IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Study On Design And Analysis Of Duplication Detection Tool For Articles

[1]Sanket Prakash Patil, [2]Harshvardhan Popat Tambe, [3]Harshwardhan Prabodhan Pawar, [4]Anuja Rajendra Bhamare

[1]Student, [2]Student, [3]Student, [4]Student
Computer Engineering,
MET's Institute of Engineering, Nashik, India

Abstract: Large-scale public opinion campaigns and internet portals that encourage people to customize content result in a lot of duplicate papers, which raises processing and storage costs but is rarely a severe issue. Duplicate papers are frequently found in media collections. While writers may make small changes to previous articles by adding new information or making minor corrections, editors often modify verified articles to suit print editions or include local context. These practices contribute to the presence of duplicated content within media collections. Since great precision is required for real-time applications, it is vital to eliminate nearly identical documents from collections. To identify duplicate articles, we suggested a system that uses the hashing technique along with a hash index and similarity.

Nowadays, due to the rapid development of electronic media and the large number of articles created online, duplication detection is required. Additionally, there is a clear connection between plagiarism in article content and the duplication of articles. Newspaper articles have been the subject of most of the previous research on duplicate detection. Further research has been done on finding duplicate content in the most recent internet articles. The proposed system will concentrate on finding duplicate articles. The fingerprinting approach and hash index are used in our suggested system to find duplicate articles. To validate our proposed approach, 'Verifyr' tool is meant to crawl huge number of articles data for detection. Moreover, it will apply our approach to detect plagiarism articles based on our duplication results in future. Further with the help of tool, it will try to conduct an empirical study and summarize a few of most used plagiarism patterns in plagiarism articles.

***Index Terms* - Duplicate detection, articles, plagiarism detection.**

I. Introduction

With the rapid development of electronic media, it enables rapid publishing and instant access to many articles. Meanwhile, many articles are produced online, and articles duplication can no longer be ignored, since duplicate articles will increase the redundancy and management costs. Moreover, articles duplication is directly related to articles plagiarism. In the field of journalism, plagiarism is considered a breach of journalistic ethics, so it is very important to detect duplication in articles.

Duplication detection is very useful for a variety of tasks (e.g., file management, copyright protection and plagiarism prevention). However, existing works about duplication detection mainly focus on documents or code duplication detection, and only a few works aim at duplication detection in articles. Moreover, existing works just detect and analyze duplication in articles. In the past, article publishing was restricted to agencies. However, with the rise of electronic media, anyone can now publish and share articles online. Unfortunately, there is a deficiency in duplication detection and analysis. Therefore, our research is concentrated on the detection of duplication in articles with a significant amount of data.

In this paper, in order to support accurate duplication detection of articles, we propose a technique and implement it as a tool. We first normalize the content of each article, such as removing spaces and images. Then we design a hash-based algorithm to process each sentence in the article, which can be seen as a fingerprinting technique. In other words, each article is processed as a set of hash values, where each hash value represents a text sentence in the article. Furthermore, for pairs of obtained hash set (i.e., pairs of articles), Our proposed tool uses Jaccard similarity coefficient as similarity function to detect duplication.
Finally, we conducted an empirical study to explore plagiarism patterns in articles and developed a model.

II. Literature Survey

Earlier research on identifying near-duplicates prioritized assessing document similarity without relying on computationally intensive whole-text comparisons. To tackle this issue, a technique was developed that involves document hashing and the comparison of individual hash values. This method enables the identification of identical content within documents. By calculating hash values for each document and comparing them, duplicate content can be effectively identified and addressed. Duplicate documents are identified when their hash values match. However, this method falls short in determining the degree of similarity between two documents, making it unsuitable for near-duplicate identification. Modern approaches for near-duplicate detection typically utilize different methods to generate and compare compact representations of documents. These approaches aim to create concise yet informative representations of documents that can be efficiently compared to identify near-duplicate content. By employing various techniques, such as shingling, locality-sensitive hashing, or other advanced algorithms, these methods effectively capture the essence of documents and enable reliable near-duplicate detection. While these approaches differ in how they create and evaluate representations, their goal remains the same – to facilitate effective near-duplicate detection.

