Full-Privacy Secured Search Engine Empowered by Efficient Genome-Mapping Algorithms

Yuan-Yu Chang ⊗, Sheng-Tang Wong ⊗, Emmanuel O. Salawu ⊗, Ming-Hsuan Liao ⊗, Jui-Hung Hung ⊗, and Lee-Wei Yang ⊗

Abstract—Since the 90s, keyword-based search engines have been the only option for people to locate relevant web content through a simple query comprising one to a few keywords. These engines, whether free or paid, retained users’ search queries and preferences, often to deliver targeted ads. Additionally, user-uploaded articles for plagiarism detection can further be stored as part of service providers’ expanding databases for profit. Essentially, users could not search without exposing their queries to these providers. We present a new solution here: a method for searching the internet using a full article as a query without disclosing the content. Our Sapiens Aperio Veritas Engine (S.A.V.E.) uses an encoding scheme and an FM-index search, borrowed from next-generation human genome sequencing. Each word in a user’s query is transformed into one of 12 “amino acids” to create a pseudo-biological sequence (PBS) on the user’s device. Plagiarism checks are done by users submitting their locally created PBSs to our cloud service. This detects identical content in our database, which includes all English and Chinese Wikipedia articles and Open Access journals up to April 2021. PBSs, longer than 12 “amino acids”, show accurate results with less than 0.8% false positives. Performance-wise, S.A.V.E. runs at a similar genome-mapping speed as Bowtie and is >5 orders faster than BLAST. With both standard and private modes, S.A.V.E. offers a revolutionary, privacy-first search and plagiarism check system. We believe this sets an exciting precedent for future search engines prioritizing user confidentiality. S.A.V.E. can be accessed at https://dyn.life.nthu.edu.tw/SAVE/.

Index Terms—Encoding, FM-Index, logistic regression, next-generation sequencing, plagiarism, privacy, biological sequence.

I. INTRODUCTION

THE worldwide web has been navigated and accessed by search engines since the middle of 90s. Since then, people used to “google” the web to locate the needed information instead of always bookmarking them. The indispensability of search engines comes with a price of compromised personal privacy where one’s query histories, browsing habits, and other personal info are collected for improved search-result relevance, while this personalization also accompanies targeted ads delivery or sales of users’ data to third parties [1]. A study suggested that a user’s query can be stored by the search engine company for 18 months [2]. There are search engines that operate by storing less personal information [3], [4] or hiding users’ IPs through thousands of relays [5]. These search methods, however, have not offered guaranteed security and intrinsic privacy for users.

Free access to online information also brought reckless misuse and unauthorized copies from internet sources. Full-article-query-based (FAQ-based) search engines have been used as a plagiarism detection tool [6] since the beginning of this century. Plagiarism constitutes the re-use of other people’s language, ideas, and expressions in one’s work without proper acknowledgment [7], usually through a direct copy-and-paste [8], which surely undermines educational efforts and originality of scientific research [8], [9]. To mitigate such an issue, a number of algorithms and software have been developed, where pre-built databases and the use of search engine APIs are two primary searching strategies. The former builds databases in advance using academic journals, magazines, news, blogs, and users’ articles, for plagiarism detection. For instance, Turnitin [10], Grammarly, Duplichecker and eTBLAST [11] operate with pre-built databases. On the other hand, search-engine-based
plagiarism detection tools, such as Plagiarisma and SNITCH [12], directly query search engines with the content of users’ articles instead of building a local database. Both types of applications require users to upload their documents to specified remote servers. The queries are compared with what is stored in the database or with results returned from popular search engines. With both, users always have limited control of their submitted content; data leakage can happen during transmission or under malicious attacks. Also, service providers could reuse uploaded articles for expanding their current databases for profit [13] or for commercial uses.

To allow users to have a privacy protection while still conducting full-text search, a framework for plagiarism detection was proposed to generate a fingerprint of the suspect file and comparing it with fingerprints of available files from the remote database to identify potential plagiarism [14]. Bruestle et al. proposed a Prefix Burrows-Wheeler Transformation (PBWT) method that allows for the efficient encoding and decoding of data sets, while also allowing for fast and memory-efficient data operations such as searching and sorting [15]. However, PBWT can only detect if the query text exists in the databases but not identify the accurate locations of the detected texts in the databases, which is therefore not suitable for full-text plagiarism detection.

