POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
BACHELOR OF SCIENCE IN COMPUTER SCIENCE

DESIGN AND ANALYSIS OF ALGORITHMS

SaLitrato: Application of Boyer-Moore String Matching Algorithm for Text Search on Multiple Image Files

Bandala, Charles Andre
Batisla-Ong, John Franky Nathaniel
Hernandez, Vince Ivan
Jamero, Emmanuel
Lopez, Toni Gaile
Pantino, Blessy Ann
Pabular, Steve
Taripe, Leejani

Table of Contents

Part I
Introduction and its Background

In today's digital age, one of the most useful functions of using an app on a computer is its capability of allowing an individual to find words or phrases quickly in a full text, also known as the full-text search feature within documents. One popular function that uses the concept of full-text search is the keyboard shortcut "ctrl+f". People often use this to find a specific phrase or characters in a long document or programs such as internet browsers, word processors, PowerPoint, etc.

Full-text search is a method to search and locate a text pattern through the actual content of your computer documents (Full Text Search | FileCenter DMS, 2023; Stasilovich, 2018). However, this feature is accompanied by a significant limitation. In this technique, the documents must have actual text and not just scanned images. Therefore, if an individual cannot select and copy a text from the document, the text cannot be searched. This also means that a scanned document, scanned image, or any type of photo is not searchable even if it has text in it.

Since images also contain valuable information and play a growing significant role in terms of communication, information sharing, and data representation, the inability to extract text from images in a full-text search feature makes it more difficult to access valuable information efficiently. To bridge the gap between text and images, Optical Character Recognition (OCR) technology was developed by converting all possible text from an image to become machine-readable. According to Kudan (2023), text recognition technology may also involve translating, processing text discovered in images, and turning printed documents into editable formats. It may even read text aloud to those who are blind. Jobs in industries including business, education, healthcare, and government may be done much more efficiently by using text recognition to automate manual activities, boost productivity, and provide access to previously inaccessible information. The trend of storing information from printed materials in digital format has greatly increased in recent years. This makes it easier to store information and access it when needed, as well as helping to preserve the information. The field of text recognition is also driven by the need for a more convenient and quick way of preserving and accessing documents that contain information. One of the easiest ways to transfer information from paper or books is to scan them. This turns the information into an image and prevents the text from being reproduced (Adyanthaya, 2020).

The feature that uses OCR technology to extract text from images and lets users search specific words, characters, or phrases within the extracted text from images is called the **in-image text searching feature**. The main distinction between the full-text search and in-image text search lies in the source of the text being searched. In-image text search searches for text specifically from images while full-text search only recognizes text from text-based content such as documents or articles. There are numerous valuable applications of **in-image text searching** across various industries. One instance is it enables an efficient method for converting physical documents into digital and searchable formats. An individual only needs to scan physical documents such as books, contracts, and receipts and upload them to software capable of in-image text searching. This method

allows faster retrieval of information and reduces the need for manually typing the content of the physical documents on a computer. The feature can also be useful in content moderation. Social media applications and platforms can use this feature to scan images for text content, then process the extracted text to detect any harmful or inappropriate text that violates their platform's policies. Furthermore, in-image text search is also notable for educational and learning purposes; students, teachers, and researchers can use the feature to take notes and search specific terms efficiently.

Various software with OCR and the capability to extract text within images already exist. However, there are still fewer studies online about applications that provide users with an added feature of searchable text within the extracted content from the images. The full-text search feature is still the most common one to be used in applications. Based on the resources and studies the researchers have gathered online, the text-searching feature is more implemented in text-based content over images. Furthermore, studies online often used the Boyer-Moore algorithm to implement their search function. For instance, Waruwu (2017), Buslim et al. (2020), Khumaidi et al. (2020), and Layustira and Istiono (2021) did use the Boyer-Moore algorithm but did not implement the algorithm for an in-image text search feature. These studies did not allow users to search for text patterns within the content of images for their application software. Additionally, in the study of Fitriyah et al. (2020), the Boyer-Moore algorithm was implemented in a web-based computer and informatic terms dictionary. In the study of Arini and Suseno (2019), Boyer-Moore Algorithm was applied to a dictionary of web-based information technology terms. Furthermore, the algorithm was also used in the application of dictionary livestock terms (Saputri et al., 2018). Based on the mentioned studies that use the Boyer-Moore algorithm for text-search features, the algorithm, and the feature is often used in dictionaries and other text-based content or forms.

As evident in the studies and resources online, Boyer-Moore Algorithm is commonly used for search functions in text-based content, such as dictionaries and documents. Thus, there is a lack of studies and resources online about applications that offer in-image text searching while also utilizing the Boyer-Moore algorithm as its string-matching algorithm. To address the aforementioned issue, the study will focus on developing an application software capable of **folder-wide detecting and locating a text pattern within multiple image files**, enabling users to efficiently locate information present in images.

In terms of implementing a text search feature into application software, it is typical to utilize a string-matching algorithm, such as the Boyer-Moore Algorithm, KMP Algorithm, Naïve Algorithm, and Brute Force Algorithm. **String-matching algorithms** are algorithms that find occurrences or matches of a pattern within a text or content of multiple files (GeeksforGeeks, 2022). Since numerous string-matching algorithms can be used, selecting one depends on the nature and requirements of the software. As such, the researchers will utilize the **Boyer-Moore algorithm** to implement the in-image text search feature on the software.

The key feature of the Boyer-Moore Algorithm is its efficiency in pattern searching within huge bodies of text. As the number of images uploaded in the in-image text search function increases, the larger the block of text will be extracted. Thus, the number of

texts to be traversed by the algorithm increases too. Boyer-Moore Algorithm is well-suited for this scenario as it works best with huge bodies of text. Furthermore, according to several studies online, the **Boyer-Moore Algorithm** is the more efficient algorithm compared to several other string-matching algorithms. (Khumaidi et al., 2020; Layustira & Istiono, 2021; Rasool et al., 2012; Supatmi & Sumitra, 2019; Dawood & Barakat, 2020; Lin & Soe, 2020; Buslim et al., 2020; Borah et al., 2013). Additionally, numerous studies online that implemented search functions using the Boyer-Moore algorithm have also proven the efficiency of the algorithm (Fitriyah et al., 2020; Arini & Suseno, 2019; Saputri et al., 2018).