Previous studies on near-duplicate detection focused on assessing the similarity between two documents without relying on computationally demanding whole-document bit-wise comparisons. An approach commonly used involved hashing the documents and comparing their hash values to identify duplicate documents, where matching hash values indicated duplicates. However, this method is not suitable for identifying nearly identical documents as it does not provide information on the degree of similarity between two documents. To tackle this issue, most algorithms for near-duplicate identification utilize compact representations of documents. These representations are created and compared using various methods to accurately identify near-duplicates. By employing efficient techniques, these algorithms can effectively determine the similarity between documents and identify near-duplicate content. This approach allows for the detection of near duplicates without the need for computationally demanding comparisons of entire texts, providing a more practical and scalable solution.

The shingle technique [1] Broder (2000) and Broder (1997), Document viewing involves treating a document as a collection of n-grams or shingles, which are overlapping textual sequences of words. The similarity between two texts can be determined by calculating the set similarity. If the calculated similarity exceeds a predefined threshold, the documents are considered near duplicates. However, the computational complexity increases due to the large number of generated shingles.
To overcome this challenge, Broder proposed a technique called document sketching. It involves either discarding specific shingles that meet a certain condition (e.g., satisfying S mod m = 0) or selecting the set of minimum hash values from a random subset of shingles using permutations. By implementing document sketching, the computational burden associated with a vast number of shingles can be mitigated. This approach offers a more efficient way to identify near duplicates while maintaining satisfactory accuracy.

Cosine similarity (Salton et al., 1975) [2] is a method used to evaluate document similarity. It involves converting documents into vectors using term frequency-inverse document frequency (TF-IDF) scores. Each element of the vector represents the significance of a particular phrase within the corpus. By calculating the cosine distance between the vectors of two documents, their similarity can be quantified. A smaller cosine distance indicates a higher degree of similarity between the documents. This approach provides a quantitative measure to assess document similarity by leveraging TF-IDF scores and cosine similarity.

Locality-sensitive hashing, introduced by Charikar (2002) [3], is a method used to measure document similarity while minimizing memory usage compared to storing complete vector representations of each document. This technique employs hashing functions to determine the likelihood of two documents having matching hash values, which indicates their similarity. Each document is hashed using one of these functions, and the resulting hash values are combined to form a vector representation of the document. To identify near-duplicates, the number of matching vector elements between two documents is compared against a predetermined threshold. If the count exceeds the threshold, the documents are considered to be near duplicates. This approach offers an efficient way to assess document similarity without requiring extensive memory resources.

Chowdhury et al. (2002) introduced the technique known as I-Match [4], which generates efficient document fingerprints by hashing important tokens found in a document. If two documents produce the same fingerprint, they are classified as near duplicates. This technique is capable of detecting even the slightest changes in document content. Another variation proposed by Kocz et al. (2004) [5] involves incorporating multiple lexicon variants. The fingerprint generation process includes hashing the document multiple times, with each iteration using a different lexicon variant and removing certain parts of the lexicon. As a result, a vector of hash values is utilized to represent each document. When determining the near duplicity of two documents, a collision is identified if any matched pairs of items in their respective hash value vectors coincide. This technique provides an efficient approach to identify near duplicates by examining collisions in the hash value vectors.

In 2006, Henzinger [6] proposed a method to identify near-duplicates by combining the techniques of shingling, introduced by Broder in 1997 [1], and locality-sensitive hashing, introduced by Charikar in 2002 [3]. The method involves two key steps. Firstly, the documents are processed using shingling, which involves generating overlapping textual sequences of words known as shingles. This step helps identify potential near-duplicate candidates by capturing common phrases and word sequences within the documents. Secondly, locality-sensitive hashing is applied to filter and identify highly similar documents. This technique efficiently reduces the computational burden by comparing hash values instead of the entire document contents. By integrating shingling and locality-sensitive hashing, the proposed method achieves effective identification of near-duplicates. Henzinger reported promising results when applying this approach to large datasets, successfully detecting close duplicate documents on the same website as well as across different websites.

III. METHODOLOGY(TEXT)

3.1 Data Extraction

3.2 Dataset Pre-processing

3.2.1 Data Pre-processing Steps

3.2.1.1 Data Cleaning

The statistics may contain a lot of incorrect and missing information. Record cleansing is carried out to handle this portion. It involves dealing with incomplete facts, noisy facts, etc.