These methods are either hard to trace the locations of plagiarism or inefficient to be practical. To address this gap and offer a search service with 100% privacy protection of a given query, either for expertise matching [11] or plagiarism detection, we have developed the Sapiens Aperio Veritas Engine (S.A.V.E.), as a plagiarism detection solution conducting plagiarism detection with full protection of user-uploaded content. S.A.V.E. takes irreversible encryption of query articles or strings for a fast database search. The irreversible encryption mapped each word or character from an article into an “amino acid”, which is a single character (see Fig. 4 for example). The pseudo-biological sequences (PBS) are composed of sets of aforementioned encrypted characters, which is analogous to biological sequences, where a long article can be a pseudo “genome”. The search for this genome can therefore lend itself to any state-of-the-art genome mapping algorithm - in our case, the FM-indexing. FM-index was first introduced in 2000 [16] as an opportunistic data structure based on the Burrows-Wheeler transform (BWT) [17] that compresses the input text for subsequent efficient substring search. The advantages of the FM-index make it suitable for DNA read mappings, which led to the rise of multiple alignment tools based on FM-index, such as Bowtie [18], BWA [19], and SOAP2 [20]. The FM-index-based approaches allow an efficient query, which takes no more than 5 seconds for S.A.V.E. to detect plagiarism among English and Chinese Wikipedia pages, PubMed abstracts and open access articles provided with an article with thousands of words. Given the irreversible encryption (ciphertext), the exact match could refer to more than one version of the plaintext, producing false positives. In this report, we minimized the chance of obtaining false positives by studying the sensitivity of queried PBSs as a function of PBSs in different lengths comprising different numbers of constituent amino acids (see below). We also found our implementation enjoys a similar running speed as Bowtie’s while being >500000 times faster than BLAST.

II. RESULTS

A. Plagiarism Detection Framework

The plagiarism detection framework of S.A.V.E. provides the highest privacy protection by not revealing the original content of the query document. S.A.V.E. uses destructive encryption to encrypt a document into a long pseudo-biological sequence (PBS) comprising a fixed number of characters, as an irreversible ciphertext. For instance, in our current implementation, any to-be-uploaded document(s), written in any language, can be translated into a long, irreversible ciphertext comprising 12 characters (i.e., A, C, D, E, G, H, I, K, L, N, Q and R, where a = 12) such as “NAGNDDHHIGQLCKNNAAQKKKDQARDD-CIGLNNDCQQNRRDDEQAREA…” by our local software in users’ personal computer before being uploaded (to S.A.V.E.). S.A.V.E. takes both plaintext and ciphertext where the former is not discussed in this article.

In the latter scheme, S.A.V.E. separates the encryption function from the searching function to prevent the exposure of users’ sensitive content, where the encryption is performed with a local device, and the searching is performed in the cloud. As shown in Fig. 1, users can encode their documents to query by an offline software we provide, which can be downloaded from https://dyn.life.nthu.edu.tw/SAVE/download. Next, users upload the encrypted documents to S.A.V.E. for plagiarism detection. The degenerate encoding nature makes data stealing, either in transmission or on the server, impossible. In the S.A.V.E. database (top of the grey area in Fig. 1(b)), web content is encoded in the same way as by the local software. The search and detection are carried out by a modified version of FM-index used in next generation sequencing (see Methods). After the search is finished, users can download the metadata of the search results in JSON format and generate the report with S.A.V.E. local software alongside the original documents. After the JSON-based meta-files are downloaded, all the operations to decipher the plagiarized loci in plain text can be done offline. At the server end, two different pathways of plagiarism detection are implemented depending on the number of uploaded documents. When users upload a single encrypted document, S.A.V.E. searches its PBS database containing encrypted web content that currently includes all the English and Mandarin Wikipedia pages and open-access journal articles (Fig. 1(b)). If users zip multiple encrypted documents into a single file and then upload the zip file to S.A.V.E., S.A.V.E. would first build a temporary database based on these uploaded PBSs and then check their mutual plagiarism in a one-to-multiple manner (also called “pairwise comparison”, Fig. 1(a)). After the pairwise comparison step, users can also select all or a subset of the documents for the search against encrypted web content using the pre-built S.A.V.E. database as described in Fig. 1(b) (almost the same as Fig. 1(a) but is for a single document). Both pathways result in a metafile (in Python JSON format) to be downloaded into a local folder in users’ local computers, which contains local mapping files and the original documents. The S.A.V.E.-returned metafile...
Fig. 1. Plagiarism detection framework. (a) The pairwise comparison for multiple documents. Users can encrypt their confidential documents on their local computers before uploading them to the S.A.V.E. server; by doing this, the sensitive content of the documents is not revealed to search engine providers and is free from the risk of being intercepted during the transmission. On the server-side (in the grey area), the uploaded encrypted files are first used to build a temporary database followed by a pairwise comparison to generate a JSON file. With the JSON metadata and the original file stored on the user’s local device, S.A.V.E. allows users to browse the result web page, highlighting the locations in the original documents copied from each other, in their local device (Fig. 3(a), bottom). (b) PBS encoded from a single document. In this pathway, S.A.V.E. detects plagiarism of the document against pre-compiled PBSs encoded from web content (supposedly the whole web) instead of performing the pairwise comparison. Users can browse their results directly on the S.A.V.E. website (Fig. 2(b)).

