize patterns, neural networks are widely used in speech recognition, picture recognition, natural language translation, and image creation applications.

Linear regression

This method uses a linear connection between several variables to predict numerical numbers. This approach, for instance, may be used to forecast home values using past neighborhood data.

Logistic regression

This supervised learning algorithm makes predictions for categorical response variables such as yes/no answers to questions. It can be used for applications like spam classification and quality control on a production line [18].

2. Available solutions

Our program is distinguishing features are its open code and free distribution. However, there are other solutions from both large companies and small developers. Below, we will compare the functionality, pros, and cons of third-party programs that perform similar tasks.

2.1 Paid solutions

As a rule, paid PDF recognition software is distributed on a monthly/annual product subscription basis. This is a logical decision since large companies with thousands of employees are involved in its development. Typically, paid programs have a pleasant and functional interface with many internal modules to edit PDF documents.

2.1.1 Adobe Acrobat DC

Adobe Acrobat is a software package developed in 1993 by Adobe Systems to create and view electronic publications in PDF format.

As we said earlier, Adobe is the progenitor of the PDF format. The internal capabilities of this program are vast, so in addition to the built-in OCR in Adobe Acrobat, it presents the ability to edit documents, comment on individual lines/-words, and work with the protection of PDF documents. The subscription for this program starts at 15 euros per month.

Pros

- Stability / Compatibility
- Ease of use
- Functionality

Cons

- Expensive
- Text recognition software is not exclusive
- Large load on the system
- Takes up a lot of hard drive space
- Difficult to integrate with services such as Sharepoint or Dropbox
- Requires an Adobe Creative Cloud license [19]

2.1.2 ABBYY FineReader

ABBYY FineReader PDF is a text recognition program supporting editing PDF files. The program allows you to convert graphical documents into editable electronic formats. This program is one of the few that emphasizes the possibilities and improvement of file recognition algorithms. The price of this software starts at 18 euros per month.

Pros

- Easy-to-use text recognition editor for manually correcting documents
- Unique document comparison function
- User-friendly interface
- Export to multiple formats

Cons

- It lacks full-text indexing for quick searches
- The prices of the software are quite high
- It is not possible to view the history of changes in a document
- It is not possible to merge several files into one
- May require some post-processing of the scanned files
- Slow processing of large files

2.2 Free Solutions

As a rule, free programs for working with PDF file recognition result from the development of single programmers or small teams of developers. Any attractive interface does not distinguish such programs; as a rule, a graphical interface is absent. In programs distributed free, usually using open source OCR, which previously trained and coached. However, due to the lack of costs for their servers and a whole staff of employees, the software is free and free to use.

2.2.1 OCRmyPDF

OCRmyPDF adds an optical character recognition (OCR) text layer to scanned PDF files, allowing you to search through them. This program makes applying image processing and OCR to an existing PDF file easy.

This project is large, has sponsors, and has more than 7 thousand stars on the GitHub platform. The program is in Python and uses Google's Tesseract document recognition software.

OCRmyPDF is available at `https://ocrmypdf.readthedocs.io/en/latest/`

The software creator is a user of **github.com** under the nickname **jbarlow83**.

Pros

- Creates a searchable PDF file from a regular PDF
- Places OCR text exactly below the image for easy copy/paste
- Optimizes PDF images, often creating smaller files than the input file

Cons

- Lack of graphical interface

2.2.2 VietOCR

This program is written in Java/.NET with a graphical interface for the Tesseract text recognition engine. It supports optical character recognition for Vietnamese and other languages supported by Tesseract.

It is a small project with a user-friendly graphical interface to work with the recognized document.

VietOCR is available at `https://sourceforge.net/projects/vietocr/`

The software creator uses **sourceforge.net** under the nickname **Quan Nguyen**.

Pros

- Supports automatic download and installation of language packs
- Ability to select ROI
- Spell check with hunspell
- Ability to select a file by dragging in the GUI window
- Inserting an image from the clipboard

Cons

- Visible disadvantages were not noticed.

3. Proposed solution

Most documents contain text, lines, and graphics. Successful document analysis depends on the proper segmentation of these elements. Distinguishing graphics from text is essential for the correct operation of OCR, which otherwise produces meaningless text due to the presence of nontext components. However, it is challenging to distinguish text from graphics because graphics have similar properties. Segmenting text from graphics is a classic document image analysis problem and is still being solved by many studies. However, a reliable method is still needed to detect all possible graphics and text types in documents. This paper will try a basic segmentation when we find segments, image areas, and characters. We will not look for characters directly; we first find lines in the document, which we remove from the image.Then, we find the non-textual elements in the image, which we also remove. After these operations, we should have an image containing only textual components.

3.1 Technologies used

The OCR module built into the Tesseract software is the primary tool.

3.1.1 Tesseract

Tesseract, initially developed by Hewlett-Packard as proprietary OCR software, originated in the mid-1980s and continued its development until the mid-1990s. After nearly a decade of inactivity, where it remained unused, or "on the shelf," Google acquired it in August 2006, releasing its source code under the Apache 2.0 license to foster broader development and application across diverse fields [20].

From 1985 to 1994, the initial version of Tesseract was crafted within the confines of Hewlett-Packard's research labs in Bristol, England, and Greeley, Colorado. The software was originally written in C programming language, which was subsequently upgraded to C++ to enhance its capabilities and adapt to evolving technological frameworks. This shift also included transitioning all subsequent code developments to be compatible with C++ compilers, marking a significant evolution in its underlying technology. However, after this phase of intense development, Tesseract experienced a period of dormancy where little to no progress was made, until its revival in 2005 when Hewlett-Packard and the University of Nevada, Las Vegas (UNLV) decided to open-source the project. This move significantly expanded its accessibility to developers globally.

Since its acquisition, Google has actively sponsored the development of Tesseract, resulting in significant enhancements such as the integration of an LSTM (Long Short-Term Memory) based OCR engine. This advancement broadened Tesseract's utility by extending its linguistic capabilities to include 116 languages and 37 different scripts, significantly improving its recognition accuracy. Such capabilities make it possible to effectively process documents that blend text from both Central and Western European languages using the Latin alphabet, among others. This expansion has not only improved the quality and flexibility

of Tesseract but also increased its appeal as a versatile tool in the field of optical character recognition [21].

The continued development and expansion of Tesseract reflect a commitment to improving digital text recognition technologies, making them more adaptable and efficient in handling a wide range of languages and scripts. This ongoing evolution aims to meet the increasing demands of globalization and digital documentation in numerous sectors.

Recurrent neural networks

Human cognition does not reset with every blink; it builds on accumulated knowledge. This continuity ensures that when you read this undergraduate thesis, your understanding of each sentence is informed by the context established by the preceding text. Our thoughts are not fleeting—they have a sustained presence, shaped by what we have already processed.

This characteristic of persistent memory is absent in traditional neural networks, marking a significant limitation in their application. These networks typically operate without the ability to retain prior information, which means each new input is processed in isolation. Traditional neural networks face difficulties in classifying sequences of events in movies because to their inability to retain information about prior scenes, which is crucial for comprehending the continuing narrative in a contextual manner.

However, this gap is bridged by Recurrent Neural Networks (RNNs). RNNs are designed to overcome the memory limitations of traditional neural networks by incorporating loops in their architecture, allowing information to persist. This architecture enables RNNs to maintain a state or memory of previous inputs and use this stored information to influence the processing of new inputs. This capability makes RNNs particularly effective for tasks that require an understanding of sequence and context, such as speech recognition, language translation, and indeed, categorizing events in films where the understanding of prior developments enhances the interpretation of subsequent actions.

LSTM

The capacity of Long Short-Term Memory (LSTM) networks to learn dependencies over extended periods of time sets them out as a specific kind of Recurrent Neural Network (RNN) architecture. When Sepp Hochreiter and Jürgen Schmidhuber initially introduced this design in 1997, it was a major breakthrough in the area. Since many researchers have improved and built upon their early work over time, LSTMs' effective management of long-term dependencies makes them a solid answer for a range of challenging sequence prediction issues, from language processing to stock market prediction [23].

Long-term dependencies are a typical problem for ordinary RNNs, which is expressly addressed and resolved in the design of LSTMs. LSTMs are good at retaining information for lengthy periods of time without running the danger of losing crucial details over time, in contrast to standard RNNs, which find it difficult to retain knowledge throughout longer sequences.

All RNNs, including LSTMs, are structurally made up of a series of network modules connected together in a chain. To provide non-linearity to the processing

of input data, each module in a basic RNN normally has a simple layer with a non-linear activation function like tanh (hyperbolic tangent) [22]. LSTM modules, on the other hand, are more sophisticated and include many gates that govern the information flow, so determining what is remembered and what is discarded. This improves their capacity to handle long-term data without deteriorating the signal.

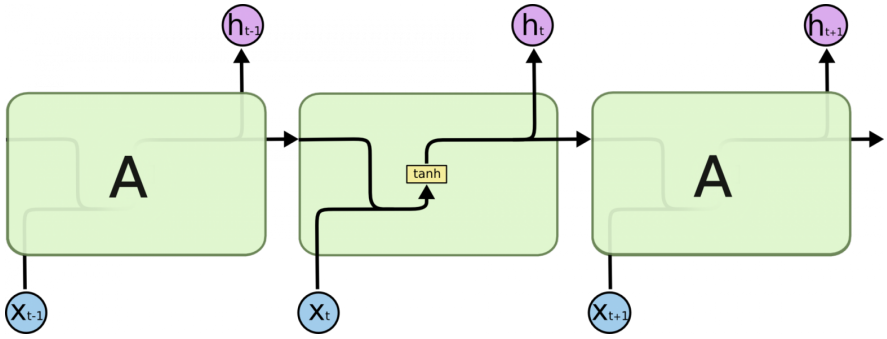


Figure 3.1: The usual RNN structure of a single module [28]

The repeating module in the standard RNN consists of one layer

Long Short-Term Memory (LSTM) networks have a structure similar to classic Recurrent Neural Networks (RNNs), but with more complicated modules. Four interacting layers are present in LSTM units, as opposed to the single layer that is usually seen in an RNN. LSTMs can handle long-term dependencies more successfully than regular RNNs because these layers work together to maintain the memory of the network. LSTMs can more precisely regulate the flow of information throughout the network because to this multi-layered configuration, which includes input, forget, and output gates.