Since the algorithm for the study has been decided, other features will be implemented into the software. The application will be integrated with an optical character recognition (OCR) API to recognize and extract texts from images. The software will also implement a function that highlights the location of the text pattern in the extracted text or shows a message if the text pattern is not detected.

Part II
Review of Related Literature

This segment of the project documentation provides an overview and discusses the related literature relevant to the topic of the project in which is the development of an image-based text searching application. The goal of this section is to provide a foundation for the research that will be presented in the rest of the document.

Full Text Search

A full text search is a query that looks for specific words within a large body of electronically stored text data and returns results that include all or some of those words (Full-Text Search: What Is It and How It Works | MongoDB, 2023). This is a way of searching the body of a document in much the same way that you search the web. It also requires that your documents contain actual text to index the files (Full Text Search, 2023). According to Stasilovich (2018), full text search has a variety of uses including searching for a word or phrase in a document, searching in a web page on the internet, and searching for a name in a list. Moreover, this can be implemented by means of indexed search or by utilizing string searching algorithms like Rabin-Karp algorithm, Knuth-Morris-Pratt algorithm, simple text searching, and Boyer-Moore algorithm. Furthermore, applications like Joplin, Notepad++, Google Search, etc. contain the full text search feature (Apps with "Full-Text Search" Feature, 2023). Additionally, applications such as SeekFast and DocFetcher are some of the applications based mainly on full text search (Tran Nam Quang, 2022; SeekFast, 2023).

Digital Libraries

Access to a huge selection of digital materials, such as e-books, e-journals, digital photographs, videos, and other resources, is now made possible through the development of digital libraries. Digital libraries are websites committed to establishing and maintaining collections of electronic books and other types of content without requiring users to pay for the items they wish to go through and read (Cordón-García, et al., 2013). They offer equal access to knowledge by making it possible for consumers to access a great amount of information via internet-connected devices at any time and from any location.

By preserving fragile or disintegrating items, digital libraries play an important part in maintaining and conserving cultural heritage, historical documents, and rare materials. They promote collaboration, citation monitoring, and data analysis while giving researchers and students access to scholarly articles, research papers, books, and other academic materials. In addition to providing access to instructional materials, works in the public domain, and user-generated content, digital libraries also extend their services to the public to promote participation and community engagement. Additionally, they offer specific features that improve user experience and speed up the process of learning, such as suggestions, bookmarking, annotation, and collaboration tools.

Image Processing API

According to Anbarjafari (2014), image processing is a process for altering an image to produce a better image or to extract some relevant information from it. It is a kind of signal processing where the input is an image, and the output can either be another image or features or characteristics related to that image. It involves performing processes on images to improve their quality, extract useful details, or enhance the way they appear. A variety of images, including photographs, medical images, satellite imagery, and more, can be processed. Image processing is a standard, modern method for manipulating photographs to obtain important information that isn't visible in conventional images. Image processing is a popular modern strategy for modifying photographs to acquire essential knowledge that isn't obvious in regular images.

The API enhances photos by adding detail using a variety of super-resolution methods (Abid, 2022). The API has the capacity to improve the image and scale the resolution by around four times. Modern algorithms in the API are to thank for all of this. The API can boost image quality before printing, do automatic image optimization for products, and boost social media image resolution. Developers can add image-processing features to programs by using an image-processing API. Applications may include features like image classification, OCR, image conversion, image compression, computer vision, image editing, and more using image processing APIs.

Pattern Searching Algorithm and Pattern Recognition

According to Sam (2019), algorithms for pattern searching are used to extract a pattern or substring from a larger string. There are various algorithms that exist. This kind of algorithm design's primary objective is to decrease time complexity. When searching for patterns in a larger text, the standard approach may take a long time. The Pattern Searching algorithms were included in the String algorithms and can also be referred to as String Searching Algorithms. These techniques are helpful when searching for a string within another string.

Pattern recognition is the ability to detect arrangements of characteristics or data that yield information about a given system or data set (Wigmore, 2018). In a technological context, a pattern could be, among many other possibilities, recurrent patterns of data over time that can be utilized for anticipating patterns, particular layouts of features in images that identify objects, regular word, and phrase combinations for natural language processing (NLP), or patterns of behavior on a system that could indicate an attack. Additionally, they can classify and identify unusual parts, identify patterns and objects that are partially hidden from view and distinguish forms and objects from various perspectives.

According to Ansari (2023), The definition of pattern recognition is the classification of data based on previously acquired knowledge or on statistical data derived from patterns and/or their representation. The potential for pattern recognition applications is one of its key features.

String-Matching Algorithms

According to GeeksforGeeks (2022), there are several known exact string-matching algorithms to use for comparing characters. The algorithms based on character comparison are Naïve, Knuth-Morris-Pratt (KMP), Boyer-Moore, and using the Trie data structure. In terms of the study being conducted, Boyer-Moore Algorithm (Fig 1) will be used for detecting and locating a text pattern within an image file.