TABLE I

| Length of the k-mer string | false positive rate (%) |
|---------------------------|-------------------------|
|                           | number of characters used (a) |
| 8                         | 12                      | 14           | 16          |
| 8                         | 100.00                  | 100.00       | 99.65       | 87.87       |
| 10                        | 78.71                   | 16.28        | 4.39        | 1.99        |
| 12                        | 4.49                    | 0.76         | 0.55        | 0.58        |
| 14                        | 1.26                    | 0.49         | 0.43        | 0.46        |
| 16                        | 1.02                    | 0.40         | 0.36        | 0.37        |

In-private searches using PBSs may return us with results where the discovered plaintexts are not the same as those originally encoded. To understand at what false positive rate our search algorithm has as a function of different lengths (k = 8, 10, 12, 14 or 16) of queries comprising a selection of a (a = 8, 12, 14 or 16) characters ("amino acids"), we conducted the following assessment.

To evaluate the false positive rate, the chance to find a piece of plaintext in the database that is not the original plaintext, encoded into the query PBS, we investigate how likely this happens when the search starts from a k-mer PBS, where k = 8, 10, 12, 14 or 16. Note that in each position of a k-mer PBS stands one of a (a = 8, 12, 14 or 16) characters. Since a k-mer PBS always corresponds to a piece of plaintext comprising k (human) words, we transform the problem into a space spanned by all the k-word segments in the Chinese and English Wikipedia pages and then examine the chance that different k-word segments correspond to the same PBS (as if our users search an encoded k-word plaintext and then ask what the chance for our S.A.V.E. engine to return results containing plaintexts that are not the originally encoded plaintext). We consider only unique k-word strings to avoid the double counting. In 9 266 370 827 unique 12-word strings (k = 12; “unique” means that a given 12-word segment only appears once in all the English and Chinese Wikipedia pages) when a = 12, there are 70 352 323 such strings encoded into a PBS that other strings are also encoded into, hence a false positive rate of 70 352 323 / 9 266 370 827 = 0.0076. Results of other k-word strings can be found in Table I.

When the length is 8, the false positive rate can be as high as 87.87% even if 16 characters (a = 16) are used. As the length of PBS grows, length 10 with 16 characters can already reduce the false-positive rate to <2.00%. No larger than 5.00% false positive rate can be found for the PBSs longer than 12, while 12-mer PBSs with a = 12 characters have a small false-positive rate <0.8%. To consider detection sensitivity (the shorter k the better), high security due to encoding degeneracy (the smaller a the better) and false-positive rates (the larger the k and a are, the better), we therefore chose a = 12 and k = 12 as our encoding and sensitivity scheme for S.A.V.E. Duplicates less than k = 12
are not reported by our software and long stretches of duplicated texts are detected as overlaps of many 12-mer PBSs.

On the other hand, “false negative rate” here would mean a queried PBS “Q”, encoded from the plaintext “A”, did not match any PBS stored in the database, while there are PBSs other than “Q” encoding the same plaintext “A”. Practically, we did not find any occurrence of false negatives in our study. Theoretically, false negative would not occur because identical plain texts encode the same PBSs (one word is mapped to a specific PBS character according to Fig. 4), and via FM-index, the same PBS will definitely be found in the search. However, it is possible to have false negatives when a 12-mer comprises solely “common words”, defined in Table S1. However, the chance to have 12-mer comprising all common words is 0.0212 (0.02 is the average occurrence of 10 most used words in English; see the footnote in Table S1); considering that we have 9266370827 unique 12-word PBS strings, there are 0.0212 × 9266370827 = 3.77 × 10⁻¹¹ such strings, which minimizes the false negative concern given the current size of our database.

C. Assess the Precision of S.A.V.E. in Real-World Educational Scene

Over the past six years, we evaluated the efficacy of the educational tool, S.A.V.E., in a class that includes introduction of research integrity. Among thousands of assignments, we selected 3 midterm/final assignments totaling 554 deidentified student reports (Supplementary Methods). These reports were written in response to research talks given by domestic and international clinicians and academicians, and were examined for potential pairwise plagiarism (i.e., check whether one report copies another among this collection of 554 reports) and internet plagiarism. In the pairwise comparison, the precision, defined as the (number of matched plain-text words) divided by (the word count of the plain text that encodes S.A.V.E.-found PBS match), was found to be 96.8% (Figure S1).

