Journal of Trends in Computer Science and Smart Technology (ISSN: 2582-4104)
www.irojournals.com/tcsst/

Winnowing vs Extended-Winnowing: A Comparative Analysis of Plagiarism Detection Algorithms

Shiva Shrestha[1], Sushan Shakya[2], Sandeep Gautam[3]

[1,3]Himalaya College of Engineering, Tribhuvan University, Kathmandu, Nepal

[2]Kathford International College of Engineering and Management, Tribhuvan University, Kathmandu, Nepal

Email: [1]shiv.shres25@gmail.com, [2]shakyasushan0@gmail.com, [3]macabresndp@gmail.com

Abstract

Plagiarism is the main problem in the digital world, as people use others' content without giving prior credit to the creator. Therefore, there should be proper and efficient algorithms to find plagiarized content on the Internet. This research proposes two algorithms: the winnowing algorithm and the extended winnowing algorithm. The winnowing algorithm can only calculate the similarity rate between documents, whereas the extended algorithm can mark the plagiarized text segment in the compared records along with their similarity rates. The similarity rate in both algorithms has been calculated using the Jaccard Coefficient. Although the extended algorithm is beneficial as it provides a text marking feature, it consumes more computation power, which is discussed in this study. There are research works done previously using this approach, but none has compared the algorithms' performance on small texts. Thus, this research utilizes the Twitter form of data to test these algorithms' performance, as it contains a maximum of 280 characters. The application proposed to detect plagiarism in tweets has been developed using Python as the backend and React as the front-end technology.

Keywords: Winnowing Algorithm, Extended Winnowing Algorithm, Jaccard Coefficient, Twitter, Python, React

DOI: https://doi.org/10.36548/jtcsst.2023.3.001
Received: 12.05.2023, received in revised form: 12.06.2023, accepted: 26.07.2023, published: 05.07.2023
© 2023 Inventive Research Organization. This is an open access article under the Creative Commons Attribution-NonCommercial International (CC BY-NC 4.0) License

1. Introduction

Plagiarism is the act of representing another person's thoughts or work as their own, with or without that person's consent, by incorporating it into their work without properly attributing the source. [1]. Plagiarism is most common in academic writing, content marketing, blogs, and many more. Plagiarizing others' creativity can prevent one from getting credit or appreciation for their effort and can make them frightened of sharing their ideas.

Nowadays, with the rapid development of the Internet, social media users are also increasing rapidly [2]. Social media usage is one of the most popular online activities. In 2021, over 4.26 billion people were using social media worldwide, a number projected to increase to almost six billion in 2027 [3]. With the increase in the number of social media users, millions of pieces of digital content are generated every day. The one who signs up for Facebook, Twitter, Instagram, and other social platforms can easily copy one authentic user's content, post it to their own, and go viral without acknowledgment. Considering the social media example, Twitter is one of the most widely used social media sites worldwide. With the increasing use of it, the act of using others' tweets as their own has become most common, and people are widely using this fashion to get more hype and become more popular in the social media world. It has created a feature called "Report Tweet," where the original author can report a case if someone seems to misuse his or her tweets [4].

Therefore, it is necessary to detect tweet plagiarism before reporting it. The algorithm for detecting plagiarism in tweets is an extended winnowing algorithm, which is a modified version of the winnowing algorithm that can mark copied text between the search tweet and tweets retrieved from the database. Implementing this, the user can easily detect how similar the tweets are and can aid in preventing plagiarism.

2. Related Works

Saul Schleimer et al., conducted research on the winnowing algorithm on 500,000 HTML pages in 2003 [5]. This algorithm extracts the document fingerprints by calculating the hashing values of each k-gram and selecting the minimum hash values in every defined window. Moreover, the research on the extended winnowing algorithm was done by Xuliang

DUAN et al., in 2017 on the PAN2013 corpus, which preserves the location of each hash value obtained after winnowing to mark plagiarism in documents [6].

In this research, Twitter datasets are used to compare these algorithms. However, these algorithms are tested on enormous documents containing many words. Furthermore, the efficacy of the application proposed in the work is tested on tweets that can contain up to 280 characters.