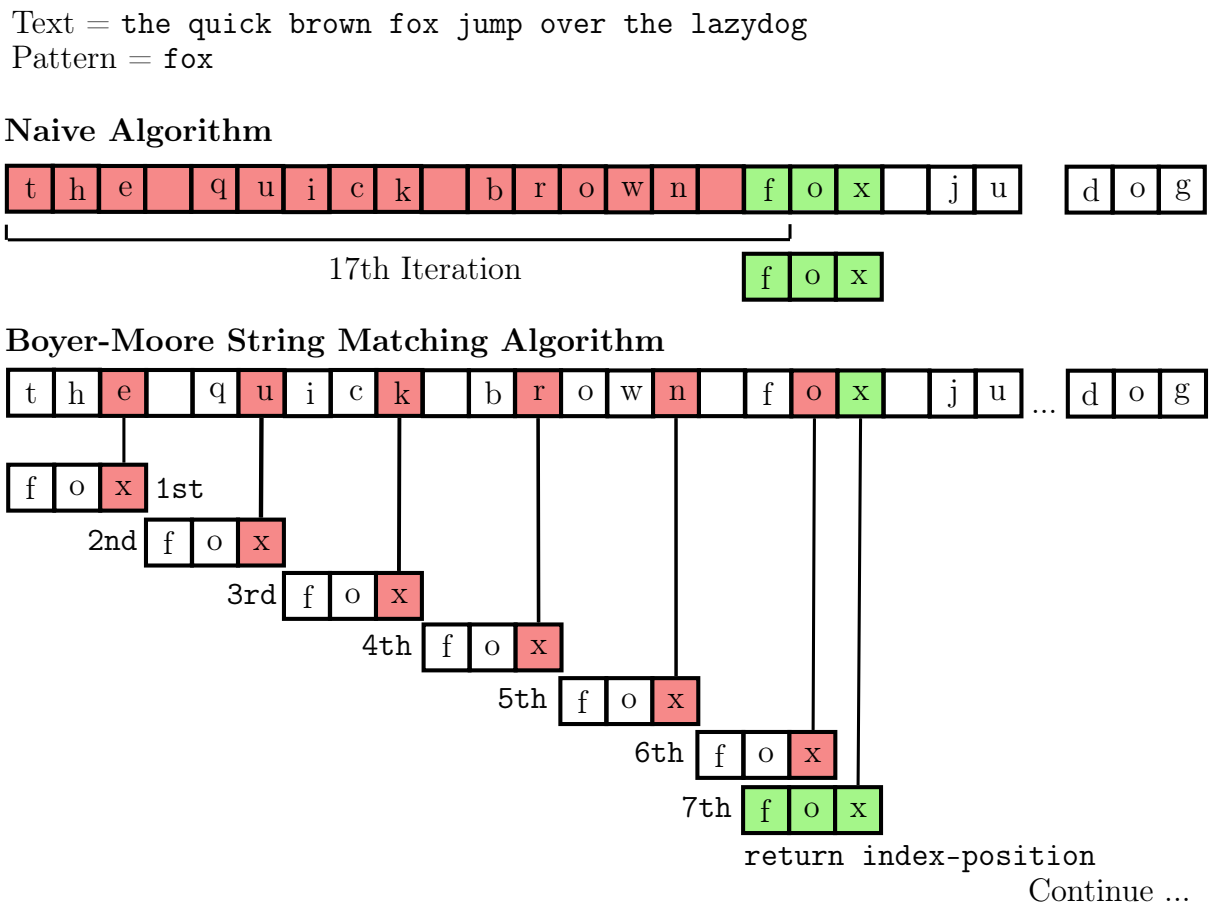

Figure 1: Comparison between Naive and Boyer-Moore Algorithm

Boyer-Moore String-Matching Algorithm

According to Waruwu (2017), the Boyer-Moore (BM) algorithm is one of the well-known string-matching algorithms due to its performance in terms of matching single patterns. The algorithm compares each character in the pattern to the haystack from right to left, but the window shift remains from left to right. In addition to that, the aforementioned algorithm was proven as one of the most efficient in terms of string search applications using natural language in which it has been implemented for the functions "Search" and "Substitute" of various text editors. Furthermore, the algorithm was utilized in the multiple studies as the search function. In the study of Fitriyah et al. (2020), the boyer-moore algorithm was implemented in a web-based computer and informatic terms dictionary. It was also used in the study of Arini and Suseno (2019) where it was applied

in a dictionary of web-based information technology terms. Lastly, the boyer-moore algorithm was used in the application of dictionary livestock terms (Saputri et al., 2018).

Comparison of Boyer-Moore Algorithm to Other String-Matching Algorithms

The reason for utilizing Boyer-Moore (BM) Algorithm in the study's application software is that several studies have demonstrated that BM Algorithm is much more efficient compared to other several string-matching algorithms such as Brute Force, KMP, Bozorth, and Rabin and Karp in most cases (Khumaidi et al., 2020; Layustira & Istiono, 2021; Rasool et al., 2012; Supatmi & Sumitra, 2019; Dawood & Barakat, 2020; Lin & Soe, 2020; Buslim et al., 2020; Borah et al., 2013). According to Dawood and Barakat (2020), the overall performance evaluation of BM and KMP algorithms showed that the BM algorithm outmatched the KMP algorithm in all test scenarios. Furthermore, the 1-4 test scenarios proved that the BM algorithm is around three times faster than the KMP algorithm. The pattern size also has no significant effect on the performance of both algorithms.

Another study for a web-based dictionary of computer terms also concluded that BM Algorithm has a better performance compared to the KMP algorithm. It used the Exponential Comparison Method (ECM) to determine which is the faster and more efficient algorithm between BM and KMP. The total ECM score shows that the BM algorithm is 0.55% more efficient than the KMP algorithm (Khumaidi et al., 2020)

In terms of comparison of the Boyer-Moore Algorithm to Brute-Force (BF) String-Matching algorithm, the BM algorithm's performance is better and much more efficient compared to BF Algorithm. According to Lin and Soe (2020), the study results based on the number of shifts, comparisons, and runtime concluded that the BM algorithm has better performance in comparison to the BF Algorithm. Borah et al. (2013) also concluded that the BM algorithm is faster and more effective than the BF algorithm. Additionally, when it comes to word suggestion search, the BM algorithm is 79.05% more time-efficient than the BF algorithm (Layustira & Istiono, 2021). Furthermore, Layustira and Istiono (2021) stated that the BF algorithm is preferable for string matching with shorter keywords and strings; while the BM algorithm is suited for all types and conditions of both strings and keywords being searched.

According to Buslim et al. (2020), the BM algorithm's search process is more efficient than BF Algorithm, KMP Algorithm, and Rabin and Karp algorithm. This is evident in the results of the study wherein Boyer-Moore has a search speed of 0 seconds on a total of 50-word search.