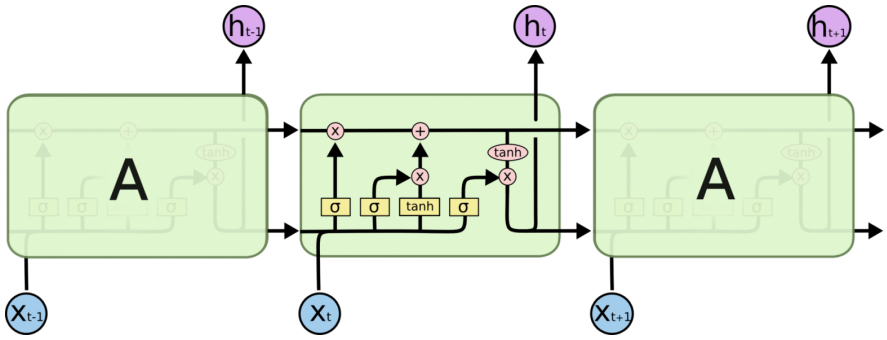


Figure 3.2: LSTM network of four interacting layers [28]

The repeating model in the LSTM network consists of four interacting layers

Let's comprehend the essential elements first, then dive into the complexity. In the sections that follow, we will go into the specifics of each step inside the LSTM framework. In order to better understand how these many layers interact to process and retain information over time, let's first become familiar with the individual components and terminology utilized in these networks.

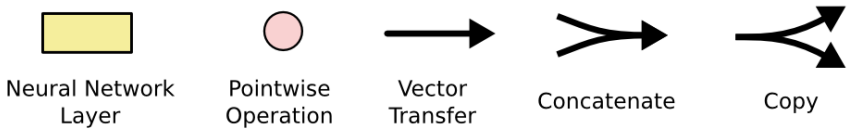


Figure 3.3: Special designations in the LSTM model [28]

- Neural Network Layer
- Pointwise Operation
- Vector Transfer
- Concatenate
- Copy

Each line in the following diagram transfers a whole vector from one node's output to the next node's input. Pointwise operations, such as vector addition, are represented by the pink circles and are essential for merging and manipulating data. In the meanwhile, the layers of the neural network that have been taught to efficiently handle and interpret the input are represented by the yellow rectangles. Converging lines indicate the joining of data streams, which improves the network's capacity to incorporate data from different sources. On the other hand, branching arrows represent data replication, making sure that exact copies are dispersed across the network for concurrent processing [25].

The basic idea of LSTM

The cell state is a crucial element of the LSTM architecture and is represented by a horizontal line that intersects the top of the diagram.

This cell state is similar to a conveyor belt that runs smoothly throughout the whole network. It does very little in the way of linear modifications, letting data flow along its length unaltered. In order to preserve long-term dependencies in the data being processed, this design makes sure that information integrity is preserved as it flows from one module to another [23].

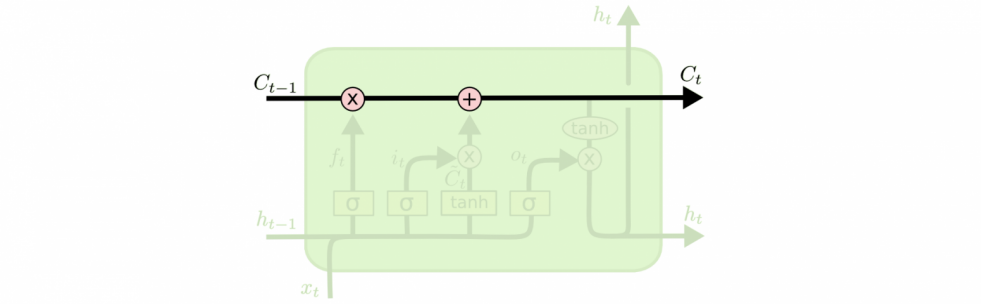


Figure 3.4: The cell state in the LSTM model [28]

Nevertheless, by employing gates or filters, LSTM has the capability to choose disregard some information from the cell state.

By permitting particular data to be skipped or passed through depending on predetermined criteria, these filters have the power to regulate the flow of information. An LSTM filter is structurally composed of a sigmoidal neural network layer that determines how much information to keep and a pointwise multiplication operation that carries out the decision [26]. This arrangement effectively controls data flow inside the network.

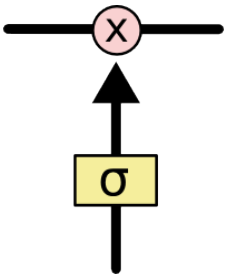


Figure 3.5: The structure of filters in the LSTM model [28]

Values between zero and one, which indicate how much information should be conveyed via the network, are produced by the sigmoidal layer of an LSTM. More specifically, zero means that no information should be sent along ("skip everything"), while one means that all information should be kept ("skip nothing").

LSTM by Tesseract 4

The presence of a high number of states in Long Short-Term Memory (LSTM) models leads to a decrease in their processing speed, notwithstanding their proficiency in learning sequences. Empirical evidence suggests that providing a lengthy sequence for LSTM to learn is more advantageous than giving it a short sequence with several courses. Tesseract is constructed using the Python OCRopus framework, which was taken from the C++ LSMT branch called CLSTM. The CLSTM library is used to implement the LSTM recurrent neural network model in C++, utilizing the Eigen package for numerical computations.

TensorFlow is not related to Tesseract's neural network framework. But it works with it since Variable Graph Specification Language (VGSL), a network description language, is also compatible with TensorFlow.

The premise of VGSL is that a neural network may be constructed and trained with little to no knowledge. Python, TensorFlow, or even C++ code writing are not required. Comprehending the VGSL specification language enough is all that is needed to construct syntactically accurate network descriptions. It will be imperative to possess a fundamental understanding of the various neural network layer types and their combinations [27].

3.2 Description of the Graphical User Interface

This chapter provides an overview of the user interface of the program for optical text recognition from PDF. It describes the basic controls that allow the user to customize the recognition process, including file selection, text language detection, and page orientation correction. Advanced settings are also considered,

which provide deeper possibilities for optimizing the quality of OCR, and the function of resetting settings to standard values.

3.2.1 The Main Application Window

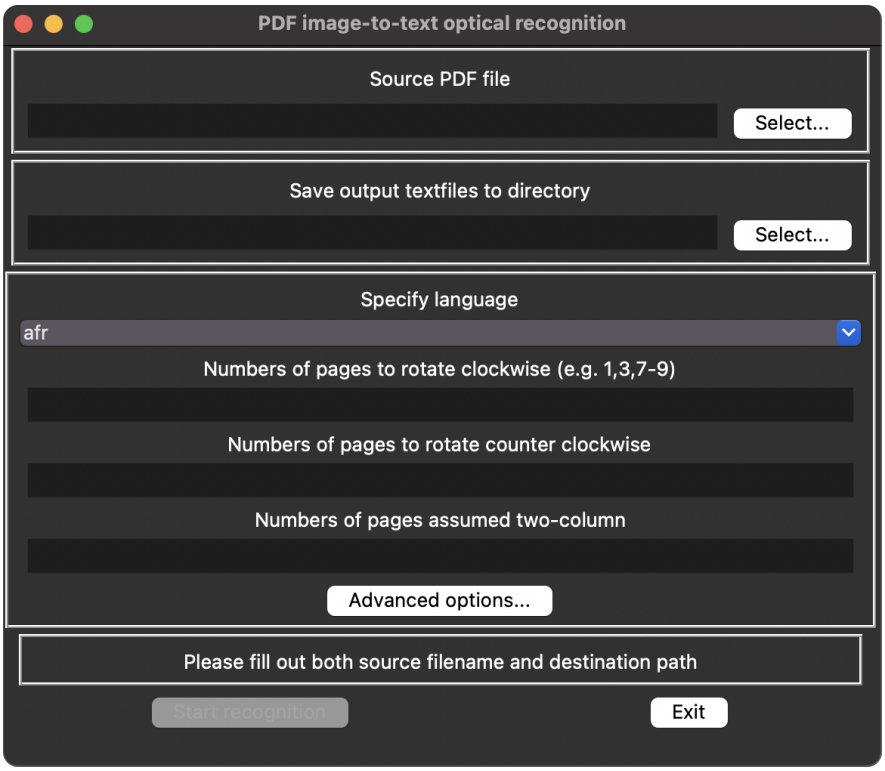


Figure 3.6: The Main Application Window

The main application window contains the following elements for user interaction:

- **Window Title**: "PDF image-to-text optical recognition", which clearly indicates its purpose.
- **Source PDF File Selection Field**: An input field with an accompanying "Select..." button provides the user with the convenience of easily uploading the source PDF file for processing.
- **Directory Selection Field for Saving Results**: Similarly, an input field with a "Select..." button allows the user to define the folder where the text files will be saved after recognition.
- **Language Selection for Recognition**: The drop-down list contains language codes (e.g., "afr" for Afrikaans), allowing the user to select the language of the text in the PDF document for accurate recognition.
- **Page Rotation Fields**: Users can specify page numbers that need to be rotated clockwise or counterclockwise to ensure proper orientation before OCR. This is crucial because text cannot be accurately recognized if the images are scanned at an incorrect angle.