2.1 Winnowing Algorithm

The winnowing algorithm is a fingerprint-based method that is used to find the similarity between the documents. This algorithm generates the fingerprint based on the windows of the hashed value of the text. By eliminating unnecessary characters like punctuation, the winnowing algorithm has met the plagiarism algorithm's criteria for whitespace sensitivity[7]. The flowchart for extracting fingerprints from the text using this algorithm can be seen in Figure 1.

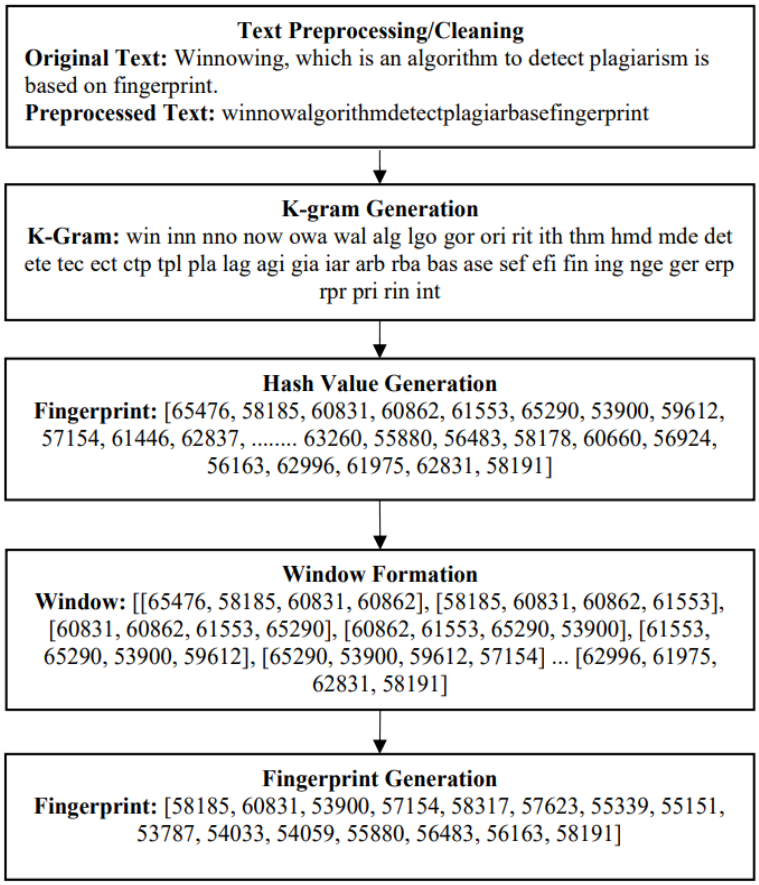

Figure 1. Winnowing Algorithm Fingerprints Generation Steps

2.1.1 Text Preprocessing/Cleaning

Text preprocessing is the first step of the winnowing algorithm. In this step, the original text is preprocessed to remove irrelevant characters. It includes lowercase conversion, stopword removal, punctuation removal, and number removal.

2.1.2 k-gram Generation

In a k-gram generation, the sequence of characters of length k is formed from the preprocessed text by extracting 'k' successive characters. The value of k can be any natural number based on the requirements [8].

2.1.3 Hash Value Generation

The rolling hash technique generates the hash value of each k-gram [9]. The following equation gives the rolling hash function:

$$H(c_1 \dots c_k) = c_1 * b^{(k-1)} + c_2 * b^{(k-2)} + c_3 * b^{(k-3)} + \cdots + c_{(k-1)} * b + c_k$$

Where 'c' is an ASCII (American Standard Code for Information) value, 'b' is the base-prime number, and 'k' is the number of characters.

2.1.4 Window Formation

The hash value thus obtained after applying the rolling hash function to each k-gram is subjected to several windows having a window size of w. The smallest value will be chosen from each of the windows, and if more than one smallest value is in a single window, then the rightmost value is selected [10].

2.1.5 Fingerprint Generation

Fingerprints are an array of the smallest value picked up from each window.