Part III
Objectives

The primary endeavor of this project is to explore and analyze the following objectives:

General Objective

The main purpose of this project is to develop an application software capable of folderwide detecting and locating of a text pattern within multiple image files, enabling users to efficiently find important information present in image(s).

Specific Objectives

- To develop an application integrated with an optical character recognition (OCR) API for extracting texts from images for pattern finding.
- To implement the Boyer-Moore algorithm as the string-matching algorithm for the efficient searching of text patterns.
- To implement a function that provides the user with the location of the text pattern in the extracted text of the OCR by means of highlighting the specific location of the text or displaying a prompt if the text pattern is not found.
- To test the application software on various image files and text patterns and evaluate its performance in terms of its string-matching algorithm.

Part IV
Scope and Limitations

The coverage of this project is focused on developing a text-searching application made particularly for multiple images, more preferably screenshots and clear scanned documents. The Boyer-Moore algorithm will be the string-matching algorithm used in this project due to its efficiency as stated by recent studies. Moreover, the optical character recognition API used in this project will be a version of Tesseract 4. Lastly, the project will be developed as a desktop application.

It is outside the scope of the project to support text searching for other file types such as documents, PowerPoints, PDFs, etc. since the project focuses on image files. In addition to that, the development of an original optical character recognition software for the application is also outside the scope of the project. On top of that, the accuracy of the integrated optical character recognition API will be outside the scope of the project. Furthermore, the project will not provide compatibility with mobile devices.

Part V
System Design and Methodology

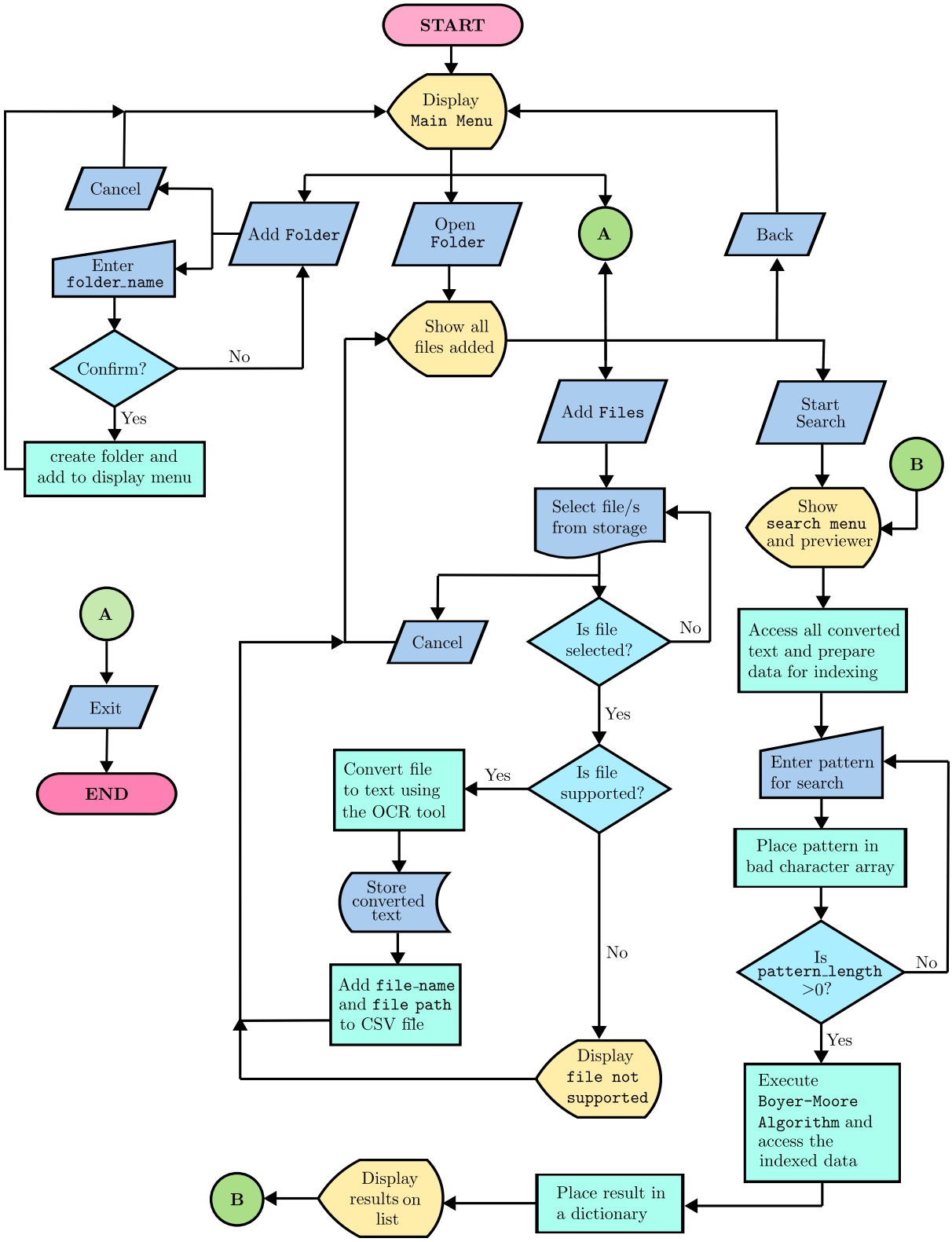

Figure 2: SaLitrato process flowchart

Design Goal Overview

The figure (Fig 2) above illustrates how the system software that implements an in-image text search method will search for and extract text within an image or multiple image files using Boyer-Moore algorithm. A text pattern within an image file will be detected and located using the Boyer-Moore algorithm. The system will additionally incorporate the use of optical character recognition (OCR) API with the version of Tesseract 4 to identify and extract texts from images.

The user will create and select a folder to which the pre-processed file would be stored, the aforementioned file would then be checked if it is one of the supported formats. It will then be subjected to the Python Tesseract package where it will undergo a connected component analysis that gathers and stores the outlines. The outlines will then be further analyzed in a fixed pitch or if they are proportional texts. They will then be separated into words according to character spacing by using character cells.

