Application of Boyer-Moore Algorithm for Cancer, Tumor, and Aneurysm Detection in Brain CT Scans

Attara Majesta Ayub - 13522139
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): attaraayub@gmail.com

***Abstract*—This program represents the simplest tool for detecting tumor, aneurysm, and cancer from brain CT scans, utilizing the Boyer-Moore and Levenshtein Distance algorithms. Brain CT scans will undergo preprocessing and pattern matching to identify precise matches, while the Levenshtein algorithm will assess similarity. The improvement of this program aims to support medical professionals in early detection and diagnosis of critical conditions in patients. This paper will analyze the Boyer-Moore logic to optimize its application in medical imaging analysis, enhancing the program's efficacy in clinical settings. The paper also discusses the incorporation of data handling techniques, specifically addressing outliers, to improve the accuracy of the analysis.**

***Keywords*—Boyer Moore; Levenshtein; brain CT scans; tumor; aneurysm; cancer; medical imaging analysis**

I. Introduction

Computed tomography (CT), commonly known as a CT scan, is a diagnostic imaging technique that combines X-rays and computer technology to generate detailed images of the body's internal structures. It provides comprehensive views of various body parts, including bones, muscles, organs, and blood vessels.

CT head, commonly known as CT brain, involves a computed tomography (CT) scan of the brain and surrounding cranial structures. The primary method used is a non-contrast study, though contrast-enhanced phases may be added for specific indications. Non-contrast CT is typically indicated for scenarios like altered mental status, cerebrovascular disease, dementia, head trauma, severe headaches, and seizures. These scans are particularly crucial for detecting acute intracranial hemorrhages, mass effects, and other neurosurgical emergencies due to their high sensitivity and wide availability.

Contrast-enhanced CT scans are employed to assess the permeability of the blood-brain barrier, which can be altered by various pathological processes such as angiogenesis, inflammation, ischemia, and increased intracranial pressure. This method allows for the identification of brain metastases, meningiomas, brain abscesses, necrotic neoplasms, meningitis, multiple sclerosis plaques, lymphomas, and other conditions through abnormal contrast enhancement. The technique for CT brain scans varies, involving either step-and-shoot or volumetric (helical) acquisition methods. Modern scanners predominantly use volumetric acquisition, enabling multi-planar reconstructions and providing comprehensive 3D data sets. This shift from traditional 2D imaging to 3D imaging enhances diagnostic accuracy and flexibility in image analysis.

A CT scan can reduce or avoid the need for invasive procedures to diagnose problems in the skull. This is one of the safest ways to study the head and neck.

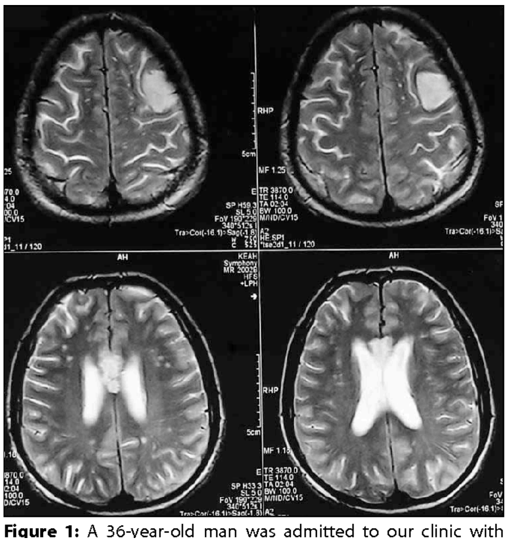


Figure 1: A 36-year-old man was admitted to our clinic with

Fig 1.1) CT scan for primary brain tumors associated with cerebral aneurysm (Taken from Semantic Scholar)

Brain tumors, characterized by abnormal cell growth in the brain, can develop in different areas of the brain or skull, such as the brainstem, skull base, sinuses, and nasal cavity. With over 120 different types, brain tumors vary based on the tissue they originate from. A brain cancer diagnosis typically refers to grade 3 or 4 tumors, which grow rapidly and have a higher likelihood of spreading within the brain.

Brain cancer can spread within the brain or to the spine but rarely metastasizes to other parts of the body. Secondary brain cancer, or metastases, occurs when cancerous cells from elsewhere in the body spread to the brain. Not all brain tumors are cancerous; benign brain tumors grow slowly, have clear borders, and seldom spread. However, they can still pose risks by compressing vital brain areas. Malignant brain tumors are cancerous, growing rapidly and invading healthy brain tissue. Examples include olfactory neuroblastoma and medulloblastoma.

A cerebral aneurysm, also known as a brain aneurysm, is a weakened spot on a brain artery that bulges and fills with blood, potentially causing hemorrhagic stroke or brain damage if ruptured. Most aneurysms form along major arteries at the base of the skull and can rupture, leading to severe health complications or death. While some small aneurysms may remain undetected, all have the potential to rupture and cause bleeding in the brain or surrounding areas.