2.2 Extended Winnowing Algorithm

By extending the classic winnowing plagiarism detection algorithm, the extended winnowing algorithm can capture the location and length of text blocks while calculating the hash value. Fingerprints' location and length information can be used to find and mark

plagiarism text blocks in original documents. An extended version of this algorithm can be used to highlight the copied portions in a user interface instead of the winnowing algorithm, which only captures the fingerprints of the document. An efficient implementation of winnowing also needs to keep track of the position of the most recently selected fingerprint [5]. The winnowing algorithm process includes first text preprocessing followed by forming k-grams of length k. After that, those k-gram hash values are computed using the rolling hash function, a window of size is formed, and then the minimum hash is selected from that window as the document fingerprint.

2.2.1 Text Preprocessing

In the winnowing algorithm, the text preprocessing steps include converting text into lowercase, tokenizing, removing noise, and stemming. However, any preprocessing that could alter the length of the text will no longer be executed in extended winnowing, and the text clean-up procedure is implemented while processing the k-gram. [6]. Therefore, to cope with tweets' irrelevant characters, a list of more than 200 stop words, symbols, numbers, and punctuation was used for preprocessing before applying the algorithm. Apart from that, the stemming process, which changes the text length during preprocessing, will not be executed anymore.

2.2.2 Fingerprint Extraction

For marking the text segment in original documents, the fingerprint (hash value) should be extracted along with the location and length of the text segment in the array. Here, the text segment length is equal to the k-gram. The i^{th} k-gram text segment seg_i has useful k characters, as the meaningless characters are removed; so the length of the text segment in the original document is not less than k; therefore, $len_i > k$. The extended fingerprint is made up of three values [6].

$$h_i = [hash(seg_i), loc_i, len_i]$$

Here, loc_i is the start location of text segment seg_i in the original document and len_i is the source length of seg_i in the original document.

Algorithm 1 describes the process of calculating text hashes and their respective locations and lengths. The algorithm calculates text hashes by dividing the input text into k-grams, replacing stop words, and computing hash values for each k-gram. The resulting hash values, along with their locations and lengths, are stored in an array and returned as output.

The window is the primary step in the winnowing technique that helps classify hash values created to create fingerprints. The lowest hash value available in each window is chosen. The rightmost occurrence is chosen if there are multiple hashes with the same minimum value. Immediately each of the chosen hashes is saved as the document's fingerprints [5].

Algorithm 2 describes the process of getting fingerprints from the text. The algorithm calculates text fingerprints by sliding a window over hash values, selecting the minimum hash value within each window, and storing distinct minimum values in the fingerprint array. The resulting array represents the text fingerprints.

Algorithm 1. Calculating Text Hashes

***Input**: text in tweets, k-gram 'k', the base-prime number 'b'*

***Output**: hash values of text and locations*

```
Initialization: hashes ← array()
LOOP Process
for loc = 0 to Len(text) - k do
    length ← k
    k-gram ← text[loc:length] //get substring of text
    k-gram ← replace_stopwords(k-gram)
    while (Len (k-gram) < k) do
        length ← length + k – Len(k-gram)
        k-gram ← text[loc:length]
```

```
10          k-gram ← replace_stopwords(k-gram)
11          if (length > Len(text)) then
                length ← Len(text)
                break
12          end if
13      end while
14      hash ← hash_function(k-gram)
15      hashes.add (hash, loc, length)
16  end for
18  return hashes
```

Algorithm 2. Getting Text Fingerprints

Input: *hash values of text, k-gram 'k', window size 'w'*

Output*: fingerprint*

```
1   Initialization: current_min_index ← −1
2   Initialization: previous_min_index ← −1
3   Initialization: fingerprint ← array()
4   LOOP Process
5   for j = i to Len(hash) - w + 1 do
6       min hash ← hash[i][0]
        //select rightmost value if there are more than
```

```
        //one minimum value in a window
7       for j = i to i + w do
8           if (hash[j][0] <= min_hash) then
9               current_min_index ← j
10              min_hash = hash[j][0]
11          end if
12      end for
13      if (current_min_index ≠ previous_min_index) then
14          previous_min_index ← current_min_index
15          fingerprint.add(hash[current_min_index])
16      end if
17  end for
18  return fingerprint
```

2.2.3 Merging the Equal Fingerprints Indexes

The fingerprints obtained from the two documents have the hash value and their respective positions and lengths in the original documents. The Jaccard coefficient calculates the similarity rate between documents. To mark the plagiarized segment in tweets, a combination of the same hash values between the fingerprints of the tweets is needed to generate the hash index.

$$tweet_{index} = \{tloc, tlen, sloc, slen\}$$

Where, *tloc* and *tlen* are the location and length of copied segments of the tweet entered in the application to detect plagiarism, while *sloc* and *slen* are the copied segments of the tweet obtained from the dataset.