Lastly, it will undertake a two-pass procedure using an adaptive classifier to enhance the accuracy of the recognition process. Only then will it be put through the Boyer-Moore Algorithm where the user-inputted characters will be compared to the text from the post-processed file. To be able to assist the users in easily identifying the result, the software also has a feature that highlights the location of the text pattern inside an image file or displays a prompt if the pattern is not recognized.

System Architecture

This software is built using a modular, object-oriented approach with focus on readability, maintainability, and extensibility. It follows the traditional `layered architecture`, with several layers managing the user interface, application logic, and system interactions.

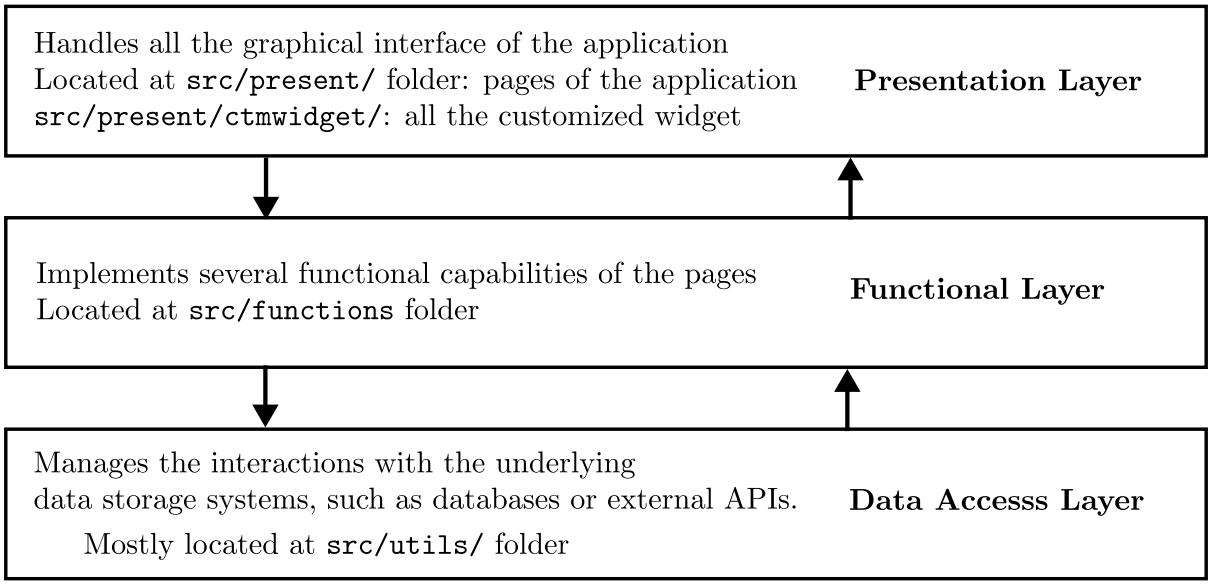

Figure 3: System architecture of SaLitrato

Data Storage Design

On initial start of the application, the user is met with the selection of add, open, and remove folder. Every folder created is stored in the current working directory (`$CWD`) of where the application was executed. This specifically appears on the `data/` folder which can be located at `$CWD/data/<folder_name>`. Inside `$CWD/data/<folder_name>`, is where the extracted text from an image will be stored. The CSV file is also generated here replicating the name of its current folder (`<folder_name>.csv`). This acts as a database of current selected image files to be run on by the Boyer-Moore Algorithm.

file_key	file_path	file_name
1	~/Projects/filename1.png	filename1.png
2	~/Photos/filename2.jpeg	filename2.jpeg
3	~/Downloads/images/filename3.jpg	filename3.jpg
4	~/Projects/filename5.webp	filename5.webp