II. Boyer-Moore Theory

The Boyer-Moore Algorithm is one of the most efficient string matching algorithms in practice, discovered by Robert S. Boyer and J Strother Moore in 1977. This algorithm works by matching from right to left and employs two main rules to shift the pattern further than one character after a mismatch occurs, namely the "bad character rule" and the "good suffix rule".

The bad character rule involves preprocessing the pattern to create a table that records the last occurrence index of each character in it. During the search phase, this information is utilized to determine the pattern's shift relative to the text when mismatches occur. By aligning the last occurrence of a mismatched character in the pattern with its corresponding occurrence in the text, the algorithm maximizes the shift distance, thereby facilitating faster pattern matching.

During the pattern searching process, the algorithm iterates through the text, attempting to align the pattern with the current substring of the text. It starts by comparing characters from right to left, seeking matches. When a mismatch arises, the algorithm consults the bad character table to determine the maximum possible shift distance. This shift is calculated based on the last occurrence index of the mismatched character in the pattern and its occurrence in the text.

The Boyer-Moore algorithm minimizes the number of comparisons required during pattern matching, primarily due to the efficiency of its heuristics. The bad character heuristic enables the algorithm to skip comparisons by leveraging the precomputed last occurrence table, while the good suffix heuristic enables efficient adjustment of the pattern's position based on matching suffixes.

For example, if we want to find the pattern "EXAMPLE" in the text "HERE IS A SIMPLE EXAMPLE," the Bad Character Table for the pattern "EXAMPLE" would be:

TABLE I. Least Occurance Table

E	X	A	M	P	L	Else
6	5	4	3	2	1	7

To facilitate a better understanding of the Boyer-Moore algorithm, a visualization has been created for study cases.

A. *Mismatched Character is Present in the Pattern*

In such cases, we call that character a bad character. When a bad character is detected, the pattern is shifted until it aligns with the mismatched character in the text. For instance, consider the pattern: RPCRQ and the input text: AYRRQMGRQRQ. As we compare the pattern with the input string, we encounter a mismatch between the character R in the text (the bad character) and the character C in the pattern. Consequently, we shift the pattern until the character R in the pattern matches the character R in the text string. Refer to the image provided below for better visualization.

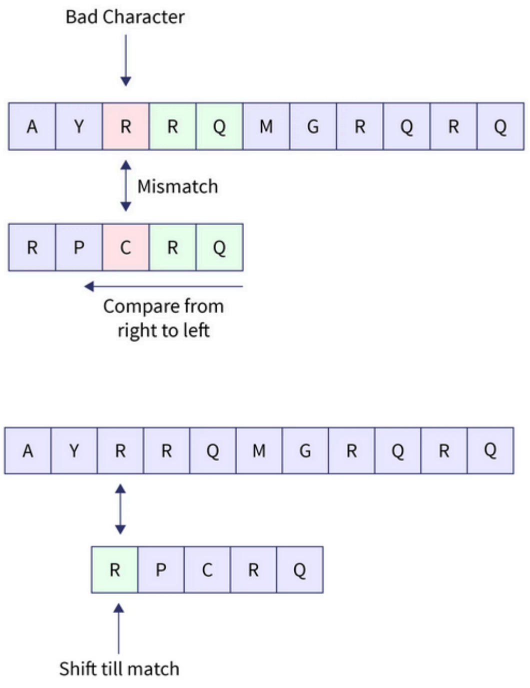


Fig 2.1) Mismatched Character is Present in the Pattern (Taken from Scaler)

B. *Mismatched Character is Not Present in the Pattern*

Using the previous pattern: RPCRQ and different input text: AYRRQMGRPCRQ. As we compare the pattern with the input string, we encounter the mismatch below:

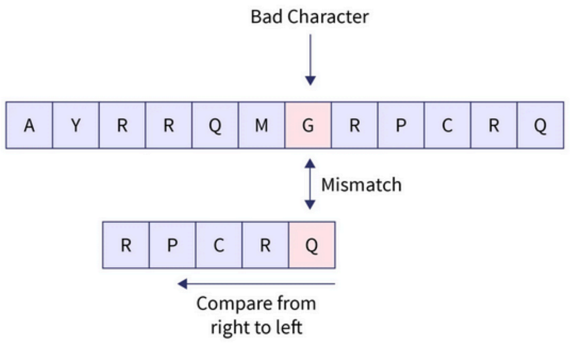


Fig 2.2) Mismatched Character is Not Present in the Pattern (Taken from Scaler)

There is a mismatch between G and Q, and since G does not appear anywhere in the pattern string, we can skip comparing the entire pattern. Instead, we simply shift the pattern until it aligns with the character G in the input text as follows:

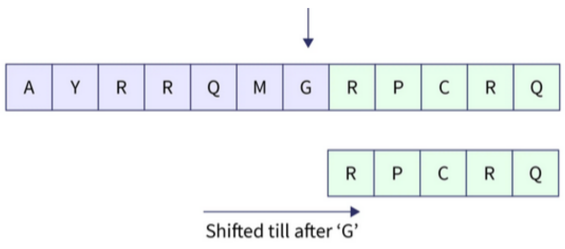


Fig 2.3) Mismatched Character is Not Present in the Pattern

(Taken from Scaler)

Overall, the Boyer-Moore algorithm's ability to minimize comparisons and efficiently adjust the pattern's position contributes to its rapid and accurate matching speed, rendering it invaluable for a wide range of string searching applications.