2.2.4 Merging the Adjacent Hash Index

The hash index generated between documents can be merged if the interval is less than a certain threshold. Algorithm 3 describes the process of merging adjacent tweet indexes. As the algorithm aims to merge tweet indexes based on a specified threshold, it starts by sorting the tweet index by tweet location and then iterates through the indexes. If the interval condition is satisfied between the current and previous indexes, they are merged by updating the length values. Finally, the merged index is returned as output.

Algorithm 3. Merging Tweets Indexes

Input: *tweet index 'ti', spacer 's'*

Output*: merged index 'mi'*

```
1   //merge tweets indexes if the interval between them
    // is less than specified threshold 'spacer'
2   ti ← sort_index_by_tloc(ti)
3   LOOP Process
    //loop through the last index
4   for i = Len(ti) − 1 to 0 do
5       if (ti[i−1][tloc]+ti[i−1][tlen]+spacer ≥ ti[i][tloc] and
        ti[i−1][sloc]+
            ti[i−1][slen]+spacer ≥ ti[i][sloc]) then
6           tlen ← ti[i][tloc]−ti[i−1][tloc]+ti[i][tlen]
7           slen ← ti[i][sloc]−ti[i−1][sloc]+ ti[i][slen]
8           ti[i−1][tlen] ← tlen
9           ti[i−1][slen] ← slen
```

```
10          delete_index(ti[i]) //delete the last index if merging successful
11        end if
12      end for
13    return mi ← ti
```

2.2.5 Jaccard Coefficient

Grove Karl Gilbert created the Jaccard similarity coefficient, often known as the Jaccard index, in 1884 to assess the similarity and diversity of sample sets. This coefficient is utilised in this study to compare two documents and determine how similar they are. The Jaccard coefficient has several variations, and its values typically range from 0 to 1, with 0 denoting no overlap and 1 denoting total overlap between the sets [12].

$$Jaccard(X,Y) = \frac{||X \cap Y||}{||X \cup Y||}$$

Where, X and Y are two sets of document fingerprints. As a result, the Jaccard distance may be erroneous in practise. For instance, if A is a proper subset of B but A is completely plagiarised, the Jaccard distance will be less than 1 [13].

Therefore, to consider this situation and improve the precision and recall ratio, the set having the minimum number of document fingerprints will be used in the denominator section instead of the union of those sets.

$$Normalized\ Jaccard(X,Y) = \frac{||X \cap Y||}{min||X,\ Y||}$$

3. Application Overview

The application designed for plagiarism detection can detect plagiarism using two algorithms: winnowing and extended winnowing. Extended winnowing is an extended version of the classic winnowing algorithm. The application consists of mainly two parts: the frontend

and the backend. The frontend part of the system is developed using the JavaScript frontend library called React, and Python FastAPI is used under the hood for the backend part.

Users are provided with the functionality of entering tweets for plagiarism checks. When the tweet is entered, both the winnowing and extended winnowing algorithms are applied to the entered text and the tweet datasets, to detect the similarity between the texts. The winnowing algorithm returns only the similarity rates, while the extended version can return the similarity rate and the text positions. Figure 2 shows the result of the search, highlighting the plagiarized text and the similarity rates.

Figure 2. User Entering Tweets for Plagiarism Detection

The application allows for different groups of datasets that contain tweets in numbers like 100, 200, 400, 800, 3200, and 6400. The execution time of the algorithms depends on the selection of the dataset group. For 100 as a dataset group, it takes less running time than for 6400 as a dataset group. Based on that, users can select the dataset group according to their use cases to reduce the comparison time while checking for plagiarism, as seen in Figure 3.