This high precision is critical for education, where false positives in plagiarism detection (the plagiarism detection tool suggests an individual plagiarized (s)he actually did not) are more detrimental than false negatives. When using S.A.V.E. in privacy mode to search our database (Fig. 1(b)), we observed a precision of 93.96% in 6 homework involving plagiarism out of a randomly selected 139 homework (see Fig. 2 and Table S2). This real-world data underscores S.A.V.E.’s use in detecting copied content from sources including Wikipedia pages or Open Access Journals.

D. The Encoding Scheme Reduces the Size of S.A.V.E. Databases

The destructive encryption algorithm of S.A.V.E. is not only irreversible but also compressed, where a word (e.g., Utopia, CRISPR-Cas9, 新 or は) is encoded into one alphabet which certainly reduces the size of a document tremendously, although the application of the herein introduced FM-index alignment algorithm for plagiarism detection requires extra reference tables (see Fig. 5), leading to a slight increase (less than 1.8-fold) in the size (Table II). Table II shows that our FM-indexed PBSs with needed reference tables are about 1/6 of the size of original web content (146.1 GB, considering the text only). The encryption of words for phonemic western languages (e.g., English, French or Spanish etc.) resulted in a bigger compression ratio (e.g., 5.9 for English Wiki pages), while the ratio for non-alphabetic eastern language (e.g., Chinese, Korean, Japanese Kanji or Tamil etc.) is understandably smaller (e.g., 2.1 for Chinese Wiki pages) because one eastern “character” (e.g., 字) is one word.

E. Logistic Regression Model to Remove References in the Plaintext

A bibliography or list of references is the source of false positives either in plaintext or when encoded into PBSs. This is because a listed reference is definitely the same or very similar to what can be found online. In a normal report, journal article or thesis, references are listed to acknowledge the sources of our background introduction, employed methodologies and arguments that corroborate our results. These references in user-uploaded articles, encoded into PBSs, should not be considered part of plagiarism, and therefore their removal prior to the
PBS-encoding and upload is essential to reduce false positives. Since the running environment may be in users' personal computers, the compatibility and portability of the reference detector are important. We used a logistic regression model to identify the lines in the article that belong to the reference and those that are not. The training and testing sets contain 100 journal papers (containing ideal, well-formatted references) and 200 student reports (containing unideal and often poorly formatted references), where symbols such as ‘;’, ‘-’, ‘(’, ‘)’ etc. and four-digit numbers, representing years, are used as training features (see details in METHODS). The performance of the trained model reaches 0.97 area under the curve (AUC) of the receiver operating characteristic (ROC) curve, where unity is the highest possible value and 0.5 is the lowest. It is worth noting that the reference “lines” do not have to be at the end of the article for correct detection, considering the situation where figures or supporting information can be listed after the references, since reference position is not one of the training features in the logistic model.

F. Speed Analysis

We compared the execution time of our FM-index implementation with Bowtie, a popular FM-index-based NGS aligner, and BLAST [21], a classic bioinformatics aligner. FM-index is the most popular algorithm in bioinformatics and is the most efficient algorithm in string matching, of which the time complexity is O(m), where m is the length of queries.

To fairly assess these tools, we generated genomes and queries selected randomly from the character set and each of the queries is sampled indiscriminately from the corresponding genome.

The results showed that the throughput of our implementation was 147.6 million bases per second, Bowtie was 73.33 million bases per second, and BLAST was 291 bases per second, which is about a half-million times slower than our current implementation. However, after the FM-index search, Bowtie performed local alignment that our implementation did not, which partly explains why Bowtie’s speed was twice lower than ours. The efficiency of our implementation was a half-million times better than BLAST, as one would expect - FM-index is intrinsically faster than non-FM-index algorithm.

G. Local Application, S.A.V.E. Database and Interfaces

To use S.A.V.E. to detect plagiarism in the privacy-protective (private) mode, one needs to download the local software from https://dyn.life.nthu.edu.tw/SAVE/download, which allows one to encrypt her/his document into a PBS for the subsequent upload and detection. When multiple files need to be detected, users can first zip the documents into a single zip file and our local software would then unzip the file, extract the text from each document, convert each document into a file containing the corresponding PBS, and then zips the resulting PBSs back into a zipped file again for subsequent upload and detection (see Introduction videos in https://dyn.life.nthu.edu.tw/SAVE/static/video/video.html). Alternatively, one can also copy the PBS encoded from a document and paste it into the search bar of S.A.V.E.’s front page.