- **Two-Column Format Specification Field**: This field allows the user to specify page numbers formatted in two columns. It enables the algorithm to properly handle page layout and ensures correct text recognition.
- **Advanced Options Button**: Provides access to deeper recognition algorithm settings, giving experienced users the ability to finely tune the OCR process according to their requirements.
- **Start Recognition Button**: Initiates the recognition process after all necessary parameters have been set. This is the main functional button that triggers the core workflow of the application.
- **Informational Message**: Below is a message with instructions for the user, indicating the need to fill in all mandatory fields before starting the program.
- **Exit Button**: Allows the user to close the application at any time, providing an opportunity to exit without starting or after completing the OCR process.

3.2.2 Advanced Options Window

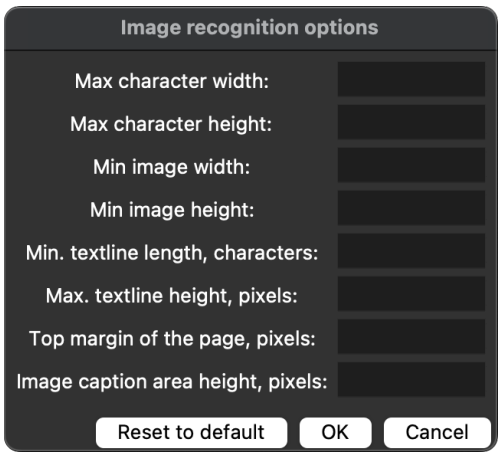


Figure 3.7: GUI: Empty Fields

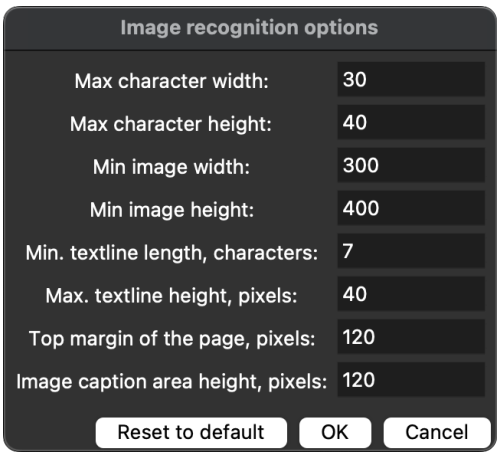


Figure 3.8: GUI: Fields Populated

The GUI's "Advanced Options" window presents a set of parameters that allow users to refine the criteria used during the image recognition process. Here's a detailed description of each element:

- **Max character width**: The maximum width in pixels that any single character can occupy. Default value is 30 pixels.
- **Max character height**: The maximum height in pixels that any single character can have. Default value is 40 pixels.
- **Min image width**: The minimum width in pixels that an image within the document must have to be recognized as an image. Default value is 300 pixels.

- **Min image height**: The minimum height in pixels that an image within the document must have to be recognized. Default value is 400 pixels.
- **Min. textline length, characters**: The minimum length of a line of text, in characters, to be considered as a valid text line. Default value is 7 characters.
- **Max. textline height, pixels**: The maximum height in pixels that a line of text can have. Default value is 40 pixels.
- **Top margin of the page, pixels**: The height in pixels of the top margin of the page. It is the space at the top before the text starts. Default value is 120 pixels.
- **Image caption area height, pixels**: The height in pixels reserved for the image caption area. Default value is 120 pixels.

The window has two states: the first with empty fields, indicating that no user-defined values have been entered, and the second state with pre-populated default values, which are activated when the **"Reset to Default"** button is pressed.

The **"Reset to Default"** button is used to quickly revert all fields to their original settings, facilitating a return to the baseline configuration without the need to manually enter each value.

The **"OK"** button confirms the changes made by the user and applies the settings to the image recognition process.

The **"Cancel"** button closes the window without applying any changes, retaining any previous configurations.

This advanced configuration provides nuanced control over the OCR process, allowing for fine-tuning according to the specific needs of the document being processed.

3.3 Work Scheme

Our Python-based algorithm for image extraction from text documents, such as PDF files, follows a carefully structured process involving various libraries, techniques, and parameters to handle text and non-text regions effectively.

3.3.1 Input and Output Specifications

Input Data

PDF Document: The program accepts a PDF document as input. The document can contain text, graphic elements, and images on different pages.

Output Data

Segmented Pages (PNG): Each page of the PDF document is processed and saved as an individual PNG file, facilitating easy viewing and processing of each page separately. Example file names include page-1.png, page-2.png, etc.

Deciphered Text (TXT): Text from each page is extracted and saved in separate text files, providing access to the textual content of each page in a readable format. Example file names include page-1.txt, page-2.txt, etc.

Images with Descriptions (Folders): All images found on each page of the PDF document are cut out and saved in separate folders corresponding to each page. Each folder contains images and their descriptions, making it easier to identify and utilize these images. Example folder names include page-images-1, page-images-2, etc.

3.3.2 Python Script

The primary script of the algorithm, written in Python, uses the following modules and libraries:

- *os, sys, copy* - Standard Python modules for various system-level interactions.
- *argparse* - A standard Python library for command-line arguments and parameters.
- *collections.namedtuple* - A factory function for creating tuple subclasses with named fields, used for storing bounding box data.
- *PIL (Python Imaging Library)* - A library for opening, manipulating, and saving many different image file formats.
- *pytesseract* - A wrapper for Google's Tesseract-OCR Engine, used for character recognition in this context.

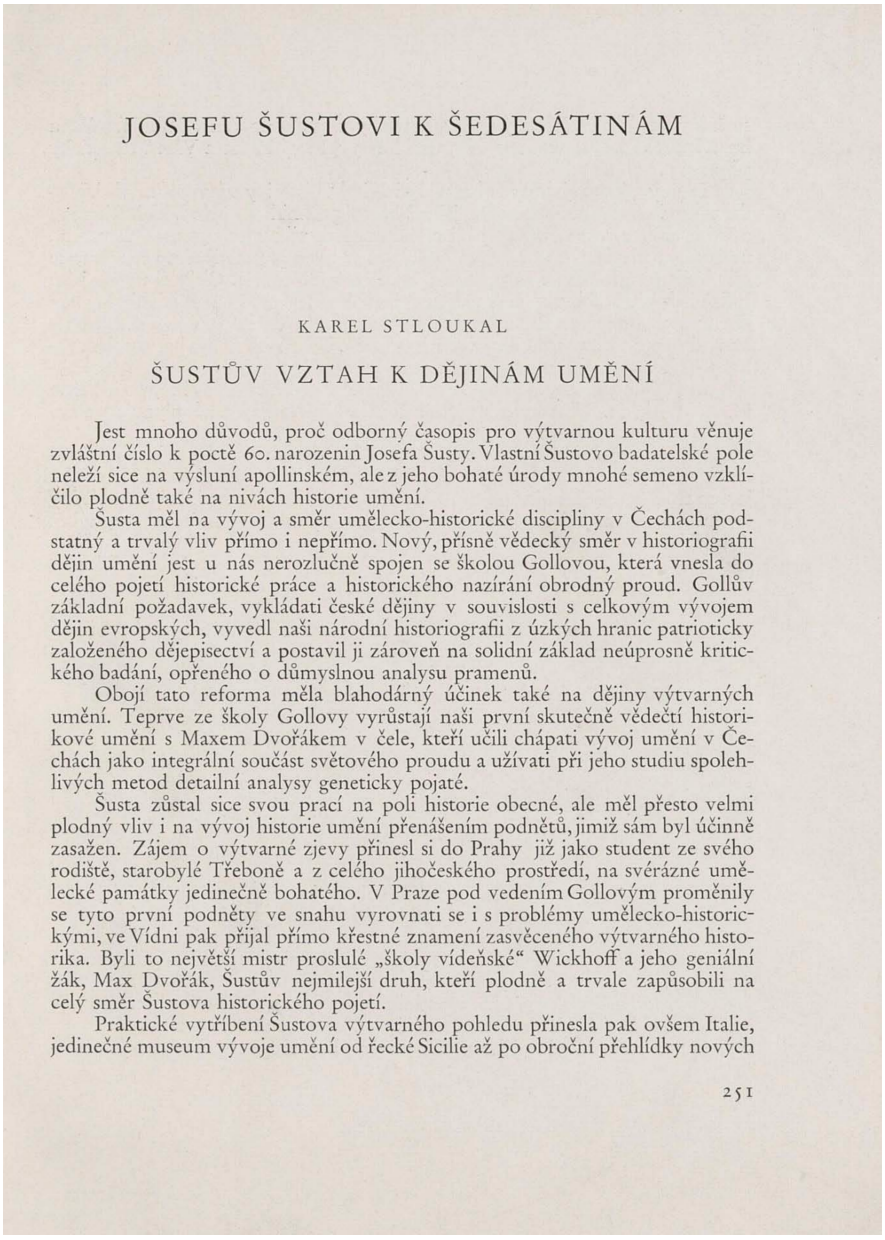
Initialization

In the initialization phase, the algorithm accepts an image or a file path containing an image and a configuration dictionary as input. The configuration dictionary contains various parameters that dictate the algorithm's behavior, such as maximum character width, minimum image width, output path, language, etc. If a file path is provided, the Image class from the PIL library opens the image, and the resultant Image object is stored for future use.