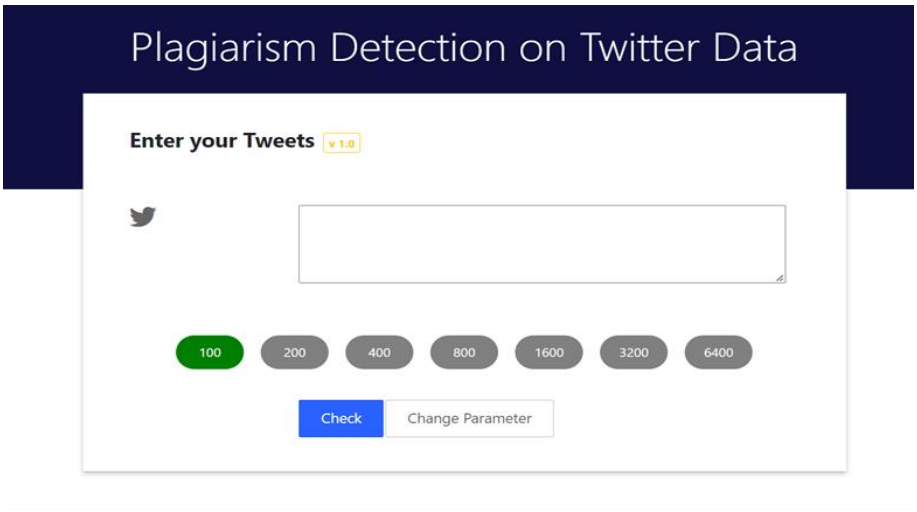

Figure 3. User Selected 100 as the Dataset Group

The application also supports changing algorithm configurations for experimenting with the k-gram, window size, base prime value, and spacer, as this plays an essential role in the accuracy of the algorithm. For example, changing the k-gram from 5 to 3 and the window size from 4 to 3 might have different results based on the input size of the text. Regarding this scenario, the application is designed to cope with the changeable configurations for algorithms based on which users can get the best result on checking plagiarism, as shown in Figure 4.

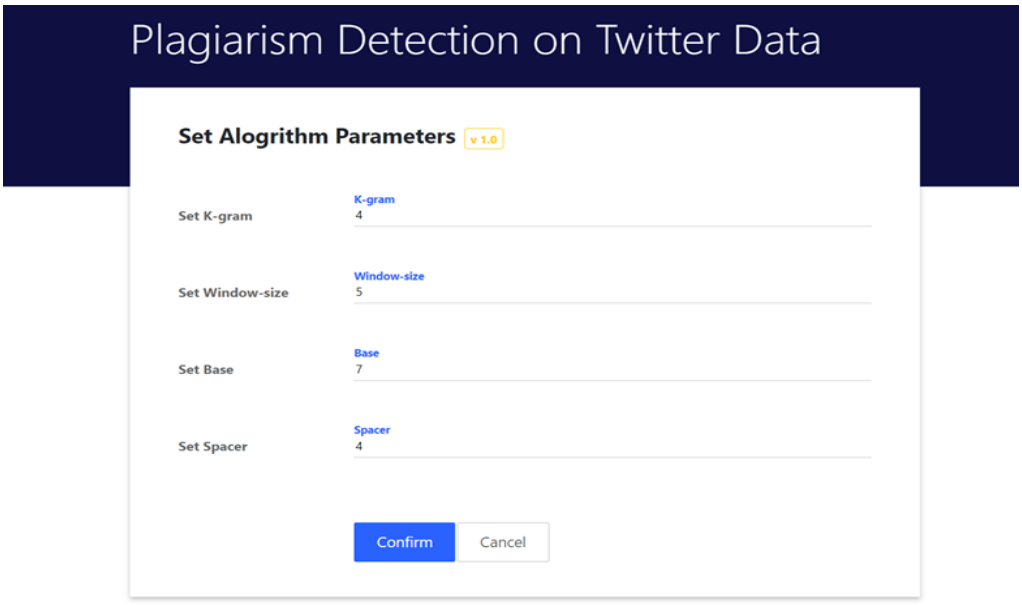

Figure 4. Example of Configuring Algorithm Parameters

4. Result and Discussion

4.1 Parameters Tuning

The application for detecting plagiarism in tweets depends on the parameters like k-gram, window size, and base-prime number set to the algorithm. These parameters play a vital role in the accuracy of the algorithm. For tuning those parameters according to the Twitter data, several test cases were implemented for winnowing and extended winnowing algorithms, as they share the same properties. The use of k-gram in the application is the basis for the smallest subset comparison desired by the user to detect substring parity [14].