Image to Text Conversion

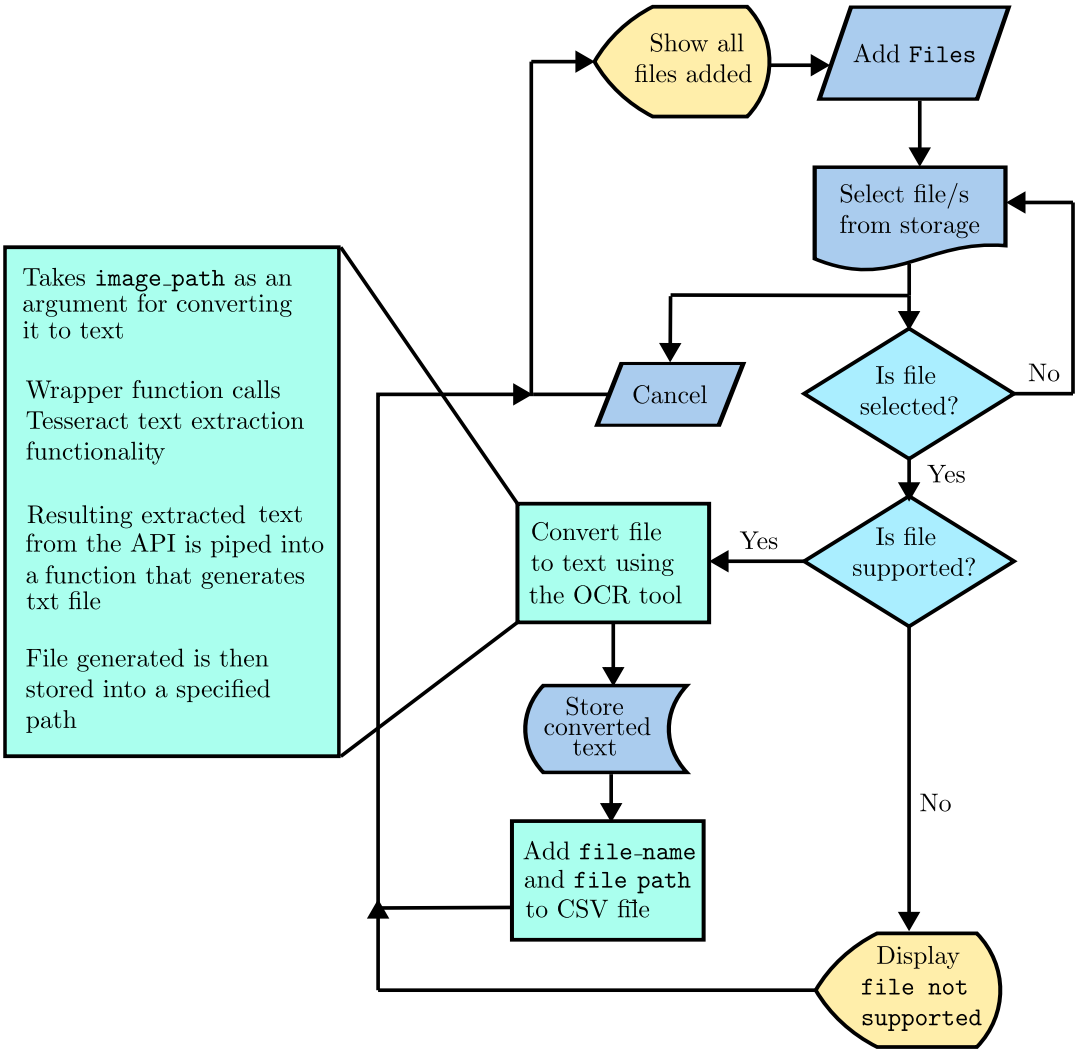

Figure 4: Image to text extraction with Tesseract 4

SaLitrato is dependent on the stable version of Tesseract 4 for extracting text from image files. Tesseract is an OCR Engine focused on line and character recognition. With Python, the text extraction API can be accessed via importing `image_to_string` on `pytesseract` package.

`pytesseract` is a wrapper class that allows interaction with the Tesseract engine.

Data Structure Preparation for Indexing

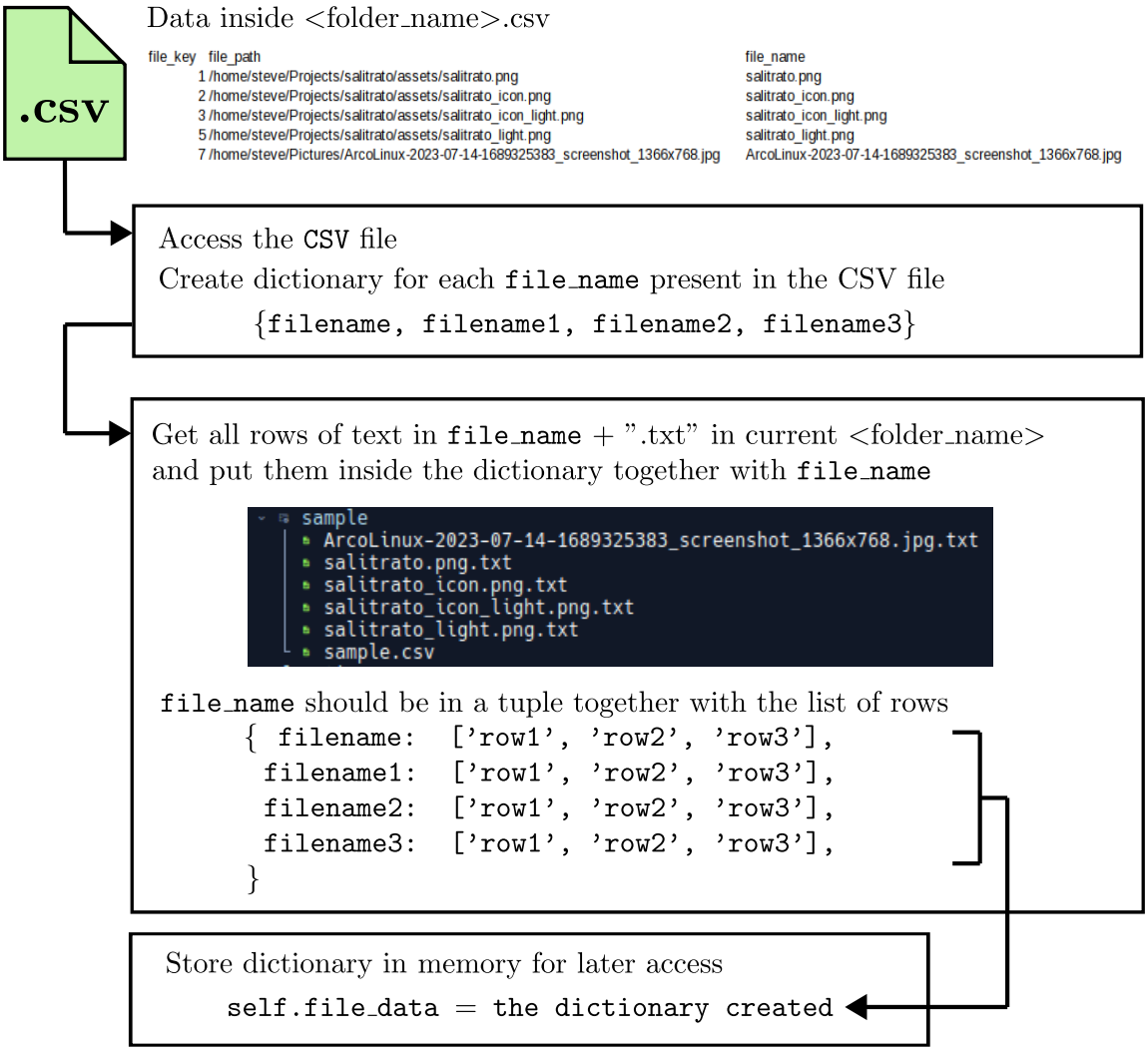

Figure 5: Structure of data for processing

In figure 5 shows how the data will be structured. This prepares the text of each file to undergo what we call as indexing. Indexing refers to identification of the location of a resource based on file names or key data fields in a record. Traversing through a record like this (Fig 6) would make tracking the index of where the current pattern appeared become easier.