Processing Image

The algorithm then executes the *run()* function, which is responsible for the following tasks:

- Extracting character boxes
- Identifying lines and columns of text
- Detecting and saving potential illustrations as separate images



JOSEFU ŠUSTOVI K ŠEDESÁTINÁM

KAREL STLOUKAL

ŠUSTŮV VZTAH K DĚJINÁM UMĚNÍ

Jest mnoho důvodů, proč odborný časopis pro výtvarnou kulturu věnuje zvláštní číslo k poctě 60. narozenin Josefa Šusty. Vlastní Šustovo badatelské pole neleží sice na výsluní apollinském, ale z jeho bohaté úrody mnohé semeno vzklíčilo plodně také na nivách historie umění.

Šusta měl na vývoj a směr umělecko-historické discipliny v Čechách podstatný a trvalý vliv přímo i nepřímo. Nový, přísně vědecký směr v historiografii dějin umění jest u nás nerozlučně spojen se školou Gollovou, která vnesla do celého pojetí historické práce a historického nazírání obrodný proud. Gollův základní požadavek, vykládati české dějiny v souvislosti s celkovým vývojem dějin evropských, vyvedl naši národní historiografii z úzkých hranic patrioticky založeného dějepisectví a postavil ji zároveň na solidní základ neúprosně kritického badání, opřeného o důmyslnou analysu pramenů.

Obojí tato reforma měla blahodárný účinek také na dějiny výtvarných umění. Teprve ze školy Gollovy vyrůstají naši první skutečně vědečtí historikové umění s Maxem Dvořákem v čele, kteří učili chápati vývoj umění v Čechách jako integrální součást světového proudu a užívati při jeho studiu spolehlivých metod detailní analysy geneticky pojaté.

Šusta zůstal sice svou prací na poli historie obecné, ale měl přesto velmi plodný vliv i na vývoj historie umění přenášením podnětů, jimiž sám byl účinně zasažen. Zájem o výtvarné zjevy přinesl si do Prahy již jako student ze svého rodiště, starobylé Třeboně a z celého jihočeského prostředí, na svérázné umělecké památky jedinečně bohatého. V Praze pod vedením Gollovým proměnily se tyto první podněty ve snahu vyrovnati se i s problémy umělecko-historickými, ve Vídni pak přijal přímo křestné znamení zasvěceného výtvarného historika. Byli to největší mistr proslulé „školy vídeňské" Wickhoff a jeho geniální žák, Max Dvořák, Šustův nejmilejší druh, kteří plodně a trvale zapůsobili na celý směr Šustova historického pojetí.

Praktické vytříbení Šustova výtvarného pohledu přinesla pak ovšem Italie, jedinečné museum vývoje umění od řecké Sicilie až po obroční přehlídky nových

251

Figure 3.9: Original PDF Page

Character Boxes Extraction The algorithm uses the function below to retrieve the data from the binding box for each character in the image.

`pytesseract.image_to_boxes()`

It discards any box exceeding the maximum dimensions of a character, as defined in the configuration, to distinguish between text and non-text elements.

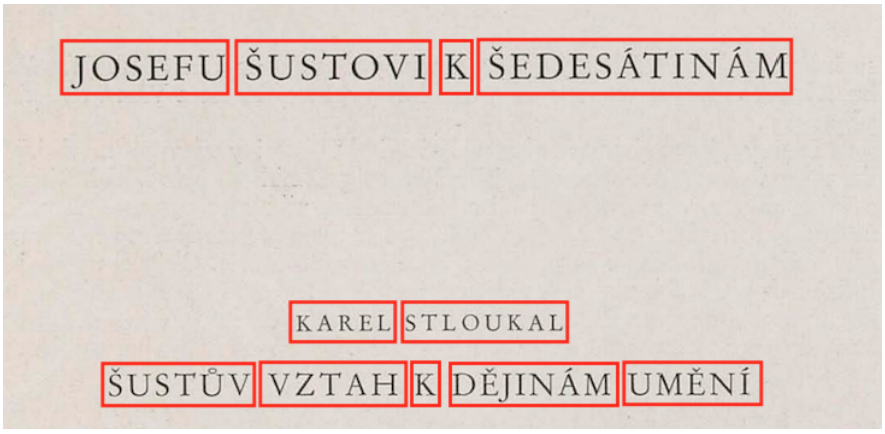


Figure 3.10: Text Region of Interest (ROI) Identification

The data for each bounding box are stored as a named tuple, "Box," comprising the recognized character and the bounding box coordinates.

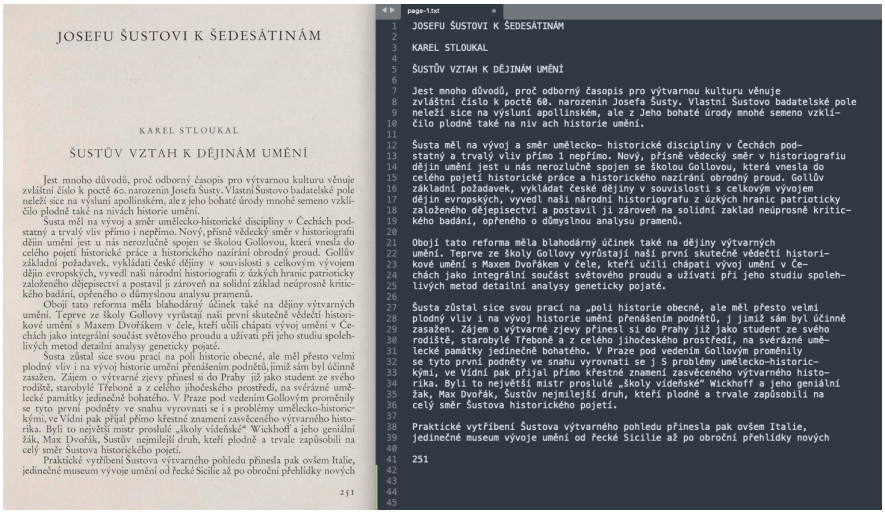


Figure 3.11: Comparison of Original PDF and Recognized Text

Text Lines and Columns Identification The algorithm identifies lines and text columns based on each character's bounding box data. It first groups the bounding boxes into lines and then columns based on their horizontal and vertical positions.

This allows the algorithm to process the text column by column, essential for documents formatted in multiple columns.

Potential Illustrations Detection In between text lines, if a gap exceeds the defined minimum image height, the algorithm assumes that an illustration is present. It crops this region from the original image and saves it as a separate image file in the output directory. The region is slightly expanded vertically to ensure complete capture of the illustration.

Two-Columns Document Handling

For documents formatted in two columns, the algorithm processes each column independently by creating a separate instance of the ImageExtractor class for each column. This ensures accurate line and column identification, as well as illustration detection, in two-column documents.

3.3.3 Algorithm Flowchart

The process begins with loading the PDF document for analysis, which involves initializing the system to read the PDF content. The algorithm determines whether the document contains vector images, crucial for selecting the processing method:

- **For vector images**: Images are isolated and saved separately, usually requiring fewer transformations and providing cleaner data for analysis.
- **For raster images**: The process involves identifying and extracting raster images, often found in scanned documents.

Image Rotation: If necessary, images within the document can be rotated according to specified settings to ensure proper orientation, which is critical for accurate text recognition.

Applying OCR: During the OCR (Optical Character Recognition) stage, such as with the Tesseract module, text is recognized and extracted from images. This process involves analyzing text areas and converting visual content into machine-readable text.

Saving the results: After processing the images and text, the results are saved as segmented images, extracted text, and associated descriptions, ensuring ease of access and use of the PDF data subsequently.

Completion of the Process: The process concludes after all operations have been performed and the results have been saved.

The flowchart 3.12 presents a comprehensive visualization of the algorithm's sequence of operations. It details the workflow, starting from the initial loading of the PDF document to the final stages where processed results are saved. Each step in the flowchart corresponds to a critical function within the algorithm, including image isolation, rotation, and text extraction using OCR. The structured layout of the flowchart delineates the procedural advancements and the interdependencies between different operational stages, thereby outlining the systematic approach employed in the image search program for PDF documents.