III. Levenshtein Distance Theory

Levenshtein Distance is defined as the minimum number of operations required to transform one string into another. These operations include insertion, deletion, and substitution. Levenshtein Distance can also be interpreted as the similarity between two strings, as the more similar two strings are, the fewer operations are needed to transform one into the other. Levenshtein Distance is the most popular distance measurement method among similar algorithms, although the difference lies only in the types of operations allowed. The Hamming Distance algorithm only allows substitution operations.

Meanwhile, the Damerau-Levenshtein Distance allows for character transposition in addition to the set of operations defined by Levenshtein Distance. It is often used as a replacement for the classic Levenshtein distance with the same name. Here is the formal definition of Levenshtein Distance between two strings *a, b* in $lev_{a,b}(|a|, |b|)$ where

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j)=0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j)+1 \\ \operatorname{lev}_{a,b}(i,j-1)+1 \\ \operatorname{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Fig 3.1) Formal Definition of Levenshtein Distance (Taken from Baeldung)

The implementation of Levenshtein Distance can utilize a recursive approach, an iterative approach with a 2D matrix, or even just two rows of a matrix. Using only two rows of the matrix helps reduce space complexity. The time complexity of this algorithm is O(3^(m+n)), where m is the length of the first string and n is the length of the second string. Meanwhile, the additional space complexity (auxiliary space) is O(m+n).

IV. Processing CT Scans

Preprocessing brain CT scan involves several steps to prepare it for further analysis. Firstly, the image is converted to grayscale to simplify the data and remove any color information that is not relevant to the analysis. Subsequently, the grayscale image is transformed into a binary format, where each pixel is represented as either black or white based on predefined thresholds, distinguishing between dark areas (ridges) and light areas (valleys).

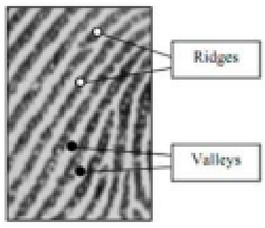


Fig 4.1) Ridge and Valley (Taken from Ogechi Ogbuokiri)

This binary representation simplifies the image further and facilitates subsequent processing steps. Finally, the binary image is divided into sequences of 8 bits, with each sequence representing a single character in the ASCII encoding scheme. This conversion process condenses the image data into a more manageable format for subsequent analysis and comparison with other brain CT scans in the database.

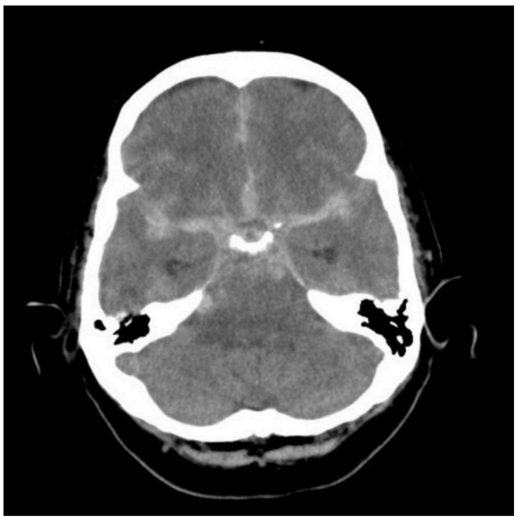
After preprocessing, the Boyer-Moore (BM) algorithm is employed to search for exact matches, representing images identical to those already present in the database. Subsequently, the Levenshtein Distance algorithm is utilized to calculate the similarity between each input image and the database images. This step allows for the detection of subtle differences between images, identifying potential matches.

Additionally, the mean similarity of each folder is computed for further consideration. This program also handles outliers. Outliers are data points that significantly deviate from the rest of the dataset. In the context of similarity calculations between images, outliers represent images that are notably different from the majority of images in a given folder or dataset. These outliers may arise due to various factors, such as anomalies in the images themselves, errors in the preprocessing or comparison process, or the presence of unusual or uncommon images.

Identifying and removing outliers is crucial in data analysis to ensure that the analysis is based on representative and reliable data. In the similarity calculation process, outliers can distort the overall results and lead to inaccurate conclusions. Therefore, by detecting and eliminating outliers, the analysis becomes more robust and meaningful, providing a clearer understanding of the dataset's characteristics and relationships.

Outliers in the similarity data for each folder are identified and removed using the delete outlier method, as no other columns depend on this data, ensuring the integrity of the dataset. Finally, the mean similarity value is calculated after removing outliers, providing a more accurate representation of the dataset for subsequent analysis and decision-making.