Twitter tweets have a maximum of up to 280 characters. The range of k-grams between 3 and 6 was evaluated to select the best parameter, which entails the smallest substring threshold needed to find between tweets for calculating the similarity rate. Table 1 shows the different combinations of parameters tested for both algorithms.

Table 1. Parameter Combination

Parameter combination	k-gram length	Window size	Base prime number
PC1	3	4	5
PC2	3	6	5
PC3	5	4	7
PC4	6	5	11
PC5	4	6	13

The algorithms' accuracy in different combinations are compared for two tweets. The XYZ plagiarism checker tool was used, and their similarity scores were used as a reference to select the best combination of parameters [8]. Table 2 below shows the five random

comparisons of tweets along with their similarity scores, where WA denotes the Winnowing Algorithm, and EX denotes the Extended winnowing algorithm.

Table 2. Test Cases for Selecting Best Algorithms Parameters

Original tweet	Compared tweet	XYZ Checker (%)	PC1(%)		PC2(%)		PC3(%)		PC4(%)		PC5(%)	
			WA	EX	WA	EX	WA	EX	WA	EX	WA	EX
A	A*	57	64	68	68	65	59	61	62	63	69	68
B	B*	46	54	58	53	53	42	44	40	44	45	48
C	C*	50	45	57	47	51	40	49	35	46	46	51
D	D*	72	69	77	67	75	62	69	62	68	71	76
E	E*	53	62	68	55	67	56	60	58	59	59	58

The color represents the minimum differences between the XYZ tool and the proposed algorithms. Here, green indicates the minimum difference between the tool and the winnowing algorithm in a particular parameter combination, whereas yellow indicates the minimum difference between the tool and the extended winnowing algorithm. It is clearly seen that in the PC3 combination (k-gram: 5, window-size: 4, and base prime number: 7), similarity rates were more proportional to the XYZ plagiarism tool than in other combinations. The average similarity rates of tweets over different combinations are shown in Figure 5.

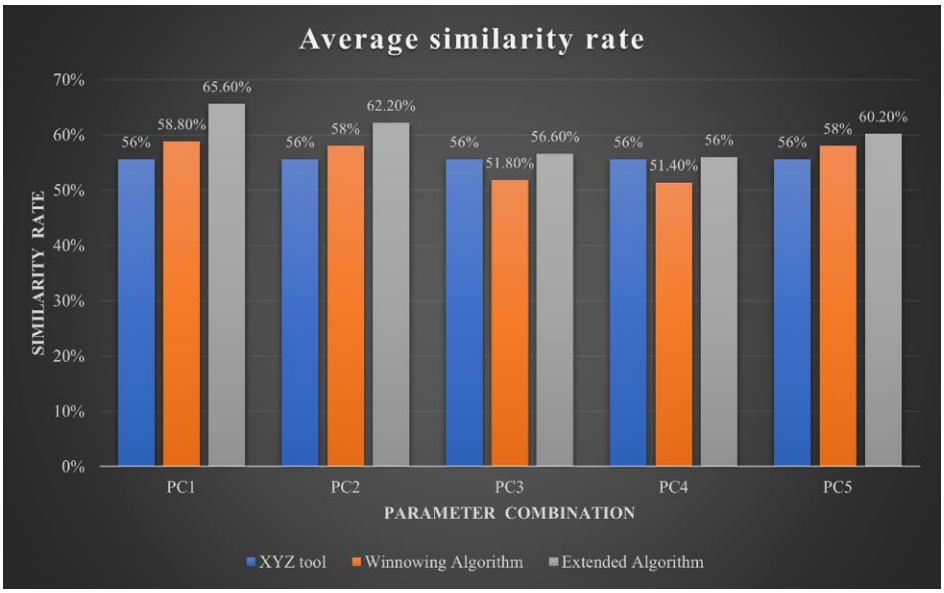

Figure 5. Bar Chart of Average Similarity Rate of 5 Tweets

From Figure 5, the similarity rate calculated using PC3 by the tool and the extended winnowing algorithm are equal, and there is a slight change in the rate for the winnowing algorithm. By considering these data, the application used PC3 as the configuration attribute, which is suitable for both algorithms, i.e., k-gram of 5, the window size of 4, and base prime number of 7.