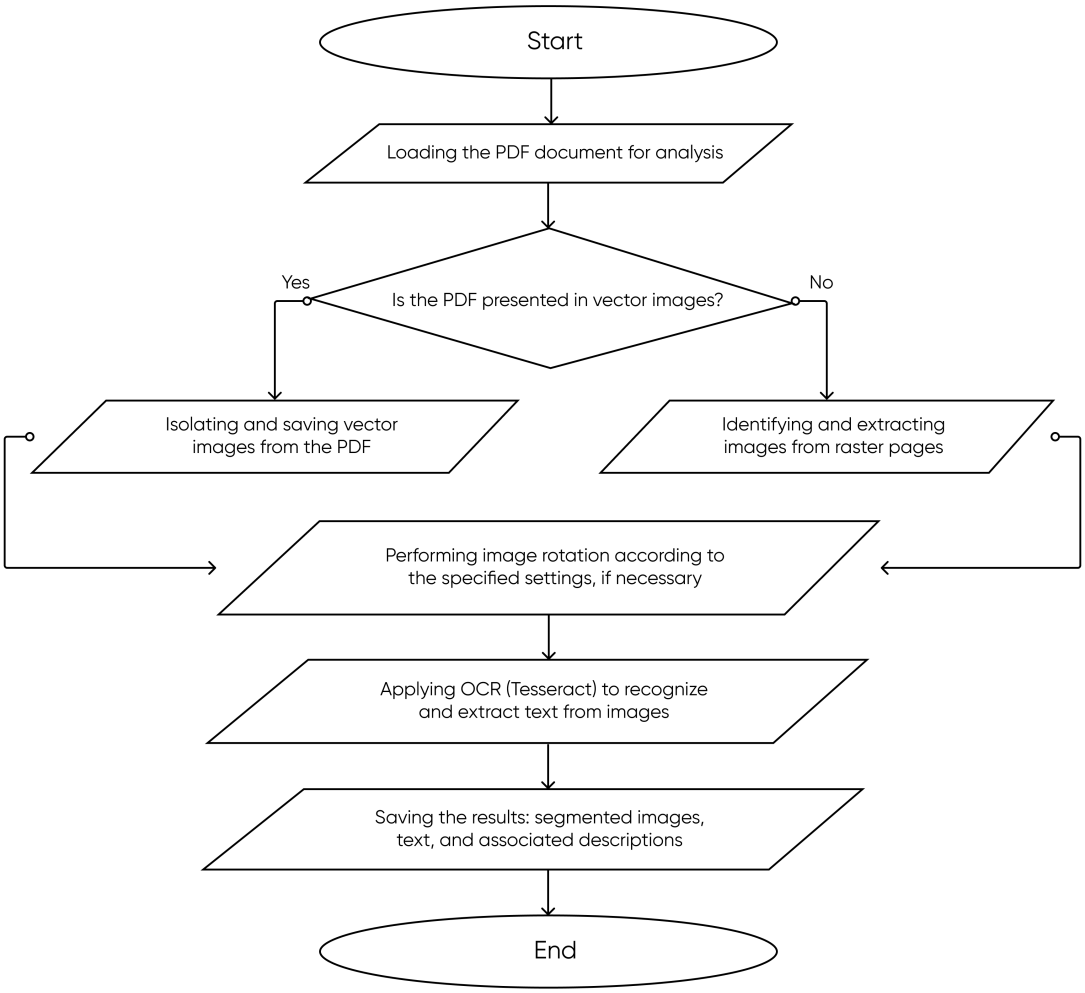


Figure 3.12: Algorithm Flowchart

3.4 Analysis of the algorithm

In the course of developing and refining our text and image recognition software, we conducted a comprehensive study to assess the efficiency of our algorithm. This research utilized five scanned books; unfortunately, the quality of these scans was suboptimal. These were PDF documents of books published around the 1900s, comprising solely scanned pages of the original texts. The study focuses on the algorithm's capacity to process and recognize text and images across different volumes and types of data under these challenging conditions. The results, which are detailed in this chapter, illustrate the effectiveness of our algorithm in handling diverse tasks and provide insights into both its strengths and areas for potential improvement.

Table 3.1: Overview of Book Processing Data

Book	Pages	Characters	Images	Reading Time	Text Proc. Time	Image Proc. Time
1	56	46,678	0	16	112	41
2	64	58,055	38	52	209	75
3	24	43,575	15	5	170	14
4	40	27,766	7	13	81	24
5	8	21,984	0	13	44	15

Book Data Analysis

Book 1

Book 1 contained 46,678 characters distributed across 56 pages, resulting in an average of 833 characters per page. The text processing time was 112 seconds, indicating good algorithm performance considering the relatively small amount of text per page.

Book 2

This book, with 58,055 characters on 64 pages (approximately 907 characters per page), showed a longer text processing time of 209 seconds. Despite only a slight increase in text volume compared to Book 1, the processing time nearly doubled, which may indicate more complex or densely packed text.

Book 3

With 43,575 characters on just 24 pages, this book demonstrated the highest text density—averaging 1,815 characters per page. The text processing time was 170 seconds, making it one of the most challenging for the algorithm due to its high text density.

Book 4

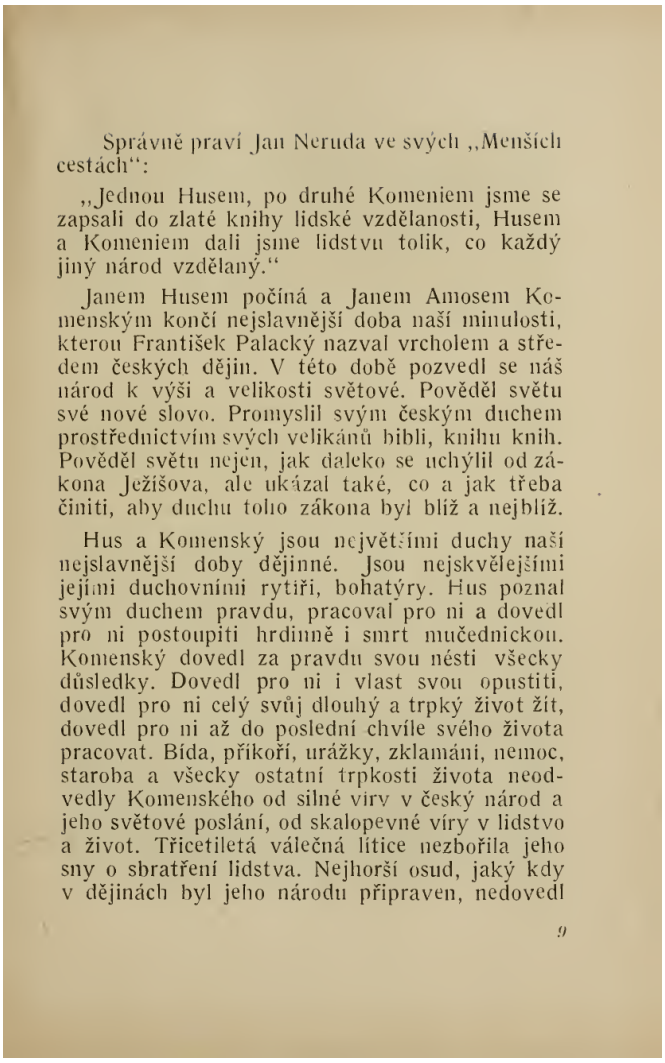
This book had 27,766 characters on 40 pages, averaging 694 characters per page. Although this is the lowest average character count, the processing time of 81 seconds was quite efficient, highlighting that the algorithm performs better with fewer characters per page.

Book 5

The smallest book by volume, with 8 pages and 21,984 characters (2,748 characters per page), showed a processing time of 44 seconds. This demonstrates that even with a high density of characters, the algorithm can be efficient if the total number of pages is small.

Visual Comparison of Pages

In the course of our research, we conducted a detailed analysis on pages with varying amounts of text to observe the performance of our algorithm under different conditions. Below are images 3.13 3.14 showcasing two specific pages that demonstrate how text volume impacts processing speed.



Správně praví Jan Neruda ve svých „Menších cestách“:

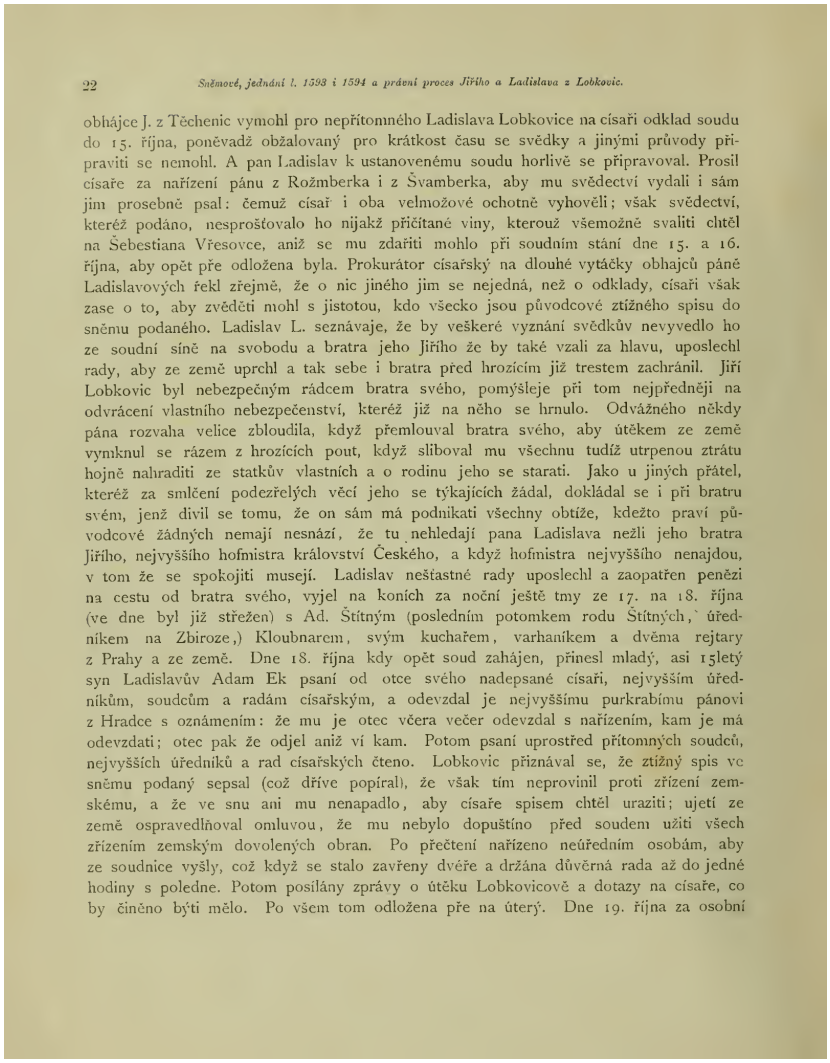
„Jednou Husem, po druhé Komeniem jsme se zapsali do zlaté knihy lidské vzdělanosti, Husem a Komeniem dali jsme lidstvu tolik, co každý jiný národ vzdělaný.“

Janem Husem počíná a Janem Amosem Komenským končí nejslavnější doba naší minulosti, kterou František Palacký nazval vrcholem a středem českých dějin. V této době pozvedl se náš národ k výši a velikosti světové. Pověděl světu své nové slovo. Promyslil svým českým duchem prostřednictvím svých velikánů bibli, knihu knih. Pověděl světu nejen, jak daleko se uchýlil od zákona Ježíšova, ale ukázal také, co a jak třeba činiti, aby duchu toho zákona byl blíž a nejblíž.