3.2.1.2 Missing Data

This situation develops when some statistics are missing from the data themselves. It can be dealt with in a number of ways. Among them are: 1. Ignore the tuples: This strategy works well when the dataset we have is huge and a tuple has numerous missing values. 2. Complete the missing values: There are numerous ways to attempt this task. You can choose to manually fill in the missing values, use a characteristic suggestion, or choose the highest possible value.

3.3 Normalization

In the context of identifying duplicate content in articles, a characteristic matrix is formed, consisting of various elements such as "Title, Abstract, Content, AuthorId, AuthorName, PublicTime, CommentNum, Category, Link." Each element represents a characteristic vector containing the corresponding information for all articles. These key components are initially extracted from the dataset for each article. Although the primary objective is to detect content duplication, other elements such as the title, author, and publication time are also considered significant as they contribute to the empirical study of plagiarism in publications. To focus on identifying content duplication, the article's content is further analyzed and normalized. During this analysis, it is observed that copycats frequently manipulate the placement of photos within the article as a means to evade plagiarism detection. This finding highlights the importance of considering various aspects when identifying potential plagiarism in articles.

3.4 Fingerprinting and Hashing algorithm.

To facilitate the detection of slight changes in duplicate content, signatures or hash values are computed for each line of text in the article using the fingerprinting process. Let's assume that A represents the collection of normalized text data, where A = a1, a2 up to an represents the content of the n-th article. In this context, each article can be seen as a series of sentences. For each sentence in every article, a hash value v is calculated. This process ensures that even small modifications within the content are captured and represented by distinct hash values. By generating hash values for each sentence, the fingerprinting technique enables efficient comparison and identification of potentially duplicate or closely related content across articles.

3.5 Jaccard similarity Algorithm

Jaccard Similarity is a commonly used measure to assess the similarity between different entities, such as text documents. It can be employed to determine the similarity between sets or uneven binary vectors. In literature, Jaccard Similarity is denoted by the letter J and is also referred to as Jaccard Index, Jaccard Coefficient, Jaccard Dissimilarity, or Jaccard Distance. In the context of article similarity, let HashSetaj represent the set of hash values contained in article aj, and let HashSetak represent the set of hash values contained in article ak. The Jaccard similarity coefficient between these two articles is defined as the ratio of the size of their intersection to the size of their union. It measures the extent of overlap between the sets of hash values, indicating the similarity between the articles. The higher the Jaccard similarity coefficient, the more similar the articles are considered to be.

$$\mathrm{sim}(a_j, a_k) = \frac{|hashSet_{a_j} \cap hashSet_{a_k}|}{|hashSet_{a_j} \cup hashSet_{a_k}|},$$

Figure 3.5: Jaccard Similarity Coefficient Formula

The quantity of matching hash values is determined by computing the intersection of hashSetaj and hashSetak. In other words, it is determined how often sentences are repeated. Besides, by further calculating a union between them, if the sim(aj , ak) satisfies θ specified by user.

3.6 Measuring and Reporting

3.7 Duplication Detection

IV. METHODOLOGY(IMAGE)

4.1 Image Extraction

The proposed system allows users to input both image and PDF files for the purpose of extracting images. The extracted images are then sent to the training module for further processing. This approach simplifies the image extraction process and makes it more efficient, as users can extract images from PDF file format using the same system. By incorporating this feature into the system, users can easily extract images from PDF files and seamlessly integrate them into their workflow for further use.

4.2 Feature Extraction

In our project, we utilized feature extraction to transform raw data into numerical features while preserving the information in the original dataset. This approach allowed us to achieve better results than applying algorithms directly to the raw data. We used feature extraction to predict data duplication, a crucial task in various applications, including plagiarism detection and content filtering. By extracting relevant features from the data and using them to train our model, we were able to accurately predict data duplication and improve the effectiveness of our system. Feature extraction proved to be an essential step in achieving our project's goals.