b. Evaluation of Running Time

For evaluating the running time between the winnowing and extended winnowing algorithms, the dataset of over 80,000 tweets from Kaggle is used [15]. The dataset was divided into ten groups of tweets ranging from 100 to 80,000. The 20 random tweets extracted from the dataset were compared against each group of datasets using the algorithms implemented in the Python programming language. The algorithms were run on a PC with an Intel (R) Core (TM) i7-10750H CPU @ 2.60GHz 2.59 GHz and 16.0 GB of RAM with Windows 10 as the operating system. The average time required by algorithms on 20 random tweets against ten groups of datasets can be seen in Table 3.

The algorithms' execution times are directly proportional to the data compared to the datasets, as seen in Figure 6. The execution time for both algorithms is proportional to the medium-sized datasets having less than 3200 tweets. However, for the larger datasets, there were drastic changes in the execution time for the extended winnowing algorithm compared to

the winnowing algorithm. From the Table 3 data, on average, the winnowing algorithm is three times faster than the extended winnowing algorithm.

Table 3. Average Execution Time of Algorithms

Dataset Group	Average execution time of 20 random tweets (in seconds)	
	Winnowing Algorithm	**Extended Winnowing Algorithm**
100	0.2627	0.6147
200	0.4214	1.1468
400	0.704	2.0706
800	1.0019	3.204
1600	1.8379	5.9375
3200	3.8244	12.3335
6400	8.0925	26.4313
12800	16.4777	53.8467
25600	32.5345	108.8037
51200	64.5361	216.6903
80000	105.9505	365.6818

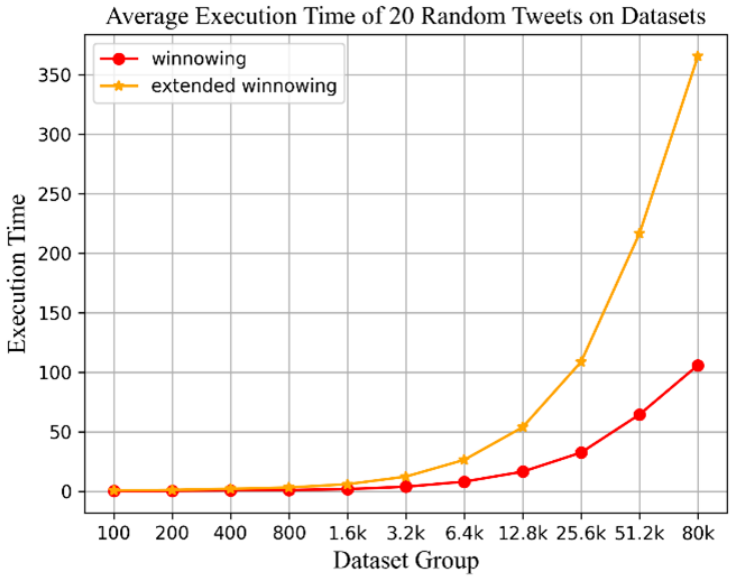

Figure 6. Line Graph of Algorithms Execution Time

5. Conclusion

The research for testing the performance of winnowing and extended winnowing algorithm has been successfully conducted on Twitter data. This research shows that the extended algorithm's execution time lags the winnowing algorithm by three times while comparing tweets with massive datasets. Furthermore, the algorithms' best configurations, like k-gram, window size, and base prime number, were predetermined to get the best accuracy. Finally, this study shows that the extended algorithm is more expensive to compute than winnowing, though it has the additional benefit of marking the plagiarized text. Although the algorithms discussed in this work may not be perfect, there is always room for optimization according to the use cases. For example, the application uses a Python list as a container for storing hash values. In that case, the NumPy array can be used as a container for hash values, which uses less memory and is faster than a Python list. The application developed cannot compare the tweet's plagiarism with live Twitter data, which can be considered a future enhancement using the Twitter API.