Hus a Komenský jsou největšími duchy naší nejslavnější doby dějinné. Jsou nejskvělejšími jejími duchovními rytiři, bohatýry. Hus poznal svým duchem pravdu, pracoval pro ni a dovedl pro ni postoupiti hrdinně i smrt mučednickou. Komenský dovedl za pravdu svou nésti všecky důsledky. Dovedl pro ni i vlast svou opustiti, dovedl pro ni celý svůj dlouhý a trpký život žít, dovedl pro ni až do poslední chvíle svého života pracovat. Bída, příkoří, urážky, zklamání, nemoc, staroba a všecky ostatní trpkosti života neodvedly Komenského od silné víry v český národ a jeho světové poslání, od skalopevné víry v lidstvo a život. Třicetiletá válečná lítice nezbořila jeho sny o sbratření lidstva. Nejhorší osud, jaký kdy v dějinách byl jeho národu připraven, nedovedl

9

Figure 3.13: Page with a small number of characters (1,432 characters), recognized in approximately 4 seconds.



22 Sněmové, jednání l. 1593 i 1594 a právní proces Jiřího a Ladislava z Lobkovic.

obhájce J. z Těchenic vymohl pro nepřítomného Ladislava Lobkovice na císaři odklad soudu do 15. října, poněvadž obžalovaný pro krátkost času se svědky a jinými průvody připraviti se nemohl. A pan Ladislav k ustanovenému soudu horlivě se připravoval. Prosil císaře za nařízení pánu z Rožmberka i z Švamberka, aby mu svědectví vydali i sám jim prosebně psal: čemuž císař i oba velmožové ochotně vyhověli; však svědectví, kteréž podáno, nesprošťovalo ho nijakž přičítané viny, kterouž všemožně svaliti chtěl na Šebestiana Vřesovce, aniž se mu zdařiti mohlo při soudním stání dne 15. a 16. října, aby opět pře odložena byla. Prokurátor císařský na dlouhé vytáčky obhajců páně Ladislavových řekl zřejmě, že o nic jiného jim se nejedná, než o odklady, císaři však zase o to, aby zvěděti mohl s jistotou, kdo všecko jsou původcové ztížného spisu do sněmu podaného. Ladislav L. seznávaje, že by veškeré vyznání svědkův nevyvedlo ho ze soudní síně na svobodu a bratra jeho Jiřího že by také vzali za hlavu, uposlechl rady, aby ze země uprchl a tak sebe i bratra před hrozícím již trestem zachránil. Jiří Lobkovic byl nebezpečným rádcem bratra svého, pomýšleje při tom nejpředněji na odvrácení vlastního nebezpečenství, kteréž již na něho se hrnulo. Odvážného někdy pána rozvaha velice zbloudila, když přemlouval bratra svého, aby útěkem ze země vyniknul se rázem z hrozících pout, když sliboval mu všechnu tudíž utrpenou ztrátu hojně nahraditi ze statkův vlastních a o rodinu jeho se starati. Jako u jiných přátel, kteréž za smlčení podezřelých věcí jeho se týkajících žádal, dokládal se i při bratru svém, jenž divil se tomu, že on sám má podnikati všechny obtíže, kdežto praví původcové žádných nemají nesnází, že tu nehledají pana Ladislava nežli jeho bratra Jiřího, nejvyššího hofmistra království Českého, a když hofmistra nejvyššího nenajdou, v tom že se spokojiti musejí. Ladislav nešťastné rady uposlechl a zaopatřen penězi na cestu od bratra svého, vyjel na koních za noční ještě tmy ze 17. na 18. října (ve dne byl již střežen) s Ad. Štítným (posledním potomkem rodu Štítných,) úředníkem na Zbiroze,) Kloubnarem, svým kuchařem, varhaníkem a dvěma rejtary z Prahy a ze země. Dne 18. října kdy opět soud zahájen, přinesl mladý, asi 15letý syn Ladislavův Adam Ek psaní od otce svého nadepsané císaři, nejvyšším úředníkům, soudcům a radám císařským, a odevzdal je nejvyššímu purkrabímu pánovi z Hradce s oznámením: že mu je otec včera večer odevzdal s nařízením, kam je má odevzdati; otec pak že odjel aniž ví kam. Potom psaní uprostřed přítomných soudců, nejvyšších úředníků a rad císařských čteno. Lobkovic přiznával se, že ztížný spis ve sněmu podaný sepsal (což dříve popíral), že však tím neprovinil proti zřízení zemskému, a že ve snu ani mu nenapadlo, aby císaře spisem chtěl uraziti; ujetí ze země ospravedlňoval omluvou, že mu nebylo dopuštíno před soudem užiti všech zřízením zemským dovolených obran. Po přečtení nařízeno neúředním osobám, aby ze soudnice vyšly, což když se stalo zavřeny dvéře a držána důvěrná rada až do jedné hodiny s poledne. Potom posílány zprávy o útěku Lobkovicově a dotazy na císaře, co by činěno býti mělo. Po všem tom odložena pře na úterý. Dne 19. října za osobní

Figure 3.14: Page with a large number of characters (3,291 characters), recognized in approximately 9.2 seconds.

This visual comparison clearly illustrates the algorithm's varying efficiency based on text density. The data from these pages provide valuable insights into optimizing the algorithm for different types of text-loaded documents, enhancing overall performance.

3.4.1 Analysis of Text Recognition Speed

Methodology

To determine the average text recognition speed, we use the following formula:

$$\text{Average Recognition Speed (characters per second)} = \frac{\text{Total Characters}}{\text{Total Processing Time (s)}}$$

Statistical Analysis

We calculate the average speed of text recognition for each book, and analyze the dispersion using variance and standard deviation. Additionally, we compute

the overall average speed across all books to evaluate the general efficiency of the algorithm.

Results

- **Book 1:** Total Characters = 46,678, Total Time = 112 s.

$$\text{Speed} = \frac{46,678}{112} \approx 417 \text{ characters per second}$$

- **Book 2:** Total Characters = 58,055, Total Time = 209 s.

$$\text{Speed} = \frac{58,055}{209} \approx 278 \text{ characters per second}$$

- **Book 3:** Total Characters = 43,575, Total Time = 170 s.

$$\text{Speed} = \frac{43,575}{170} \approx 256 \text{ characters per second}$$

- **Book 4:** Total Characters = 27,766, Total Time = 81 s.

$$\text{Speed} = \frac{27,766}{81} \approx 343 \text{ characters per second}$$

- **Book 5:** Total Characters = 21,984, Total Time = 44 s.

$$\text{Speed} = \frac{21,984}{44} \approx 500 \text{ characters per second}$$

Overall Average Speed

To calculate the overall average recognition speed, we take the average of the speeds obtained for each book:

$$\text{Overall Average Speed} = \frac{417 + 278 + 256 + 343 + 500}{5} \approx 359 \text{ characters per second}$$

Conclusion

This analysis of text recognition speed across different books has highlighted significant variability in performance, which is influenced by factors such as the number of characters and the complexity of the text on each page. The calculated average recognition speeds for each book provide a clear indicator of the algorithm's efficiency in various conditions. The overall average speed of approximately 359 characters per second offers a benchmark for assessing the general efficiency of our text recognition system. This benchmark can serve as a guide for future optimizations and enhancements to the algorithm, aiming to improve consistency and reduce processing times across a wider range of text densities and complexities.

3.4.2 Analysis of Image Recognition Efficiency

Background

Image recognition in document processing is notably complex due to variations in image types, sizes, and the contextual information they contain. This complexity increases the computational demands, especially when images are distributed throughout the document.

Methodology

To evaluate image recognition efficiency, we consider the total number of pages, as images need to be detected on each page, not just where they are known to exist. We calculate the average time per image by considering the number of pages:

$$\text{Average Image Processing Time (s)} = \frac{\text{Total Image Processing Time (s)}}{\text{Total Number of Pages with Images}}$$

Data Collection

Data regarding the total number of pages, images, and corresponding processing times is collected from each book that includes images.

Analysis

- **Book 2:** Contains 38 images across 64 pages, with a total image processing time of 75 seconds.

$$\text{Average Time per Image per Page} = \frac{75}{64} \approx 1.17 \text{ seconds per page}$$

- **Book 3:** Features 15 images across 24 pages, processed in 14 seconds.

$$\text{Average Time per Image per Page} = \frac{14}{24} \approx 0.58 \text{ seconds per page}$$

- **Book 4:** Includes 7 images on 40 pages, with a total processing time of 24 seconds.

$$\text{Average Time per Image per Page} = \frac{24}{40} \approx 0.60 \text{ seconds per page}$$

Overall Average Image Processing Speed

To calculate the overall average image processing speed across the books that include images, we take the average of the image processing speeds obtained for each book:

$$\text{Avg. Image Speed (sec/page)} = \frac{1.17 + 0.58 + 0.60}{3} \approx 0.78 \text{ sec/page}$$

Statistical Analysis

Statistical analysis involves assessing the variability and efficiency of image recognition across books, with a particular focus on the average processing time per page containing images. This will help understand the dispersion and reliability of the image recognition process.

Conclusion

This detailed analysis underscores the impact of page volume on image processing times. The findings indicate significant variability in efficiency, driven by the number and complexity of images relative to page count. These insights guide future optimizations to enhance image processing strategies, aiming to reduce processing time while maintaining accuracy across a diverse set of document types.