When a zipped file containing multiple PBS files is uploaded (Fig. 1(a)), all the PBSs will first be pairwise compared to identify mutually plagiarized PBSs (files); in other words, S.A.V.E. server examines which PBS files contain identical PBS stretches, longer than 12 encoded alphabets in length. One report is returned for each PBS (file). In this html-formatted report, the PBS file of interest is present together with the files that plagiarize this file (share identical PBS stretches) as well as the location where plagiarism occurs (Fig. 3(a), top). At this stage, only PBSs are present, not all the corresponding plaintexts that are not uploaded to the server in the first place. Currently, S.A.V.E. allows users to download the metadata, which can be used to map the original plaintext to plagiarized parts of the PBS through referencing locally stored original documents. Users’ local folders have had the mapped location information between the original text and the PBS into which it is encoded, while the downloaded metafile contains the information on which parts of the uploaded PBS have the plagiarized sources in the plaintext format. With the metadata, users can render the report on their local devices (Fig. 3(a), bottom).

Also, after the pairwise comparison finished, all or selection of these PBS files can be marked for a database search to identify plagiarism from online sources. Detection result for each PBS query is presented in the same way as that obtained when a single PBS is being uploaded (Figs. 1(b) and 3(b)) where plagiarized plaintexts in Wikipedia pages or Open Access journal articles are highlighted (left, Fig. 3(b)) side by side with the corresponding PBSs in the query (right, Fig. 3(b)).

| Database                  | Capacity (GB) | compression ratio* |
|---------------------------|---------------|--------------------|
|                           | Raw | PBS | FM-Index |
| Pubmed OA Commercial†     | 58.7 | 5.9 | 9.8 | 6.0 |
| Pubmed OA Noncommercial†  | 24.2 | 2.4 | 4.0 | 6.1 |
| Pubmed Abstract           | 45.6 | 4.6 | 7.4 | 6.2 |
| Wiki English              | 16.0 | 1.7 | 2.7 | 5.9 |
| Wiki Chinese              | 1.6  | 0.5 | 0.8 | 2.1 |
| Total                     | 146.1| 15.1| 24.7| 5.9 |

* Compression Ratio = size of raw data / size of the index;
† Pubmed OA stands for the PMC open access journal article where the content that can be used for commercial purposes is classified as "Commercial"; "Noncommercial" otherwise.
The length of the longest verbatim copy is reported along with how much percentage of the uploaded PBS can be found duplicated in the two files. In the demonstrated case, founder.docx has mutual plagiarism and (2) the locations of duplicated PBS segments are displayed on the right side. The pairwise comparison results are rendered in users’ local devices as the two files are simultaneously displayed on both sides. The pairwise comparison results are rendered in users’ local devices as the two files are simultaneously displayed on both sides. Corresponding texts on both sides are simultaneously displayed and dynamically highlighted when the cursor hovers on the relevant texts.

The compressed nature of our encoding scheme also promises a reduced storage space (and therefore the cost), while our search scheme easily allows a search parallelization and database expansion. For instance, sBWT, using Schindler transform, is readily deployable using GPU-accelerated hardware solution [23]. For the false positive issue, even if a longer stretch of PBS is used, the current implementation could have an intrinsic false positive rate of 1/a where a is the number of constituents “amino acids” (see Table II). It is inevitable that similar sentences can be downloaded for resolving the exact locations of the matched text; if not, a URL is provided to point the web source for subscribers (while non-subscribers cannot) to download the original piece. This can be then followed by uploading both the query and the downloaded original article to S.A.V.E., in either plaintext mode or encoded mode, for plagiarism detection in a pairwise comparison.

C. Parallelized Search in a Compressed Space

One of the biggest costs for plagiarism-detection is to maintain a database containing subscription-based journal articles and other online sources with copyright protections. Our current solution suggests a future possibility to store copyright-protected content in the form of encrypted PBSs. The detection results link back to the corresponding plagiarized sources from journal publishers, patent offices, government agencies, banks or hospitals without infringing their copyrights. In this setting, PBSs and their corresponding article URLs are stored. A URL reveals the full content of an article only when the viewers browse the content from an authorized IP address (by paying the subscription fee to the journal publisher), but reveals only the abstract for non-subscribers. In other words, S.A.V.E. stores the PBSs of web content, including those encoded from subscriber-based online articles, as well as their URLs. Users search S.A.V.E. database with query-encoded PBS and find the match. If the match has corresponding plaintext stored in S.A.V.E. database, a metafile can be downloaded for resolving the exact locations of the matched text; if not, a URL is provided to point the web source for subscribers (while non-subscribers cannot) to download the original piece. This can be then followed by uploading both the query and the downloaded original article to S.A.V.E., in either plaintext mode or encoded mode, for plagiarism detection in a pairwise comparison.