TABLE II. IMAGE PREPROCESSING

CT Scan

Binary
11111111 11011000 11111111 11100000 00000000 00010000 01001010 01000110 01001001 01000110 00000000 00000001 00000001 00000001 00000000 01100000 00000000 01100000 00000000 00000000 11111111 11011011 00000000 01000011 00000000 00001000 00000110 00000110 00000111 00000110 00000101 00001000 00000111 00000111 00000111 00001001 00001001 00001000 00001010 00001100 00010100 00001101 00001100 00001011 00001011…
ASCII
ÿØÿà�JFIF��`�`��ÿÛ�C� $.' ",#(7),01444'9=82<.342ÿÛ�C 2!!22222222222222222222222222222222222 222222222222222ÿÀ���"�ÿÄ����������� ÿÄ�µ���}�!1AQa"q2 ‘¡#B±ÁRÑð$3br, %&'()*456789:CDEFGHIJSTUVWXYZcdefghijs tuvwxyzƒ„�†‡ˆ‰Š’ “”” ––˜™š¢£¤¥¦§¨©ª²³´µ¶·¸ ¹ºÂÃÄÅÆÇÈÉÊÒÓÔÕÖ…

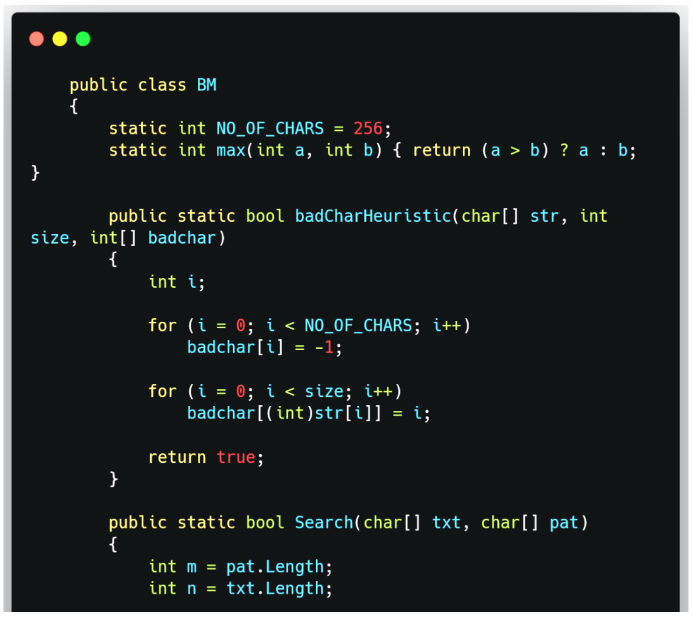
V. THEORY IMPLEMENTATION

A. *Boyer-Moore Algorithm*

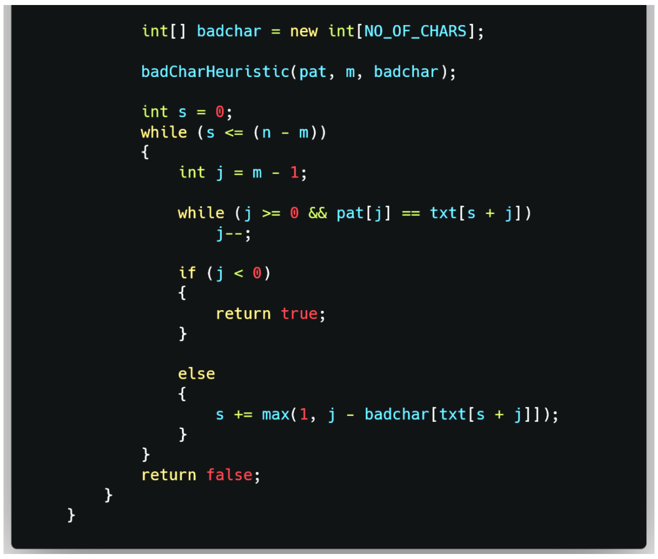
In the provided implementation below, the Boyer-Moore algorithm is encapsulated within the BM class. The Boyer-Moore algorithm implementation in this code focuses on using the bad character heuristic for efficient pattern searching in text. The code includes two main functions: badCharHeuristic() for preprocessing the pattern and Search() for searching the pattern in the text. The NO_OF_CHARS variable defines the number of unique characters (256).

The badCharHeuristic() function initializes all elements of the `badchar` array to -1, then iterates through the pattern to store the last occurrence index of each character. The `Search()` function uses this `badchar` array to find the pattern in the text. It initializes the shift variable `s` to zero and runs a loop while `s` is less than or equal to `n - m`, where `m` is the pattern length, and `n` is the text length. The loop compares characters from the pattern and text. If a mismatch occurs, the pattern shifts based on the `badchar` array values, ensuring a positive shift using max(1, j - badchar[txt[s + j]]).