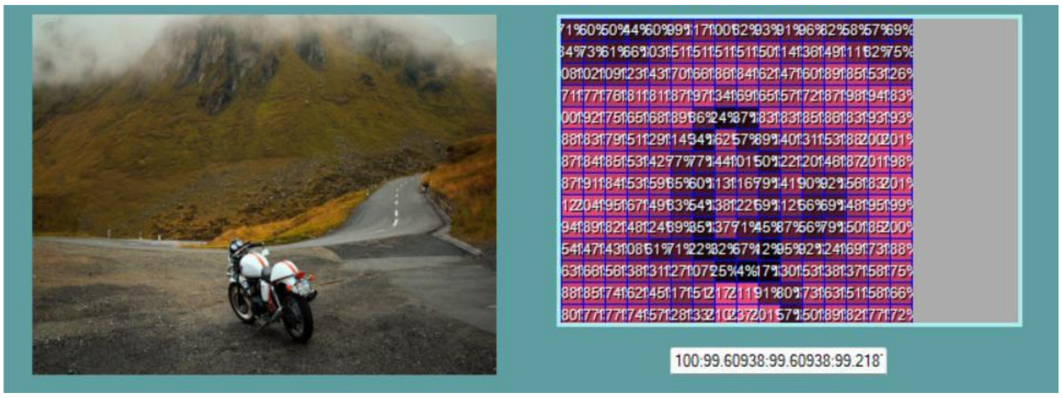


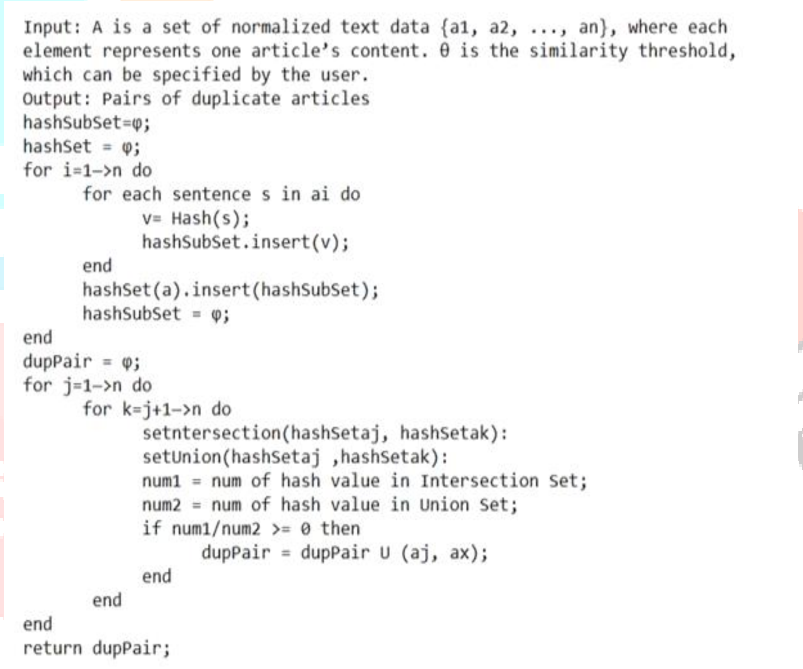
Figure 4.2: Feature Extraction of Image in Verifyr

4.3 Duplication Detection

V. Approach

5.1 Algorithm for Text Plagiarism

Given a set of normalized text data, we use fingerprinting algorithm combining with hash technique to process each. article's data. We evaluate pairs of articles' text are similar by measuring their ratio of matched hash values (i.e.. matched sentences), and report those satisfying the similarity threshold. Algorithm I describe the steps in detail. It works in two steps: (1) Fingerprinting and generating hash index (lines 1- 10); and (2) Measuring similarity and reporting duplication (lines 11-23). Each step is described as follows.



```
Input: A is a set of normalized text data {a1, a2, ..., an}, where each
element represents one article's content. θ is the similarity threshold,
which can be specified by the user.
Output: Pairs of duplicate articles
hashSubSet=φ;
hashSet = φ;
for i=1->n do
        for each sentence s in ai do
                v= Hash(s);
                hashSubSet.insert(v);
        end
        hashSet(a).insert(hashSubSet);
        hashSubSet = φ;
end
dupPair = φ;
for j=1->n do
        for k=j+1->n do
                setntersection(hashSetaj, hashSetak):
                setUnion(hashSetaj ,hashSetak):
                num1 = num of hash value in Intersection Set;
                num2 = num of hash value in Union Set;
                if num1/num2 >= θ then
                        dupPair = dupPair U (aj, ax);
                end
         end
end
return dupPair;
```