D. Appropriate Metric is Needed to Quantify the Extent of Plagiarism

Many plagiarism detection services report the plagiarism percentage in uploaded articles, which is de facto different from the academic definition of plagiarism - continuously copied words (CCW) exceeding an allowed limit. Plagiarism percentage for an uploaded article can also be misleading for cases where only 1% of the plagiarism is reported if one page in a 100-page article is made it impossible for data stolen during transmission or from the servers as well as storage of user-provided content for profit [22].

B. A Lower Cost to Maintain Databases for Plagiarism Detection Without Compromising the Copyright Protection for Subscription-Based Articles

Although major search engine companies and plagiarism detection services have policies to protect users’ uploaded information, the promises are often compromised during data transmission or constant malicious attacks. Our design provides a prototype to fundamentally eliminate such a possibility.

Although it is beyond the scope of this article, S.A.V.E. does support the non-private search, conducting the plagiarism detection using one’s uploaded plaintext(s). The functionality on this is detailed in the online user guide and the video tutorial at https://dyn.life.nthu.edu.tw/SAVE/.

III. DISCUSSIONS

A. The New Choice of a Search Engine That Provides Ultimate Privacy Protection of Uploaded Query Texts

S.A.V.E. enabled plagiarism detection for confidential documents without revealing the exact content of the query, which
fully duplicated from the internet, or 50% plagiarism is reported if the second page in a two-page report contains exclusively the references that the software fails to filter out. S.A.V.E. reports both the CCW and the copy percentage of an article and users can rank the examined articles by the longest CCW for each article.

E. Similarity and Distinction Between S.A.V.E. and Typical Search Engines

Similar to popular search engines nowadays, our PBS-based search also yields search results containing gaps or mismatches when aligning with a query, while our results respect the order of “amino acids” in a PBS more so than keyword-based results. On the other hand, our CCW-based ranking is carried out by prioritizing the highest local similarity, while popular search engines rank the pages containing specific keywords by how “central” these pages are in the network. S.A.V.E. allows the search of the full article (either encoded or plaintext) and a single keyword (plaintext only), while many popular search engines allow <35 keywords without a privacy-protection scheme herein introduced. In this scheme, even the search engine providers cannot know for sure what is searched while users still can get their search results.

F. Plagiarism Detection for Minor Paraphrasing and Tense Changes

We have implemented “a gap allowance feature” in our system that can tolerate mismatches up to 3 PBS characters. The feature had been shown to solve simple paraphrasing, tense modifications or syntactic changes that authors may use to evade the detection (see Figure S3). As shown in Figure S3 in SI, the Max CCW of the same article increased from 71 to 191 after applying the gap allowance feature to overcome minor paraphrasing. The Max CCW was found to increase by an average of 41.33 words with the gap-allowing S.A.V.E. for a collection of 30 articles from Chinese/English Wikipedia pages and PubMed articles (see gap_allowance_feature.xlsx in SI). On the other hand, extensive paraphrasing, if it indeed occurs (very rarely in our real-world experience), means that the students of interest have to understand the context first, which has achieved the minimal educational expectation (understanding what they are reading). Going beyond the point of simple paraphrasing, educators might have noticed - whenever extensive paraphrasing does occur, it is difficult to distinguish between heavily paraphrased ideas copied from others and authentic ideas, when the ground truth cannot be objectively known.

IV. CONCLUSION

S.A.V.E. enabled a new form of search engine where sensitive query content is the outstanding plagiarism detection search engine with the state-of-art bioinformatics algorithm, FM-index. S.A.V.E. has an efficient throughput as Bowtie and 5 orders better than BLAST. The encryption method promises confidentiality, where the method has a low false-positive rate when we set the plagiarism threshold at 12 continuous words or above, where text compression is realized by encryption of every word into a single alphabet. S.A.V.E. also recruits a in-house designed reference-removal module, which reaches 0.97 AUC.