If the entire pattern matches the text, the function returns true. If the pattern is not found after all iterations, it returns false. The Boyer-Moore algorithm optimizes pattern searching by minimizing unnecessary comparisons, making it more efficient than simple pattern matching methods.



```
    public class BM
    {
        static int NO_OF_CHARS = 256;
        static int max(int a, int b) { return (a > b) ? a : b;
}

        public static bool badCharHeuristic(char[] str, int
size, int[] badchar)
        {
            int i;

            for (i = 0; i < NO_OF_CHARS; i++)
                badchar[i] = -1;

            for (i = 0; i < size; i++)
                badchar[(int)str[i]] = i;

            return true;
        }

        public static bool Search(char[] txt, char[] pat)
        {
            int m = pat.Length;
            int n = txt.Length;
```



```
        int[] badchar = new int[NO_OF_CHARS];

        badCharHeuristic(pat, m, badchar);

        int s = 0;
        while (s <= (n - m))
        {
            int j = m - 1;

            while (j >= 0 && pat[j] == txt[s + j])
                j--;

            if (j < 0)
            {
                return true;
            }

            else
            {
                s += max(1, j - badchar[txt[s + j]]);
            }
        }
        return false;
    }
}
```

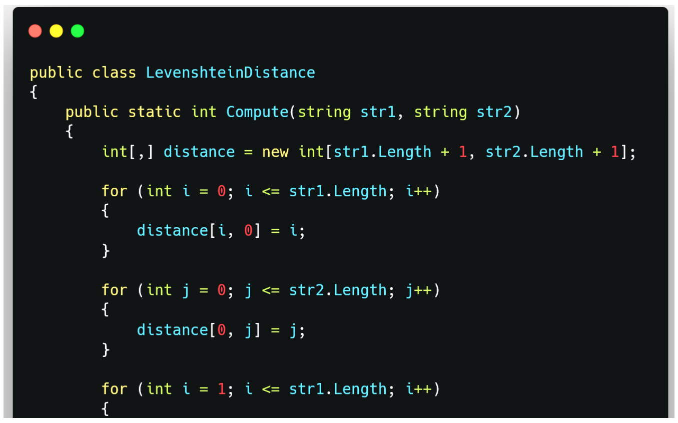
Fig 5.1) Boyer-Moore Algorithm (Taken from program)

B. *Levenshtein Distance*

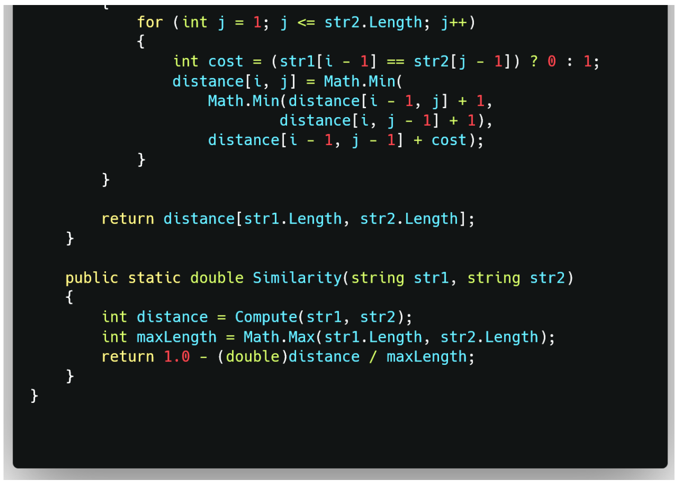
This code implements the Levenshtein Distance algorithm to calculate the minimum number of edits (insertions, deletions, or substitutions) needed to transform one ASCII fingerprint string into another. It includes two main functions: Compute() and Similarity().

The Compute() function takes two strings, `str1` and `str2`, and initializes a 2D matrix `distance` to store edit distances at each comparison point. The first row and column are initialized to represent the edit distance when one string is empty. The algorithm then fills the matrix using nested loops, where each cell `[i, j]` contains the minimum of three operations: deleting, inserting, or substituting a character, with a cost of 0 if characters match or 1 if they differ. The function returns the value in the bottom-right cell, which is the Levenshtein distance between the strings.

The Similarity() function calculates the similarity between two strings using the Levenshtein distance. It calls Compute(), then computes the similarity as `1.0` minus the ratio of the Levenshtein distance to the maximum length of the two strings. The result is a value between 0 and 1, where 1 means the strings are identical, and 0 means they are completely different.



```
public class LevenshteinDistance
{
    public static int Compute(string str1, string str2)
    {
        int[,] distance = new int[str1.Length + 1, str2.Length + 1];

        for (int i = 0; i <= str1.Length; i++)
        {
            distance[i, 0] = i;
        }

        for (int j = 0; j <= str2.Length; j++)
        {
            distance[0, j] = j;
        }

        for (int i = 1; i <= str1.Length; i++)
        {
```