Figure 5.1: Duplication Detection For Text

5.2 Algorithm for Image Plagiarism

The algorithm receives a dataset subset, an autoembedder model, and an epoch parameter as inputs. It initializes the autoembedder model, which is a neural network designed to encode data into a lower-dimensional space. It also constructs a Siamese network, a specific type of neural network architecture where two identical subnetworks share weights. During training, the algorithm uses the Siamese network to compare pairs of input samples and determine their similarity. The goal is to learn representations that capture meaningful information about the input data. By adjusting the shared weights during the training process, the network improves its ability to differentiate between similar and dissimilar samples. The algorithm iterates through multiple epochs, updating the weights and optimizing the network's performance. The aim is to retrieve similar outputs based on the learned representations in the lower-dimensional space created by the autoembedder model.

```
Input: Subset of the dataset for training X, AutoEmbedder model M, Number of iterations epochs,
 Training batch per iteration batchSize, Distance hyper parameter α, Query images Q

Result: Retrieve images of the query image initialize an AutoEmbedder models m;
Build a siamese network S with AutoEmbedder m as identical subnetwork;

iter ← 0;
while iter < epochs do initialize two empty input data set I and I0 ;
      initialize an empty target output set Y ;
      b ← 0;

      while b < batchSize do
            Randomly make a boolean choice on 0.5 probability;
            if choice is true then
                  select two random data input xi and xj containing must link constraint cij == 0;
                  append xi to I, xj to I 0 and cij to Y ;
            else
                  select two random data input xi and xj containing must link constraint cij == α;
                  append xi to I, xj to I 0 and cij to Y ;
       b = b +1;
train siamese network S with inputs I, I0 and Y ;
iter = iter +1;
Generate embedding points of X, using trained AutoEmbeder AE;
Apply k-mean to generate embedding point of X;

for each q ∈ Q do
      Generate embedding point of the query image q;
      Apply k-means to find relevant image of the coresponding q;
```

Figure 5.2: Duplication Detection For Image

VI. System Architecture

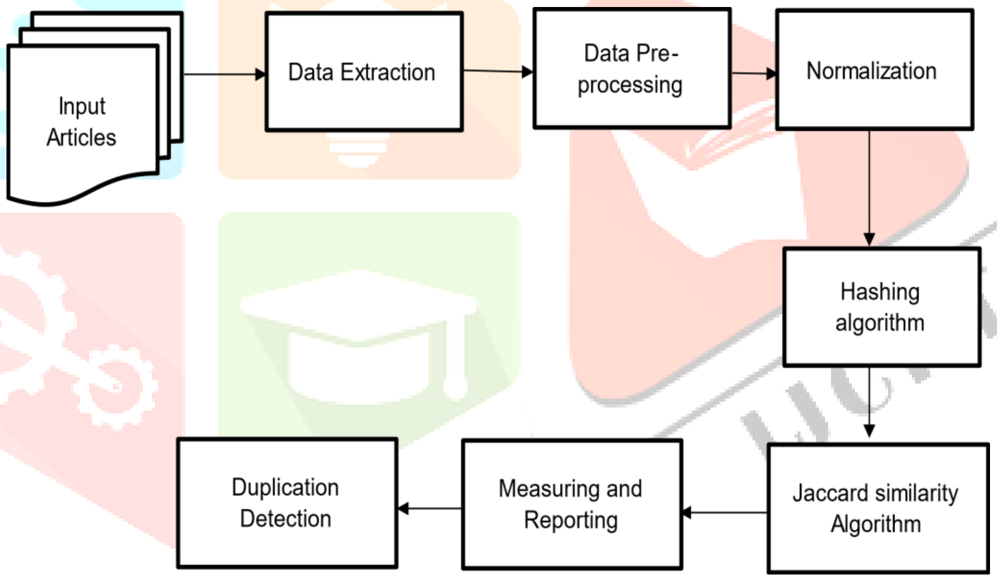


Figure 6.2: Duplication Detection For Image

VII. Experimentation And Results

To validate our proposed approach, we initially gathered a dataset of articles by crawling various sources. Subsequently, we evaluated our tool's effectiveness by utilizing it to detect data duplication within the collected articles. This approach allowed us to assess the tool's capabilities and determine its accuracy in identifying duplicated content. By applying our proposed approach to the collected dataset, we were able to demonstrate the effectiveness of our tool and provide insights into its performance. Overall, the evaluation process was crucial in validating our proposed approach and demonstrating its ability to detect data duplication effectively.