{'2023-07-17_15-33.png': ['“The figure above illustrates how the system software that implemens
 an in-image text', 'search method will search for and extract text within an image or multiple
 image fils using', 'Boyer-Moore algorithm. A text pattern within an image file will be detecte
d and located using', 'the Boyer-Moore algorithm. The system will additionally incorporate the
use of optical', 'character recognition (OCR) API with the version of Tesseract 4 to idenify an
d extract texts', 'from images.', ''], '2023-07-17_15-34.png': ['Lastly it will undertake a two
-pass procedure using an adaptive classifier to enhance', 'the aceuracy of the recognition proc
ess. Only then wil it be put through the Boyer-Moore', 'Algorithm where the user-inputied chara
cters will be compared to the text from the', 'post-processed file. To be able to assist the us
ers easily identify the result, the software also', 'has a feature that highlighs the location
of the text pattern inside an image file or displays a', 'prompt ifthe patten is not recognized
.'], '2023-07-17_15-34_1.png': ['The reason for utilizing Boyer-Moore (BM) Algorithm in the stu
dy's application software is', 'that several studies have demonstrated that BM Algorithm is muc
h more efficient compared to other', 'several string-matching algorithms such as Brute Force, K
MP, Bozorth, and Rabin and Karp in most', 'cases (Khumaidi et al., 2020; Layustira & Istiono, 2
021; Rasool et al., 2012; Supatmi & Sumitra,', '2019; Dawood & Barakat, 2020 Lin & Soe, 2020; B
uslim et al., 2020; Borah et al., 2013). According', 'to Dawood and Barakat (2020, the overall
performance evaluation of BM and KMP algorithms', 'showed that the BM algorithm outmatched the

Figure 6: Sample data

Algorithm Design

Having multiple lists of text to traverse through requires a fast and efficient algorithm for pattern matching. There are multiple string matching algorithms that exist like Naive Algorithm, Z Algorithm, Knuth Morris Pratt, and Boyer-Moore algorithm. Observing that as the number of image files being included for search increases, the number of text for an algorithm to pass also increased drastically. Among said string matching algorithms, **Boyer Moore string matching algorithm** works best in this kind of scenario. Though at worst case it runs on $O(nm)$ where n is the length of text and m is the length of pattern, it is outweighed by its best case of $O(n/m)$ assuming there are a limited number of matches that occured which happens to be the case on having multiple texts to traverse.

The **Boyer-Moore Algorithm** in this program uses bad character heuristics to determine where the mismatch appeared. Multiple references exist for getting the code of this algorithm, which all comes with several drawbacks. We take the example of what was written in GeeksForGeeks [21].

Bad Character Preprocessing

```
1 NO_OF_CHARS = 256
2
3 def badCharHeuristic(string, size):
4     badChar = [-1]*NO_OF_CHARS
5     for i in range(size):
6         badChar[ord(string[i])] = i;
7     # return bad_char_array
8
9 def search(txt, pat):
10    badChar = badCharHeuristic(pat, m)
11    # return indexes of where the patterns appeared
```

When this type of code is executed, multiple instances of `badChar` will be recreated for every row there is to search for a match. This ramps up the complexity by increasing orders of magnitude. The `badCharHeuristic()` function itself can get $O(size)$ as its complexity. Without refactoring the code, it could reach a complexity of $O(size * rows)$ given the data is now being traversed. It can be prevented by decoupling `badCharHeuristic()` inside the `search()`. Making the search contain `badChar` only as an argument.

Another problem for the code shown above is the `NO_OF_CHARS = 256`. Giving a limit to values that can be inserted into the bad character array, would most likely generate an `Out of Bounds Exception` or `IndexError`. The said error is raised when an index that is out of range or invalid for a given data structure is being accessed. Making this array a hash table is one of the solution. Instead of being allocated manually and limiting the extension of bad character array, it becomes dynamically allocated on its own.

Modified Bad Character Preprocessing

```
1 def badCharHeuristic(pat):
2     table = {}
3     for i in range(len(pat)):
4         table[pat[i]] = i
5     return table
```

Sample input: `and`
Output: `{'a': 0, 'n': 1, 'd': 2}`

Retrieval of Matched Patterns

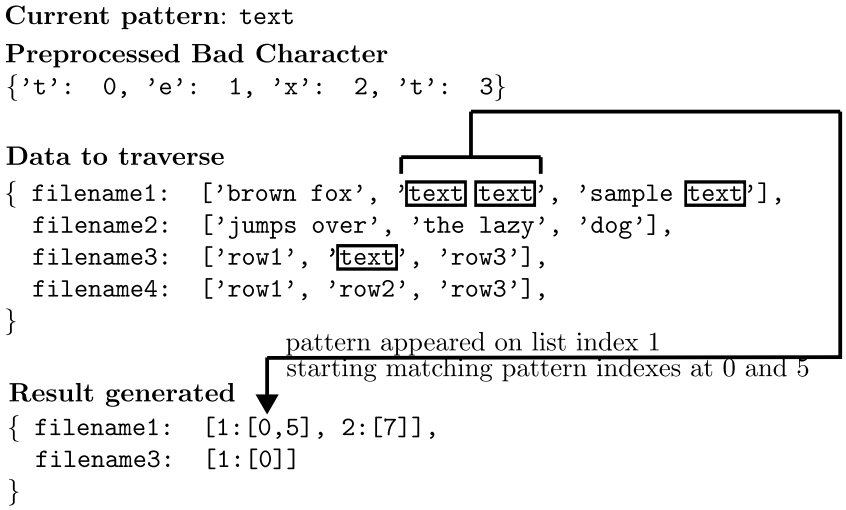

Figure 7: Generation of result

Getting the result of matches occured requires the algorithm to traverse on multiple list of rows on each file while retrieving the current index of where each pattern started (Fig 7). It accesses the data prepared for indexing which is stored in a dictionary or what we also call as hash table. This hash table with key as file name and list of rows as value creates a versatile way of identifing where we could find the result of the matching patterns.