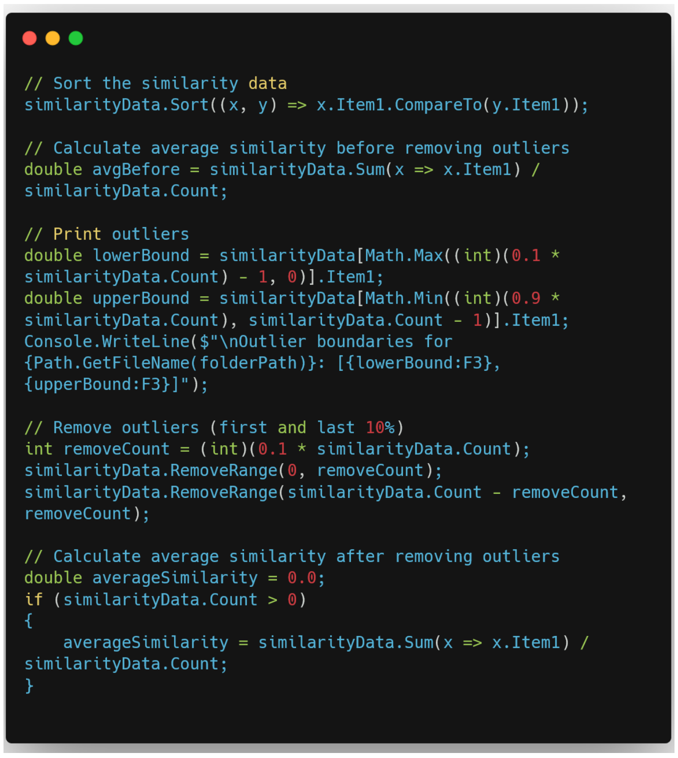


```
            for (int j = 1; j <= str2.Length; j++)
            {
                int cost = (str1[i - 1] == str2[j - 1]) ? 0 : 1;
                distance[i, j] = Math.Min(
                    Math.Min(distance[i - 1, j] + 1,
                            distance[i, j - 1] + 1),
                    distance[i - 1, j - 1] + cost);
            }
        }

        return distance[str1.Length, str2.Length];
    }

    public static double Similarity(string str1, string str2)
    {
        int distance = Compute(str1, str2);
        int maxLength = Math.Max(str1.Length, str2.Length);
        return 1.0 - (double)distance / maxLength;
    }
}
```

Fig 5.2) Levenshtein Distance Algorithm (Taken from program)

C. *Handling Outliers*

The code sorts a collection of similarity data in ascending order based on the similarity values. It then calculates the average similarity before removing any outliers. The outliers are defined as the first and last 10% of the sorted data. To determine these outlier boundaries, the code identifies the similarity values at the 10th and 90th percentiles. It then prints these boundaries for reference. After identifying the outliers, the code removes them from the dataset by deleting the first and last 10% of the entries. Finally, the code calculates the average similarity of the remaining data and stores it in the averageSimilarity variable, ensuring that this calculation is only performed if there is data left after outlier removal.

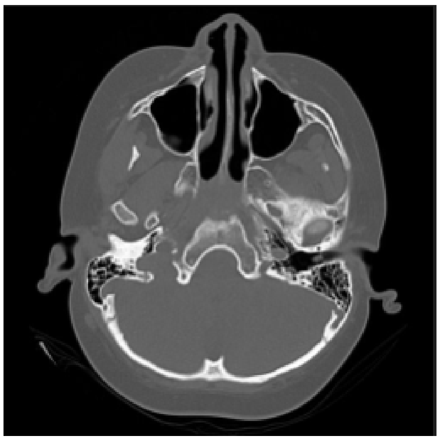


```
// Sort the similarity data
similarityData.Sort((x, y) => x.Item1.CompareTo(y.Item1));

// Calculate average similarity before removing outliers
double avgBefore = similarityData.Sum(x => x.Item1) /
similarityData.Count;

// Print outliers
double lowerBound = similarityData[Math.Max((int)(0.1 *
similarityData.Count) - 1, 0)].Item1;
double upperBound = similarityData[Math.Min((int)(0.9 *
similarityData.Count), similarityData.Count - 1)].Item1;
Console.WriteLine($"\nOutlier boundaries for
{Path.GetFileName(folderPath)}: [{lowerBound:F3},
{upperBound:F3}]");

// Remove outliers (first and last 10%)
int removeCount = (int)(0.1 * similarityData.Count);
similarityData.RemoveRange(0, removeCount);
similarityData.RemoveRange(similarityData.Count - removeCount,
removeCount);

// Calculate average similarity after removing outliers
double averageSimilarity = 0.0;
if (similarityData.Count > 0)
{
    averageSimilarity = similarityData.Sum(x => x.Item1) /
similarityData.Count;
}
```