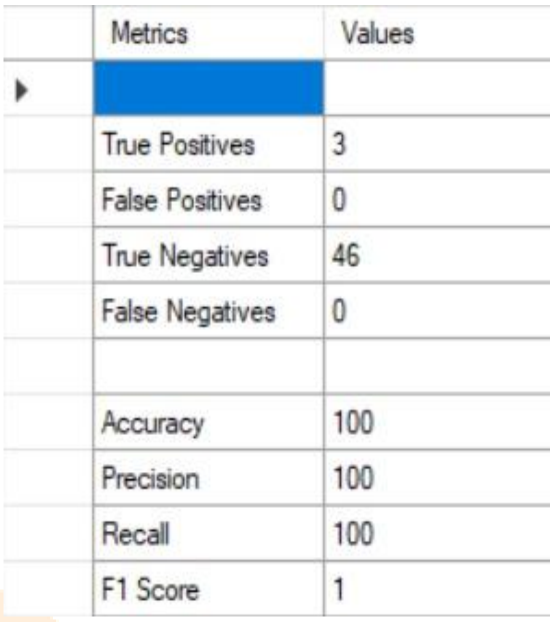
7.1 Data Collection

The dataset used in our review paper comprises 50103 articles sourced from various authors and publications. The articles span across different genres, including news, games, educational, organizational, and more, providing a diverse and comprehensive dataset for our research. This dataset offers insights into the preferences of different authors, and the quality and diversity of content in various publications. Our tool provides opportunities for researchers for data preprocessing, identifying and removing duplicated content, exploring the potential techniques and identifying duplicate images in published articles.

To validate the performance of our proposed approach, we collected a dataset of 50103 articles from various sources. We then applied our approach to detect data duplication within this dataset. By doing so, we were able to assess the effectiveness of our approach and determine its accuracy in identifying duplicated content. This validation process was crucial in demonstrating the utility of our approach and providing insights into its performance. Overall, by applying our proposed approach to this large and diverse dataset, we were able to show its effectiveness in detecting data duplication and improve its potential for use in various applications, including content filtering, plagiarism detection and feature extraction.

7.2 Data Duplication Detection Results (Text)

Our tool for detecting duplication in articles is showcased after gathering a vast amount of data from articles. The tool was tested on a machine equipped with a quad-core CPU, 8GB of memory, and an SSD. We set the similarity threshold of the tool to 75%.



Metrics	Values
True Positives	3
False Positives	0
True Negatives	46
False Negatives	0
Accuracy	100
Precision	100
Recall	100
F1 Score	1

Figure 7.2.1 : Text Articles Duplication Results

To begin, we employed the tool to identify duplication in each category of articles and presented the detection outcomes as a percentage of duplicate articles in descending order. Additionally, we evaluated the precision of our tool using a confusion matrix and validated the results manually.

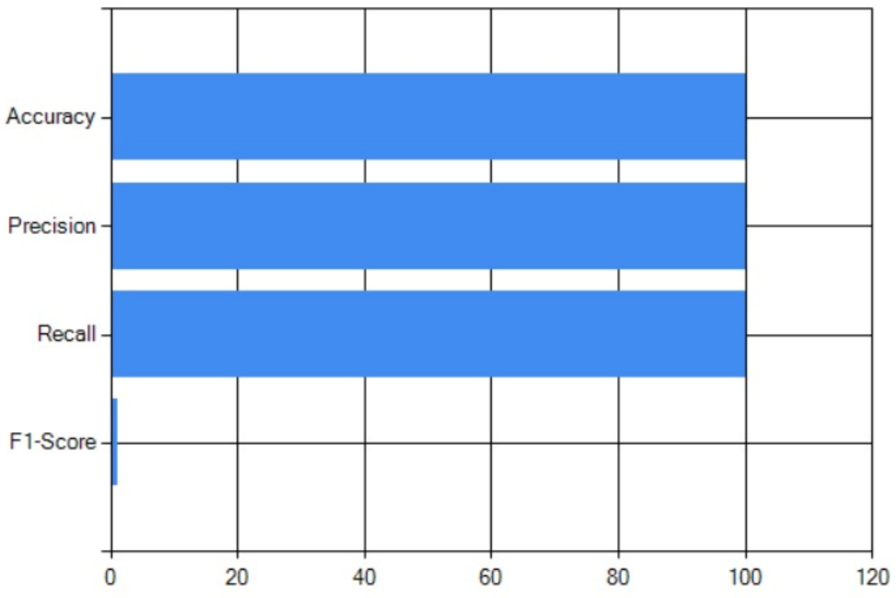


Figure 7.2.2 : Diagrammatical Representation of Text Articles Duplication Results