3.4.3 Integration of Libraries and Modules

The effective functioning of our algorithm relies on the seamless integration of various Python libraries, each playing a crucial role in the image processing and OCR workflow.

3.4.4 Program Limitations

- **Input Format:** The program exclusively accepts PDF documents. Other file formats such as PNG, JPG, etc., are not supported.
- **Handling Columnar Pages:** The program does not automatically recognize pages where text is divided into columns. However, the GUI allows manual specification of such pages.
- **Automatic Language Recognition:** The language of the text is not automatically determined and must be manually set through the GUI.
- **Page Rotation Recognition:** The program does not automatically recognize rotated pages. The GUI allows specification of rotated pages and the direction of rotation.
- **Image and Description Boundary Recognition:** There may be difficulties in accurately determining the boundaries of images and their descriptions. The GUI includes settings for adjusting the minimum height and width of images and descriptions.
- **Technical Limitations:** The program relies on external libraries supporting each stage of the algorithm, which may limit updating and scaling capabilities.

Interactions Between Libraries

- **PIL and pytesseract**: PIL is used for initial image processing, preparing the images for the OCR process. pytesseract then takes these processed images to extract textual data.

- **System-level Modules**: Modules like *os*, *sys*, and *copy* are used for file system operations and memory management, ensuring efficient handling of image and text data.
- **argparse and configuration**: The *argparse* module is utilized to interpret user commands and arguments, feeding into the configuration of the OCR process.

Data Flow Through Modules

The data flow through these modules is structured as follows:

1. Image loading and processing using PIL.
2. Text and non-text elements separation using pytesseract.
3. Handling of command-line inputs and configuration using argparse.
4. Storing and manipulating extracted text and image data using collections.namedtuple and other system-level modules.

3.4.5 Script Execution

The script accepts command-line arguments for the source filename, output directory, language of the text, and a boolean flag that indicates whether the document is in two columns. Create a configuration dictionary based on these arguments and pass it along with the source filename to an instance of the ImageExtractor class.

3.4.6 Validation of the Algorithm

To validate the efficiency of this algorithm, we can use a set of test cases consisting of images with known outcomes. The extracted results can then be compared with these expected results. The algorithm can also be evaluated based on its performance in edge cases, such as when images are given with different languages, sizes, colors, qualities, and document layouts.

To visually validate the intermediate results, we could modify the algorithm to save images representing the identified lines and columns of text. This would involve drawing bounding boxes around the identified text lines and columns based on their coordinates and saving these images in the output directory. This process could be performed in the `get_character_lines()` and `get_character_columns()` methods, respectively.

3.4.7 Strengths

- The algorithm can handle images in various formats and sizes.
- It is capable of recognizing text in multiple languages.
- Works well with documents formatted in one or two columns.
- It allows fine-tuning of its behavior through various configurable parameters.

3.4.8 Weaknesses

- The algorithm is highly dependent on the performance and accuracy of the Tesseract OCR engine.
- It may need to identify text lines and columns in complex document layouts correctly.
- The algorithm may not accurately distinguish between large text and small non-text elements.
- Its performance may be affected by the quality and clarity of the image.

Further validation techniques may include peer code review and stress testing with large or many images. Moreover, the algorithm can be improved by implementing more sophisticated text and non-text element detection methods, such as machine learning techniques or more advanced image processing libraries.

4. Program Development Journey

4.1 First Version: OpenCV and Textract Integration

4.1.1 Objective and Tasks

My first version of the script was developed to automate the process of extracting text from images and documents. The main motivation was to create an effective tool that could handle various file formats and extract text with high accuracy.

4.1.2 Technology Usage

- **OpenCV**: This powerful computer vision library was used for initial image processing. Tasks included loading images, converting them to grayscale to improve contrast, and preparing them for text recognition.
- **Textract**: I chose *textract* for text extraction because it provides a universal interface to various OCR engines and supports many file formats, making it an ideal solution for my tasks.

4.1.3 Development Process and Implementation

- Image Conversion and Saving: Images were first converted to grayscale, a common practice to enhance text analysis. They were then saved as temporary files for subsequent recognition.
- Text Recognition: Using *textract*, the script extracted text from the saved images. *Textract* automatically handled various aspects of OCR, simplifying the process.
- Output of Results: The recognized text was displayed in the console, and the processed images were shown for visual verification.

4.1.4 Challenges and Issues

- Accuracy of Recognition: Despite the use of *textract*, the accuracy of OCR sometimes fell short, especially with low-quality source images.
- Performance: The process of saving images as temporary files and their subsequent processing could be inefficient and slow down script execution.
- Configuration Complexity: Despite the versatility of *textract*, careful configuration was required for optimal performance with various formats and languages.

4.1.5 Conclusions and Lessons Learned

The first version of the script proved to be a valuable learning experience that highlighted the importance of choosing the right tools for specific tasks. The challenges I encountered prompted further research and experimentation, which ultimately led to significant improvements in subsequent versions of the script. The experience gained from working on the first version served as a catalyst for developing an intermediate stage, where new technologies and approaches were introduced, significantly improving the text recognition process and image management.

4.2 Intermediate Version: script improvement

4.2.1 Objectives and Goals

After the first version of my script, where I encountered several performance and accuracy issues, I decided to reconsider my development approach. The goal of the intermediate version was to enhance the user interface and integrate more powerful tools for text and image processing.

4.2.2 Technology Usage

- **Tkinter**: In this version, I introduced a graphical user interface using Tkinter, allowing users to more comfortably manage the text recognition process and select files through dialog windows.
- **Pytesseract and Textract**: I continued using pytesseract for OCR and textract to support various file formats. These tools provided the foundation for text extraction, but I hoped to improve their integration and efficiency.

4.2.3 Implementation and Functionality

- **Graphical Interface**: Users could select images for processing and save results through the graphical interface, significantly simplifying interaction with the program.
- **Image Processing**: The images were uploaded and processed using the PIL library to improve quality before transferring them to pytesseract. Preprocessing techniques such as contrast correction and cropping have been used to increase recognition accuracy.
- **Output of Results**: Recognition results were displayed in the interface and also saved in text files for further use.

4.2.4 Challenges and Issues

- **Recognition Accuracy**: Despite improvements, the accuracy of OCR remained below expectations, particularly in cases with complex layouts and low-quality images.

- **Performance**: Processing time remained an issue due to suboptimal preprocessing and the time taken to load large images.

4.2.5 Conclusions and Lessons

The intermediate version of the script demonstrated a significant improvement in the user interface and some improvement in text processing. However, problems with accuracy and performance indicated the need for further improvement of the technology stack and optimization of algorithms. These findings pushed me to the next stage of development, where I focused on closer integration with pytesseract and additional optimization to improve text recognition results and overall system performance.

4.3 Final Version of the Script: Completion of Development

4.3.1 Development Overview

The final version of the script is the result of numerous iterations and enhancements aimed at improving previous versions. The main focus in this version was on enhancing the accuracy of automatic text and image recognition and optimizing PDF document management to ensure increased performance.

4.3.2 Goals and Achievements

The goal of the final version was to create an effective and stable tool for extracting text and images from PDF documents without the need for additional parameter settings.

4.3.3 Features and Technologies

- **Enhanced Graphical User Interface (GUI)**: Implemented in Tkinter, the GUI allows users to configure the saving parameters of the results and edit them directly from the interface.
- **Advanced OCR Features**: The functionality of pytesseract has been significantly optimized to flawlessly identify areas with text and images, thus solving the problems of previous versions.
- **Automatic Processing**: The program can now process documents with minimal user intervention, providing quick access to results.

4.3.4 Implementation

The functionality of the program allows users to easily upload and process PDF documents, automatically identifying and processing text and graphic areas. The processing process can be customized and a location can be selected to save the results through a flexible user interface.

4.3.5 Conclusion and General Reflection

The final version demonstrates significant improvements in functionality and usability, providing an effective solution for recognizing and processing data from PDF documents.

4.3.6 Project Outcome and Lessons Learned

This project emphasizes the importance of an iterative approach in software development, where each version is aimed at improving and solving the problems of the previous one. The main goal—creating a stable working product—was successfully achieved, confirming the importance of persistence and innovation in the development process.

Conclusion

At the conclusion of this thesis, we aim to evaluate all the goals achieved and discuss future work that would complement and expand upon this program.

Achieved Goals

Comprehensive OCR System Development

- **Robust OCR Implementation:** This thesis accomplished a significant milestone by implementing a robust OCR system with Tesseract. The solution stands out for its enhanced ability to accurately recognize and process text from various printed materials, especially from older documents where text clarity might be compromised.
- **Theoretical Insights into Image Preprocessing:** The research deeply explored image preprocessing theories, establishing a solid understanding of their importance in enhancing OCR. This comprehensive theoretical examination illuminated various techniques, emphasizing their role in boosting text recognition from challenging sources. These insights indicate a roadmap for future OCR system improvements, focusing on performance and accuracy.
- **Innovative Text and Graphics Segregation:** The development of innovative methods to effectively distinguish text and graphical elements within documents marked a significant achievement. This advancement substantially improved extraction of accurate information from documents with intricate layouts.
- **Extensive Language Support:** The software's multilingual capabilities were expanded, making it a versatile tool for OCR applications across various languages and scripts, broadening its usability and applicability.
- **Technology Integration:** The research successfully explored the integration of OCR with cutting-edge technologies like machine learning and artificial intelligence, laying the groundwork for future advancements in this field.

Future Work

Evolving OCR Technologies

- **Advanced OCR Accuracy Enhancements:** Future work should focus on refining OCR accuracy, especially in processing handwritten texts and documents with complex layouts. Implementing more sophisticated algorithms and learning models could significantly boost recognition precision.