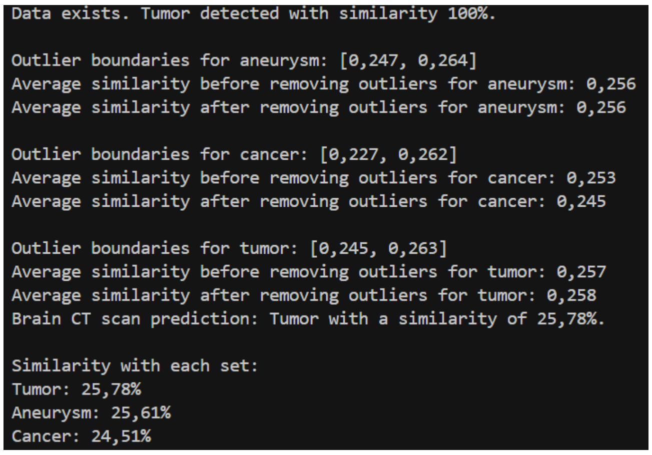
Fig 5.3) Handling Outlier Algorithm (Taken from program)

VI. Program Testing

Testcase 1

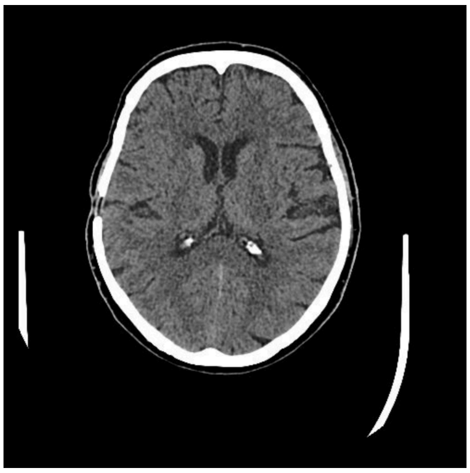


Output
Predicted: Tumor
Actual: Tumor

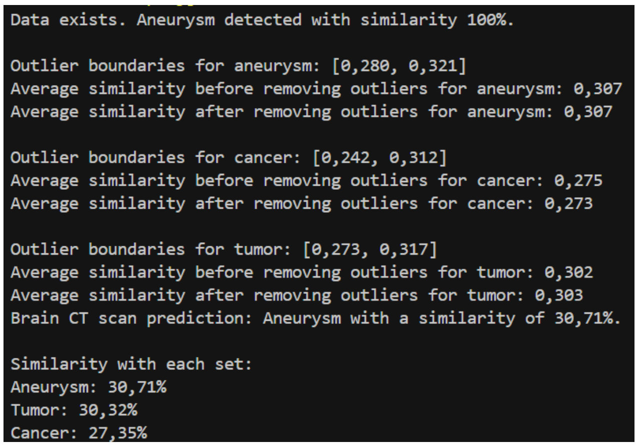


```
Data exists. Tumor detected with similarity 100%.

Outlier boundaries for aneurysm: [0,247, 0,264]
Average similarity before removing outliers for aneurysm: 0,256
Average similarity after removing outliers for aneurysm: 0,256

Outlier boundaries for cancer: [0,227, 0,262]
Average similarity before removing outliers for cancer: 0,253
Average similarity after removing outliers for cancer: 0,245

Outlier boundaries for tumor: [0,245, 0,263]
Average similarity before removing outliers for tumor: 0,257
Average similarity after removing outliers for tumor: 0,258
Brain CT scan prediction: Tumor with a similarity of 25,78%.

Similarity with each set:
Tumor: 25,78%
Aneurysm: 25,61%
Cancer: 24,51%
```

Testcase 2

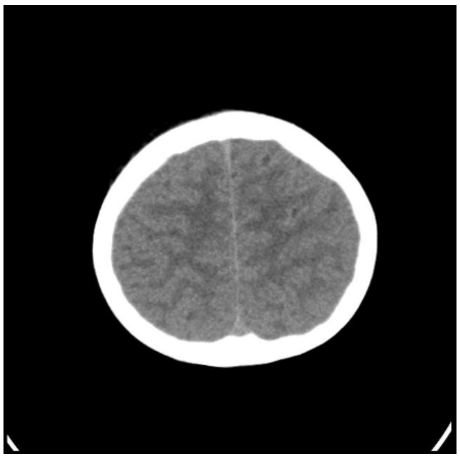


Output
Predicted: Aneurysm
Actual: Aneurysm

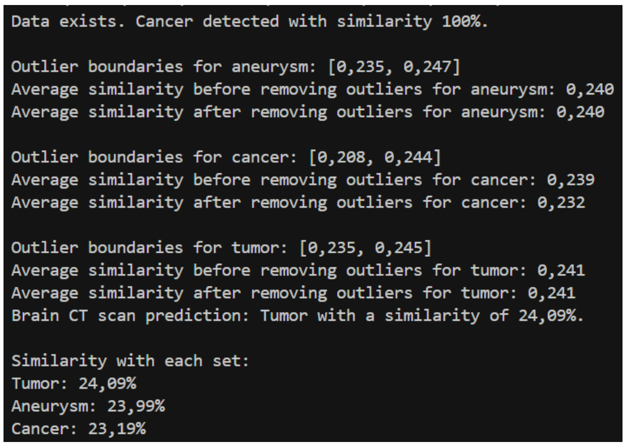


```
Data exists. Aneurysm detected with similarity 100%.

Outlier boundaries for aneurysm: [0,280, 0,321]
Average similarity before removing outliers for aneurysm: 0,307
Average similarity after removing outliers for aneurysm: 0,307

Outlier boundaries for cancer: [0,242, 0,312]
Average similarity before removing outliers for cancer: 0,275
Average similarity after removing outliers for cancer: 0,273

Outlier boundaries for tumor: [0,273, 0,317]
Average similarity before removing outliers for tumor: 0,302
Average similarity after removing outliers for tumor: 0,303
Brain CT scan prediction: Aneurysm with a similarity of 30,71%.

Similarity with each set:
Aneurysm: 30,71%
Tumor: 30,32%
Cancer: 27,35%
```