Declarations

Conflict of Interest

The authors declare that they have no conflict of interest.

Funding

Not applicable.

References

[1] Plagiarism | University of Oxford. (n.d.). Retrieved from https://www.ox.ac.uk/students/academic/guidance/skills/plagiarism/

[2] Ulinnuha, N., Thohir, M., Novitasari, D. C. R., Asyhar, A. H., & Arifin, A. Z. (2018). Implementation of winnowing algorithm for document plagiarism detection. Proceeding of EECSI, 631-636.

[3] Number of worldwide social network users 2027 | Statista. (2023, February 13). Retrieved from https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/]

[4] Mason, S., & Singh, L. (2022). Reporting and discoverability of "Tweets" quoted in published scholarship: current practice and ethical implications. Research Ethics, 18(2), 93–113. https://doi.org/10.1177/17470161221076948

[5] Schleimer, S., Wilkerson, D. S., & Aiken, A. (2003, June). Winnowing: local algorithms for document fingerprinting. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (pp. 76-85).

[6] Duan, X., Wang, M., & Mu, J. (2017). A plagiarism detection algorithm based on extended winnowing. In MATEC Web of Conferences (Vol. 128, p. 02019). EDP Sciences.

[7] Haryadi, D. (2012). Implementasi Algoritma Winnowing dengan Tahapan Preprocessing pada Aplikasi Pendeteksi Plagiarisme Dokumen Teks. Undergraduate, Universitas Multimedia Nusantara.

[8] Shrestha, S., Gautam, S., Sharma, K. & Bhandari, A. (2023). Winnowing Algorithm: A Powerful Tool for Identifying Plagiarism in Assignments. Journal of Trends in Computer Science and Smart Technology, 5(2), 168-189. doi:10.36548/jtcsst.2023.2.006

[9] H. Jiang and S. -J. Lin, "A Rolling Hash Algorithm and the Implementation to LZ4 Data Compression," in IEEE Access, vol. 8, pp. 35529-35534, 2020, doi: 10.1109/ACCESS.2020.2974489.

[10] Hasan, E. G., Wicaksana, A., & Hansun, S. (2018, June). The implementation of winnowing algorithm for plagiarism detection in Moodle-based e-learning. In 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS) (pp. 321-325). IEEE.

[11] Ridho, M. (2013). Rancang Bangun Aplikasi Pendeteksi Penjiplakan Dokumen Menggunakan Algoritma Biword Winnowing (Doctoral dissertation, UNIVERSITAS ISLAM NEGERI SULTAN SYARIEF KASIM RIAU).

[12] Arnaboldi, V., Passarella, A., Conti, M., & Dunbar, R. I. (2015). Online social networks: human cognitive constraints in Facebook and Twitter personal graphs. Elsevier.

[13] Jaccard Similarity. (n.d.). Retrieved from https://www.learndatasci.com/glossary/jaccard-similarity/

[14] Tung, K. T., Hung, N. D., & Hanh, L. T. M. (2015). A Comparison of Algorithms used to measure the Similarity between two documents. Int. J. Adv. Res. Comput. Eng. Technol., no.

[15] S. T. Demirel and R. Das, "Software requirement analysis: Research challenges and technical approaches," 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey, 2018, pp. 1-6, doi: 10.1109/ISDFS.2018.8355322

Author's biography

Shiva Shrestha is currently working as a software engineer at Verisk Analytics. He received his bachelors in computer engineering from Tribhuvan University in 2021. His area of research includes natural language processing, cloud computing, and decoding machine learning algorithms.

Sushan Shakya is currently working as a software engineer at Verisk. He received his bachelors in computer engineering from Tribhuvan University in 2021. His area of research includes data engineering, data science, and cloud computing.

Sandeep Gautam is pursuing his MBA-IT in the Faculty of Management at Tribhuvan University, Nepal. He received his Bachelor of Computer Engineering from Tribhuvan University in 2021. He is currently engaged in SOMTUVmag.