- **Deeper Machine Learning Integration:** Further integration of machine learning algorithms is essential. These would enhance text recognition capabilities and introduce predictive and adaptive features for various document types and languages.
- **Language and Script Expansion:** Expanding OCR capabilities to include a wider range of languages, particularly those with non-Latin scripts or complex orthographic systems, is a critical area for future development.
- **Preprocessing Technique Optimization:** Continuous improvement and optimization of image preprocessing techniques are necessary to ensure consistently high-quality outputs, regardless of the document's initial condition.
- **User Interface Enhancements:** Developing a more intuitive and feature-rich graphical user interface would significantly improve user experience, making the software more accessible to a broader audience.
- **Mobile Platform Adaptation:** Adapting the OCR software for mobile platforms would make this powerful tool accessible on the go, opening up many practical applications.
- **Real-time Processing Capabilities:** Implementing real-time text recognition could revolutionize how OCR is used in live scenarios, such as in augmented reality applications or instant translations.
- **Cross-technology Collaborations:** Future developments could explore collaborations with other digital technologies, such as blockchain for document verification or cloud computing for enhanced data processing capabilities.

Bibliography

[1] Wikipedia contributors. Portable document format — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/PDF, 2023. [Online; accessed 5-July-2023].

[2] Wiki2 contributors. Xfdf — Wiki2, the free encyclopedia. https://wiki2.org/en/XFDF, 2023. [Online; accessed 6-July-2023].

[3] Adobe PSIRT. Adobe security bulletin. https://helpx.adobe.com/security.html, 2022. [Online; accessed 5-July-2023].

[4] Jens Müller, Dominik Noss, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. Processing dangerous paths – on security and privacy of the portable document format. 2020. [Online; accessed 30-April-2024].

[5] Wondershare. What is ocr and how does ocr work. https://pdf.wondershare.com/ocr/what-is-ocr.html, 2023. [Online; accessed 5-July-2023].

[6] Wikipedia contributors. Optical character recognition — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Optical_Character_Recognition, 2023. [Online; accessed 6-July-2023].

[7] List Software Info. Optical character recognition. https://listsoftwareinfo.blogspot.com/2018/10/optical-character-recognition.html, 2018. [Online; accessed 6-July-2023].

[8] Dataconomy. What does ocr stand for? https://dataconomy.com/2023/05/what-does-ocr-stand-for/, May 2023. [Online; accessed 6-July-2023].

[9] Amazon. What is ocr? https://aws.amazon.com/what-is/ocr/, 2022. [Online; accessed 5-July-2023].

[10] Wikipedia contributors. Optical character recognition — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Optical_character_recognition, 2023. [Online; accessed 6-July-2023].

[11] Boudy Technology. What is machine learning (ml)? https://www.boudy-technology.tn/2022/04/WhatismachinelearningML.html, 2022. [Online; accessed 6-July-2023].

[12] Microsoft Azure. What is a machine learning platform? https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-machine-learning-platform/, 2023. [Online; accessed 6-July-2023].

[13] Wikipedia contributors. Machine learning — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Machine_learning, 2023. [Online; accessed 6-July-2023].

[14] Department of Aeronautical Engineering, Malla Reddy College of Engineering & Technology. Digital notes on business analytics basics. Technical report, Malla Reddy College of Engineering & Technology, Secunderabad – 500100, Telangana State, India, 2023. [Online; accessed 6-July-2023].

[15] Saikat Das, Sampita Mallick, Shyamapriya Chatterjee, and Sujata Kundu. Machine Learning in Big Data Analytics. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT)*, 09(11):01–05, 2021.

[16] Phil Heidenreich. What are the types of machine learning? `https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f`, 2018. [Online; accessed 6-July-2023].

[17] Shiteng Yang. Ds01-supervised-and-unsupervised-learning. `https://github.com/yangshiteng/DS01-Supervised-and-Unsupervised-Learning`, 2023. [Online; accessed 6-July-2023].

[18] IBM. What is machine learning. `https://www.ibm.com/hk-en/topics/machine-learning`, 2023. [Online; accessed 6-July-2023].

[19] Nanonets. Ocr software: The best ocr software. `https://www.nanonets.com/blog/ocr-software-best-ocr-software/`, 2023. [Online; accessed 6-July-2023].

[20] S. V. Yelisieieva. *Information Technologies in Translation*. PMBSNU Publishing House, 2018.

[21] Wikipedia contributors. Tesseract (software) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Tesseract_(software)`, 2023. [Online; accessed 5-July-2023].

[22] Ekaterina Obalyaeva. Time series prediction using reinforcement learning. Master's thesis, National Research University Higher School of Economics, 2019.

[23] Kirill Kolodiazhnyi. *Hands-On Machine Learning with C++*. Packt Publishing, 2020.

[24] Xiaoge Zhang. *Machine Learning and Optimization Models to Assess and Enhance System Resilience*. PhD thesis, Graduate School of Vanderbilt University, Nashville, Tennessee, May 31 2019.

[25] Masoud Daneshtalab and Mehdi Modarressi. *Hardware Architectures for Deep Learning*. The Institution of Engineering and Technology, 2020.

[26] Igor Kotenko, Igor Saenko, Oleg Lauta, and Mikhail Karpov. Methodology for management of the protection system of smart power supply networks in the context of cyberattacks. *Energies*, 14:5963, 2021.

[27] Neural nets in tesseract 4.00. https://tesseract-ocr.github.io/tessdoc/tess4/NeuralNetsInTesseract4.00.html, 2023. [Online; accessed 6-July-2023].

[28] Chris Olah. Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. [Online; accessed 5-July-2023].

List of Figures

List of Abbreviations

- **PDF:** Portable Document Format
- **OCR:** Optical Character Recognition
- **PDL:** Page Description Language
- **DDL:** Data Description Language
- **ISO:** International Organization for Standardization
- **LSTM:** Long Short-Term Memory
- **RNN:** Recurrent Neural Network
- **AI:** Artificial Intelligence
- **NLP:** Natural Language Processing
- **ML:** Machine Learning
- **PIL:** Python Imaging Library
- **ROI:** Region of Interest
- **GUI:** Graphical User Interface
- **PPP:** Paycheck Protection Program
- **API:** Application Programming Interface
- **SAP:** Systems, Applications, and Products in Data Processing

A. Documentation

A.1 User documentation

A.1.1 Downloading the development environment

- Download the current version for your operating system from `https://www.python.org/downloads/`
- Install the Python program thanks to the previously downloaded file.

A.2 Downloading and installing the program

- Download the current version of PureOCR from `https://gitlab.mff.cuni.cz/teaching/nprg045/ikudov/saydametov-nikita/-/releases`
- After downloading, you will have a zip file, which you need to unzip to a folder convenient for you
- Go to the folder PureOCR where the file **requirements.txt** is located
- From here, we launch the console and write the command

  ```
  pip install -r requirements.txt
  ```

- After writing this command, all the libraries you need for work will be loaded on your computer
- The program is installed and ready to use!

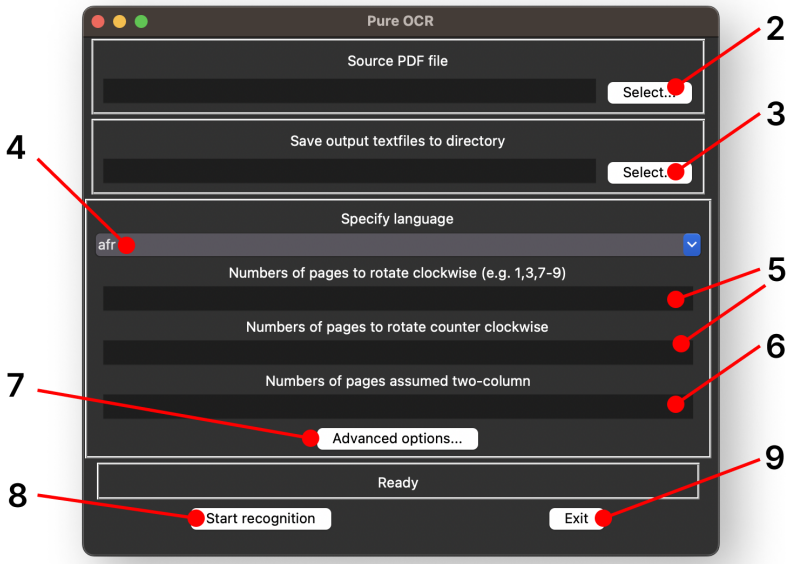
A.2.1 Starting and using the program

1. Run the program using the console command

   ```
   python3 ocr_gui.py
   ```

2. Then select PDF file using **Select...** button in the **Source PDF file** window
3. After selecting the PDF file, select the directory where the recognized files will be saved. To do this, click **Select ...** in the **Save output text files to directory** window
4. Next, select the language used in the recognized PDF file. This is done using the drop-down list in the **Specify language** window. You can only select languages from those you have installed for PyTesseract.
5. Next, select the page numbers you want to rotate clockwise or counterclockwise. You can do this using the appropriate **Numbers of pages to rotate** windows

Figure A.1: User interface



6. In addition, if the recognized PDF file contains pages consisting of several columns, specify the numbers of those pages using the **Numbers of pages assumed two-column** window.

7. You can also adjust the minimum and maximum sizes of images and letters in the advanced settings (but as a rule, the default values are suitable in 90% of cases)

8. To start the recognition process, click the **Start recognition** button

9. When the recognition process is over, you can exit the program using the **Exit** button

10. All the files processed by the program will be located in the folder you specified in step 2