According to the figure above, our plagiarism detection tool successfully identified duplicate articles with a matching rate of 95.84%, exceeding the threshold value. Based on this outcome, our tool calculated the confusion matrix values, resulting in 100% accuracy, precision, and recall, and an F1 score of 1. These metrics indicate the high performance and effectiveness of our tool in identifying plagiarism in text articles. This analysis provides evidence of the reliability of our approach and highlights its potential for use in various applications that require accurate and efficient plagiarism detection. Overall, these results demonstrate the value of our tool in addressing the issue of content duplication in the digital age.

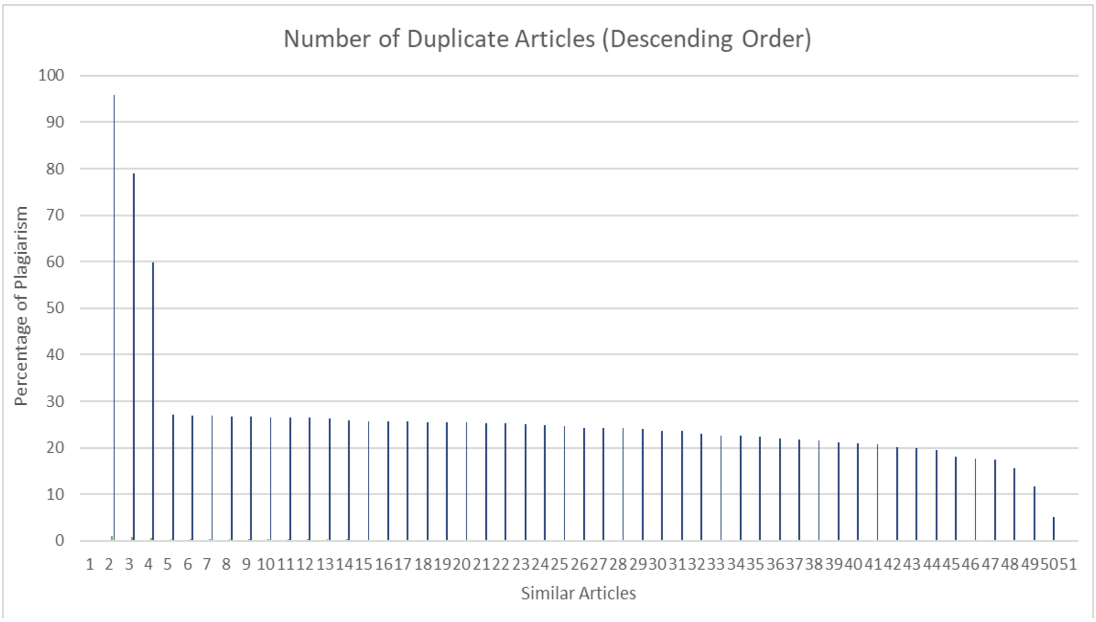
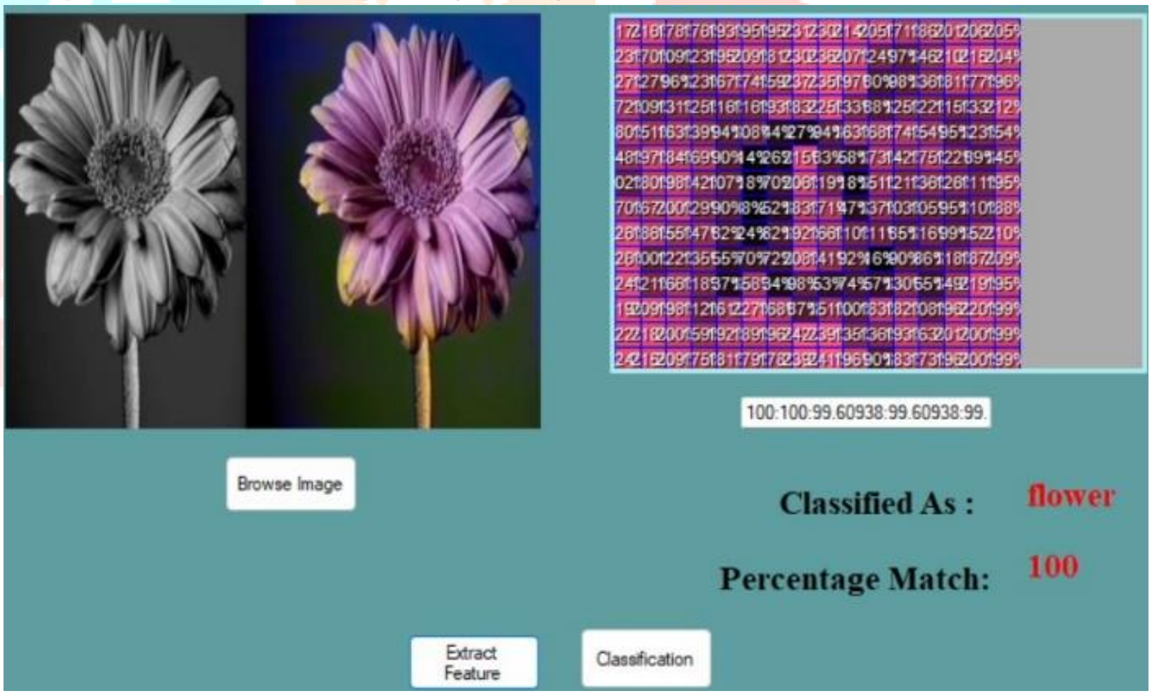