V. METHODS

A. Encryption

In our current implementation, plaintexts from Microsoft Office documents, Open documents, txt files and pdf files etc., in all languages, are first extracted by tools including Apache Tika. Before encoding, the common English words in text are removed due to their lack of meaning except for grammatical purposes (Table S1). The alphabets in Western languages are recognized as those whose Unicode numbers are less than 1000 and that are not special characters. Any character, e.g., “岐” in Chinese (Fig. 4), “_elt” in Korean, “杂质” (any hiragana, katakana and kanji) in Japanese, having a Unicode higher than 1000 is
B. Search With Full-Text Index in Minute Space (FM-Index)

As previously stated, the FM-index is derived from Burrows-Wheeler transform (BWT) and is widely used in various NGS aligners [18], [19]. FM-index search first converts the raw PBS strings in our database into a sorted suffix array table, where the first column is called F and the last column is called L, also known as the BWT string (Fig. 5). First, we need to tally the number of each character in the column F and column L; for that, we use the C table and occ table to store the tally information for the F and L columns, respectively. The core search method is based on the LF mapping, which allows us to recover the string by mapping the characters of the F column and the L column. For example, suppose we have a query string “DNGQDRGĐQDRN$” (the first k-mer in Fig. 6) and a reference string “EDNGQDRGĐQDRN$”, we want to know where the query string occurs in the reference string (Fig. 6). The LF mapping searches character by character starting from the back of the string, in this case “D”. We look for the position of “D” in the F column by C table, and we know that the range of “D” in the F column is from row 1 to row 4. Next, we search for “G”, the next character of “D”, in the L column from row 1 to row 4 and in the F column. As Fig. 6 shows, the range of “G” in the F column is from row 6 to row 7 and row 2 has “D” in the F column and “G” in the L column, where the “G” is the first “G” in the L column (the second “G” is at row 10). Accordingly, we find that the first “G” in the L column is the same “G” in the F column. Hence, we find that the prefix of row 6 contains “GD”, which is the last two characters of our query. The LF mapping follows the same procedures above, keep searching for “R”, “D”, “Q”, “G”, “N”, and “D”. Finally, we can find that the query string “DNGQDRGĐ” is at row 1 in the suffix array table, which can then be converted back to the real position in the reference string.

The time complexity of FM-index search is O(m), where m is the length of the query string. Since m is normally shorter than the length of the database, the FM-index search is the most efficient algorithm known to such use cases [16].

C. Plagiarism Detection

Plagiarism detection in S.A.V.E. can be any method with the sequence comparison ability, especially widely used sequencing alignment methods in bioinformatics. All text converted into PBS in the encryption step can directly fit into those algorithms. In the implementation of the current version of S.A.V.E., we used BWT [17] and FM index [16] for plagiarism detection, which is also a widely used algorithm in sequencing alignment of bioinformatics. The alignment algorithms with BWT and FM index can efficiently search the exact match between query and reference. To apply BWT and FM index in plagiarism detection, we first need to build FM-index, the fastest among its kind, for references as the database. After establishing the database, we can perform plagiarism detection on the encrypted content efficiently due to the reduced time complexity of the algorithm. Furthermore, as we have encrypted all plaintext into PBS, the original content of the query document can never be exposed during the searching process.

D. Build Database

In order to perform the plagiarism detection on PBS generated from query documents, the database, or pre-built FM-index table, is necessary. In this study, we constructed databases with English Wikipedia pages, Chinese Wikipedia pages, PubMed Open Access articles, and PubMed Abstract as of the end of
Fig. 6. Alignment procedure example. The alignment procedure is the core of our plagiarism detection, which is not a typical string-matching algorithm. Before the alignment procedure begins, we will load the reference string that has been converted to archived FM-index tables (see Fig. 5), including a suffix array matrix shown on the left. A query string is first split into k-mers (k = 8, in this example), while each k-mer is searched against the FM-index tables, base by base, where the searching order is in grayscale from light to dark (or index 0 to 7). The searching results of each step with its range on the suffix array matrix are displayed as arrows with the same grayscale color and index on the left as well. After searching through all k-mers, we then merge the results reporting the aggregated result.

As of the end of 2021, we have collected 146.1GB of text data including a total 17802587 Wikipedia pages, 5613955 PubMed Open Access articles, and 40150958 abstracts.

Following the publication of this service and subsequent increase of web traffic, we will have an annual update for this. Also, we would like to prioritize the collection of additional text sources based on the plagiarism patterns we observed in the past 12 years, including researchers’ websites and scientific news.

### E. Pairwise Comparison and Temporary Database

For submissions of a set of encrypted documents, S.A.V.E. performs plagiarism detection on these documents, checking plagiarism content among the documents, called pairwise comparison (Fig. 1(a)). Since the current implementation of S.A.V.E. plagiarism detection is based on FM-index, S.A.V.E. builds an FM-index with the documents before performing plagiarism detection, where the FM-index is used for this particular set of documents only.

### F. Search Against Database

As mentioned, we used an FM-index-based searching algorithm for plagiarism detection. In detail, the encrypted document, where plaintext has been converted into ciphertext (PBS), will be split into k-mers (default k = 8). And each k-mer will search reversely from the end of the k-mer for exact matches on the database. The results will then be merged and report all matched sequences and their position aligned to the database. An example of the protection search was shown in Fig. 6.