Results generated is then used for identifying what position should be highlighted on the Textbox. Highlight for Textbox require the coordinates `begin=(row, column)` and `end=(row, column + pattern.len)`.

References

[1] Abid, E. B. (2022). *What Is An API In Image Processing?*. APILayer Blog. https://blog.apilayer.com/what-is-an-api-in-image-processing/

[2] Adyanthaya, S. K. (2020). *Text Recognition from Images: A Study. International Journal of Engineering Research & Technology, 8(13)*. https://www.ijert.org/text-recognition-from-images-a-study

[3] Anbarjafari, G. (2014). *Digital Image Processing*. University of Tartu. https://sisu.ut.ee/imageprocessing/book/1

[4] Ansari. (2023). *Pattern Recognition | Introduction*. GeeksforGeeks. https://www.geeksforgeeks.org/pattern-recognition-introduction/

[5] *Apps with "Full-Text Search" feature*. (2023). AlternativeTo. https://alternativeto.net/feature/full-text-search/

[6] Borah, P. P., Talukdar, G., & Jayanta, Y. (2013). *A comparison of String matching algorithms-Boyer-Moore algorithm and Brute-Force algorithm*. ResearchGate. https://www.researchgate.net/publication/265251889_A_comparison_of_String_matching_algorithms-Boyer-Moore_algorithm_and_Brute-Force_algorithm

[7] Buslim, N., Faradita, D. T., Anif, S., & Masruroh, S. U. (2020). *Mobile Based Application "Hadith for Women" Using Algorithm Boyer Moore*. https://doi.org/10.4108/eai.7-11-2019.2294564

[8] Cordón-García, J., Alonso-Arévalo, J., Gómez-Díaz, R., & Linder, D. F. (2013). *Open access eBooks*. In Elsevier eBooks (pp. 121–141). https://doi.org/10.1016/b978-1-84334-726-2.50004-5

[9] Dawood, S. S., & Barakat, S. A. (2020, May 19). *Empirical Performance Evaluation Of Knuth Morris Pratt And Boyer Moore String Matching Algorithms*. https://journal.uod.ac/index.php/uodjournal/article/view/732/560

[10] *Full Text Search | FileCenter DMS*. (2023). Filecenter. https://www.filecenter.com/info-text-search

[11] Stasilovich, S. (2018) Issart.com. https://blog.issart.com/full-text-search-how-it-works/

[12] *Full-Text Search: What Is It And How It Works | MongoDB*. (2023). MongoDB. https://www.mongodb.com/basics/full-text-search

[13] GeeksforGeeks. (2022). *Applications of String Matching Algorithms*. GeeksforGeeks. https://www.geeksforgeeks.org/applications-of-string-matching-algorithms/

[14] How does text recognition work. (n.d.). *How Does Text Recognition Work*. https://toloka.ai/blog/how-does-text-recognition-work/

[15] Khumaidi, A., Ronisah, Y. A., & Putro, H. P. (2020). *Comparison of Knuth Morris Pratt and Boyer Moore algorithms for a web-based dictionary of computer terms.* Jurnal Informatika, 14(1), 7. https://doi.org/10.26555/jifo.v14i1.a17038

[16] Layustira, V. A., & Istiono, W. (2021). *Comparative Analysis of Brute Force and Boyer Moore Algorithms in Word Suggestion Search.* International Journal of Emerging Trends in Engineering Research, 9(8), 1064–1068. https://doi.org/10.30534/ijeter/2021/05982021

[17] Lin, L. L., & Soe, M. T. (2020). *A Comparative Study on Brute-Force String Matching and Boyer- Moore String Matching.* Journal of Science and Technology for Sustainable Development, 1. https://www.uit.edu.mm/storage/2021/01/1-LLL.pdf

[18] Rasool, A., Tiwari, A., Singla, G., & Khare, N. (2012). *String Matching Methodologies: A Comparative Analysis.* International Journal of Computer Science and Information Technologies, 3. https://www.researchgate.net/publication/268273984_String_Matching_MethodologiesA_Comparative_Analysis

[19] Sam, S. (2019). *Introduction to Pattern Searching Algorithms.* Tutorials Point. https://www.tutorialspoint.com/introduction-to-pattern-searching-algorithms

[20] Supatmi, S., & Sumitra, I. D. (2019). *Fingerprint Identification using Bozorth and Boyer-Moore Algorithm.* IOP Conference Series. https://doi.org/10.1088/1757-899x/662/2/022040

[21] Waruwu, F. T. (2017). *Application Of Boyer Moore Algorithm for Text Searching.* International Journal of Informatics and Computer Science, 1(1). https://www.semanticscholar.org/paper/Application-Of-Boyer-Moore-Algorithm-for-Text-Waruwu/

[22] Wigmore, I. (2018). *pattern recognition.* WhatIs.com. https://www.techtarget.com/whatis/definition/pattern-recognition

[23] Arini, A., & Suseno, H. B. (2019). *Application of the boyer moore method in the application dictionary of web-based information technology terms (case study: pt. erefka tiga pilar utama).* INTEGRATED (Journal of Information Technology and Vocational Education), 1(2), 29–36. https://ejournal.upi.edu/index.php/integrated/article/view/21995

[24] Fitriyah, F., Gayo, W. R. P., Handayani, A. N., Wibawa, A. P., & Kurniawan, F. (2020). *The Implementation of Boyer-Moore Algorithm in WEB Based Computer and Informatic Terms Dictionary.* https://doi.org/10.1109/icovet50258.2020.9230344

[25] Saputri, W. S., Dhenabayu, R., & Febrinita, F. (2018). *APPLICATION OF STRING MATCHING USING BOYER MOORE ALGORITHM IN THE APPLICATION OF DICTIONARY LIVESTOCK TERMS.* https://doi.org/10.35457/josar.v1i01.614

[26] GeeksforGeeks. (2022). *Boyer Moore Algorithm for Pattern Searching.* GeeksforGeeks. https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/?ref=lbp