Figure 7.2.3: Line Chart of Output Percentages

Empirically depicted in the above figure is the outcome of our tool, wherein the percentage of plagiarism is shown on the y-axis, while the articles from the dataset are plotted on the x-axis.

7.3 Data Duplication Detection Results (Image)

The same tool was implemented to identify duplicate images in articles, and it was tested on the same machine. By utilizing feature extraction techniques, the tool calculated the percentage of image matches in the dataset.



[illegible]

```
100:100:99.60938:99.60938:99
```

Classified As : flower

Percentage Match: 100

Figure 7.3: Image Duplication Results

Empirically demonstrated in the above figure is the correct classification of a given image, represented as a 100% match as ‘flower’. This classification was achieved through our tool, for the image which was previously trained by extracting features. The percentage match was calculated based on the extracted features of the image, indicating the effectiveness of our approach for accurate image classification.

In the image duplication detection module, the images are thoroughly checked for any manipulations done by another author with the intention of committing plagiarism in their article. The changes made to the image can range from simple rotations to complex partial morphing, making it difficult to detect without a proper tool. Our proposed approach employs feature extraction techniques to identify the unique characteristics of the image, allowing for precise matching and detection of any alterations made. The tool has shown promising results in detecting image duplications in our dataset of articles, which consists of a wide range of images from various sources and authors.

VIII. Conclusion And Future Work

We have presented an article duplication detecting technique and worked on text and image and implemented proposed system, Verifyr. Based on our collected data, a total number of 50,103 Articles, our approach can accurately detect duplicate articles with the precision of 98%. Furthermore, articles duplication is directly related to articles plagiarism. We successfully apply our approach to identity articles within dataset of plagiarism articles based on our duplication results, and then show empirically the plagiarism patterns we found. We offer our approach as an article, image and plagiarism detector, which is useful for duplication management and plagiarism prevention of articles.

For our future work, we plan to extend our study by collecting a larger dataset of articles from multiple sources. This will enable us to investigate the issue of cross-platform duplication and plagiarism detection, which involves identifying instances of plagiarism across different publishing platforms.

References

[1] Andrei Z. Broder, "On the resemblance and containment of documents", digital Systems Research Center 130 Lytton Avenue, 1997.

[2] G. Salton, A. Wong, C. S. Wang, "A vector space model for automatic indexing", Association for Computing Machinery, November 1975

[3] Moses S. Charikar, "Similarity estimation techniques from rounding algorithms", Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 2002

[4] Chowdhury et.al., "Fingerprint recognitionsystem using hybrid matching techniques", Computer and Information Science, ACISInternational Conference, 2002

[5] Aleksander Kołcz, "Improved robustness of signature-based near-replica detection via lexicon randomization", Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2004

[6] Monika Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms", Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, August 2006

[7] Lu Lu and P. Wang, "Duplication Detection In News Articles Based on Big Data", 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), April 2009