---

**TABLE III**

| Description of the pattern | Pattern in regular expression | Weight  |
|-----------------------------|------------------------------|---------|
| Semicolon                   | ;                            | 15.79   |
| A capital alphabet followed by a full-stop and a comma (e.g., J.,) | [A-Z].[,] | 12.58   |
| Asterisk                    | \*                           | 11.62   |
| Ampersand                   | \&                           | 11.46   |
| 4 digits enclosed in a bracket | \(\{d(4)\}\)            | 10.70   |
| An upper-case alphabet followed by a full-stop, blanked by a blank space each side. Identifies the first or middle initials of an author. | (?<c=\w)([A-Z]).([?c=\w]) | 9.42   |
| Identifies the journal volume and page numbers separated by a colon | \d\{0,1\}:s*d+[-]\d+    | 9.05   |
| Identifies a year            | (?c=\w)[d(4)\[\[.\]    | 7.69   |
| Three digits at the beginning of each line. (The index of bibliography) | \[.\]s*d+[-]\d+         | 6.61   |
| Identifies the journal volume and page numbers separated by a comma or full-stop. | \[.\]s*d+[-]\d+         | 6.41   |
| A blank space followed by an upper-case alphabet and a comma. Identifies the first or middle initials of an author. | \d[A-Z]         | 5.97   |
| Identifies the first and middle initials of an author. (e.g. Chang Y.Y.) | (?c=\w)([A-Z]).([A-Z]).([?c=\w]) | 5.60   |
| DOI                         | (?c=\w)doi:DOI(?c=\w)     | 5.08   |
| et al                       | et al                       | 2.75   |
| Identifies names             | (?c=\w)([A-Z]).w+(?c=\w) | -0.35  |
| ,                           | ,                           | -0.72  |
| A capital alphabet          | [A-Z]                       | -0.82  |
| Identifies names followed by a comma | \d[A-Z].(\[a-z]\)+ | -0.83  |
| Identifies the index of a reference | \(\{\*\}\)        | -1.96  |

---

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
G. The Metrics of Plagiarism

In order to make it easier for users to identify if a given document contains plagiarized texts, we quantify the extent of plagiarism by reporting a whole-document similarity percentage (copy coverage) as well as the number of continuously copied words (CCW) for each potentially plagiarized paragraph. Herein, CCW score is a better measure of plagiarism than the similarity score because it follows a well-accepted academic standard to define plagiarism. CCW score is the number of consecutive words in the queried document that are found identical to those on the internet or in other documents uploaded together with this query file. In such a string-matching procedure, we allow groups of mismatches up to three consecutive words.

H. The Logistic Regression for Detecting Reference

The reference detector, a logistic regression model, takes the text in an article as the input and goes through it line by line to mark each line as either reference or text. Table III shows the list of patterns that commonly appear in the reference. The algorithm counts the appearance of the patterns in each line, normalized by the total number of characters in that line.

After scanning through the entire document, a pattern density matrix, \( D \), in a dimension of \( N_{\text{line}} \) by \( N_{\text{pattern}} \) is established, where \( N_{\text{pattern}} \) is the number of patterns (currently 19, listed in Table III) we examined for a given line, and \( N_{\text{line}} \) represents the total number of lines in the document. \( D_{i,j} \) value in the matrix \( D \) is the count of feature \( j \) in the \( i \)-th line, normalized by the total number of characters in the \( i \)-th line. The references are usually listed together (usually toward the end), we apply the moving average of window with a size of 3 to allow the pattern density per line to represent a neighborhood of 3 lines. The moving average of line density is written into the matrix \( D \) of the same size as \( D \). The logistic regression model is listed below.

\[
P_{\text{ref}}(z_i) = \frac{1}{1 + e^{-z_i}}
\]

where \( z_i = D \cdot W^T + c \). Here, \( i \) indexes the line, \( D \) is the \( i \)-th row of matrix \( D \) and \( W \) is a row vector of weight parameters, each of which is assigned for the on and off of one pattern and \( c \) is a constant. \( P_{\text{ref}} \) is a probability of line \( i \) of the document being a reference. The parameters in \( W \) listed in Table III are optimized by Maximum Likelihood Estimation [24] using 100 journal papers and 200 student assignments as the training examples where each line is manually assigned as reference or non-reference text.

ACKNOWLEDGMENT

We acknowledge the programming support and advice from Dr Soumya Ray, Dr Chi-Ching Lee, Dr Hongchun Lee, Cheng-Hann Yang, Yu-Xuan Wang and Chia-Yung Shih.