Testcase 3



Output
Predicted: Tumor
Actual: Cancer



```
Data exists. Cancer detected with similarity 100%.

Outlier boundaries for aneurysm: [0,235, 0,247]
Average similarity before removing outliers for aneurysm: 0,240
Average similarity after removing outliers for aneurysm: 0,240

Outlier boundaries for cancer: [0,208, 0,244]
Average similarity before removing outliers for cancer: 0,239
Average similarity after removing outliers for cancer: 0,232

Outlier boundaries for tumor: [0,235, 0,245]
Average similarity before removing outliers for tumor: 0,241
Average similarity after removing outliers for tumor: 0,241
Brain CT scan prediction: Tumor with a similarity of 24,09%.

Similarity with each set:
Tumor: 24,09%
Aneurysm: 23,99%
Cancer: 23,19%
```

VII. Conclusion

The Boyer-Moore algorithm, known for its efficiency in accelerating data search processes, plays a crucial role in enhancing the speed and accuracy of pattern recognition tasks. However, to achieve accurate predictions in medical imaging, additional components beyond the algorithm itself are necessary. These components include sophisticated data modeling techniques, extensive datasets, and supplementary methodologies tailored to specific domains.

In the context of medical imaging, particularly in brain CT scan analysis, accurate predictions are essential for diagnosing neurological conditions like tumors, aneurysms, and other abnormalities. While the Boyer-Moore algorithm aids in quickly identifying patterns within imaging data, its effectiveness in making precise predictions relies on the quality and quantity of available data. Therefore, integrating advanced data modeling techniques allows for a more comprehensive analysis of imaging data, enabling the algorithm to discern subtle patterns indicative of medical conditions with greater accuracy.

Furthermore, the inclusion of supplementary methodologies, such as machine learning algorithms or statistical analysis techniques, can complement the capabilities of the Boyer-Moore algorithm in medical imaging applications. These methodologies help refine predictions by identifying complex patterns and correlations within imaging data, thereby improving diagnostic accuracy and clinical decision-making.

In summary, while the Boyer-Moore algorithm provides a robust framework for accelerating data search processes, achieving accurate predictions in medical imaging and other domains requires a multifaceted approach. By incorporating advanced data modeling techniques and supplementary methodologies alongside the algorithm, healthcare professionals can enhance diagnostic precision, optimize patient care, and ultimately improve outcomes for individuals undergoing medical imaging procedures like brain CT scans.

Acknowledgment

I would like to sincerely express my gratefulness to Allah SWT. that this paper can be completed well. Additionally, I extend my deepest appreciation to Mr. Rila Mandala, Mr. Monterico Adrian, and Mr. Rinaldi Munir, Algorithm Strategies lecturers at Institut Teknologi Bandung, for their invaluable guidance, support, and engaging lectures. I am also grateful to my family and friends, particularly K-03 Teknik Informatika 2022, for their encouragement and for the endless support.

References

[1] Boyer, R. S., & Moore, J. S. (1977). A fast string searching algorithm. Communications of the ACM, 20(10), 762-772.

[2] Rinaldi Munir, diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/String-Matching-dengan-Regex-2019.pdf

[3] Anonymus. (2023, May 15). Boyer–Moore string-search algorithm. In Wikipedia, The Free Encyclopedia

[4] Scaler. "Boyer-Moore Algorithm." Accessed June 12, 2024. https://www.scaler.com/topics/boyer-moore-algorithm/

[5] Hopkins Medicine. "Brain Tumor." Accessed June 12, 2024. https://www.hopkinsmedicine.org/health/conditions-and-diseases/brain-tumor

[6] The Brain Tumour Charity. "Brain Cancer." Accessed June 12, 2024. https://www.thebraintumourcharity.org/brain-tumour-diagnosis-treatment/how-brain-tumours-are-diagnosed/brain-tumour-biology/brain-cancer/

[7] Mayo Clinic. "Brain Aneurysm: Symptoms and Causes." Accessed June 12, 2024. https://www.mayoclinic.org/diseases-conditions/brain-aneurysm/symptoms-causes/syc-20361483

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

[illegible]

Attara Majesta Ayub 13522139