SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V10I10P102

Volume 10 Issue 10, 14-24, October 2023

Original Article

Improving Keyword Search and Data Retrieval in MCC Using Unigram Computing Based on a Probabilistic Approach

P. Venkat Reddy[1], Arif Mohammad Abdul[2], M. Kiran Sastry[3], Arshad Ahmad Khan Mohammad[4], C. Atheeq[5]

[1]*School of Technology, Woxsen University, Hyderabad, Telangana, India.*
[2,3,4,5]*School of Technology, Department of CSE, GITAM Deemed to be University, Hyderabad, India.*

[4]*Corresponding Author : ibnepathan@gmail.com*

Received: 04 August 2023 Revised: 16 September 2023 Accepted: 06 October 2023 Published: 31 October 2023

Abstract - *In the context of mobile communication, Mobile Cloud Computing (MCC) is a fast-expanding field that seeks to remedy the shortcomings of mobile devices. MCC can provide users with cost savings and dependable data maintenance since it uses cloud computing. Multi-keyword queries and fuzzy keyword-based searches are two examples of the current computational methods used for keyword searches in MCC; nevertheless, they also have drawbacks, such as random returns and irrelevant matches. This paper suggests using a Unigram Computing Probabilistic (UCP) approach to solve these problems. The method was designed to find incorrectly spelt terms and return relevant, timely results during keyword searches and data retrieval in MCC. The proposed improvements to keyword search and data retrieval speed and accuracy are substantial. MCC and UCP work together to give customers the best of both worlds: a mobile-friendly and cloud-based computing environment that can keep up with today's cellular communications demands.*

Keywords - *Data, Computing, Retrieval, Mobile cloud, Fuzzy.*

1. Introduction

The emergence of mobile devices, such as smartphones and tablets, has altered our communication and social interactions. Web browsing, social networking, gaming, and online purchasing are increasingly performed on mobile devices.

However, these devices' limited computational power and storage capacity make conducting computing-intensive tasks such as data processing and analysis challenging. By providing on-demand access to computing resources such as storage, processing capacity, and software applications, cloud computing has emerged as a promising solution for addressing these issues.

Cloud computing is a swiftly expanding technology that provides users with numerous advantages, such as cost savings, dependable data maintenance, high computational power, and scalability. In cloud computing, on-demand provisioning of resources to users with minimal administrative effort eliminates the need for upfront hardware and software investments. In addition, cloud computing provides inexpensive access to expensive resources on demand, allowing users to pay only for the resources they utilize. Cloud computing works with scaled data, making it an ideal solution for processing and analyzing large amounts of data.

Cloud computing's ability to facilitate service-oriented computing, also known as cloud computing, is one of its primary advantages. Service-oriented computing is a hub that maintains a pool of services where devices with limited computing capacity, battery life, bandwidth, software cost, and memory can receive excellent support. Cloud computing has emerged as a dependable method for overcoming the challenges of mobile devices' limited resources. By leveraging cloud computing, mobile devices can access computing resources and efficiently complete computationally intensive tasks.

Mobile Cloud Computing (MCC) has emerged as a potent computing paradigm that grants users access to vast computational resources and information storage, allowing them to surmount the limitations of mobile devices in mobile communication [3]. With advantages such as cost savings, dependability, and scalability, cloud computing is an attractive option for consumers to store and manage data. However, mobile devices' limited computing capabilities and constrained resources, such as low memory space, little battery life, and low computing power, continue to present

significant obstacles for computing-intensive tasks such as natural language processing, decision-making, and image recognition.

Data retrieval and keyword search are essential components of any efficient computing environment. Existing computing techniques, such as multi-keyword query and fuzzy keyword-based search, produce arbitrary results and irrelevant matches, among other limitations. These restrictions can hinder users' ability to efficiently retrieve and access cloud-based information.

To address these issues, this paper proposes using Unigram Computing based on a Probabilistic approach (UCP) for efficient keyword search and data retrieval in MCC. The UCP method seeks to identify misspelt words and generate accurate and effective results for keyword searches and data retrieval. The proposed method substantially enhances the precision and speed of keyword searches and data retrieval, providing MCC with a trustworthy computing environment. By combining MCC and UCP, this paper seeks to create an efficient computing environment that satisfies the requirements of contemporary mobile communication while offering users the benefits of cloud computing.

The rest of the paper is organized as follows. Section 2 provides an overview of related work in MCC's keyword search and data retrieval field. Section 3 discusses MCC's proposed UCP approach for efficient keyword search and data retrieval. Section 4 presents the experimental results and analysis of the proposed approach. Finally, Section 5 concludes the paper and provides future research directions.

2. Literature Survey

Mobile Cloud Computing (MCC) has been introduced to address the limitations of mobile devices in mobile communication. However, it is still insufficient for computing-intensive activities like decision-making, picture recognition, and natural language processing. The cloud database contains a high volume of data, making it challenging to extract required data through mobile cloud computing efficiently. Efficient computing techniques like keyword search have been introduced to overcome this challenge. Keyword search is essential for getting helpful documents from the cloud storage or database.

The precise and quick keyword search and data retrieval technique is known as efficient computing. Keyword search is essential for getting helpful documents from the cloud storage or database in every user's life. Searching from the cloud database is difficult because data in a cloud is stored in an encryption format for privacy preservation. Suppose the users keep their personal, critical, valuable, confidential, and delicate data in plaintext format in the cloud, which is not in the user's control. In that case, there may be a chance to attack and leak their information. Encryption is the best approach to preserving users' sensitive information [1]. If the data is encrypted, searching becomes challenging [2]. Thus, searching from the cloud database is difficult because data in a cloud is stored in an encryption format for privacy preservation.

In a recent review, many authors proposed searchable encryption methods to retrieve valuable information from the cloud even though data is encrypted. Most of the existing methods are designed to search data over encrypted form. Here, Figure 1 shows that data owners can encrypt and store their files on other servers without worrying about sensitive file leakage.

Searching these encrypted files over the cloud using Searchable Encryption Methods (SEM) has to sacrifice access and search patterns, leading to data leakage. According to Song et al. [4], the onion scheme is based on a symmetric system, which scans all the documents in the cloud database by comparing them with the trapdoor search keyword; if it matches, then reveals the exact location of the cloud database and the search time is linear. To secure the pattern, encrypt [4] the keyword and place it in the index.

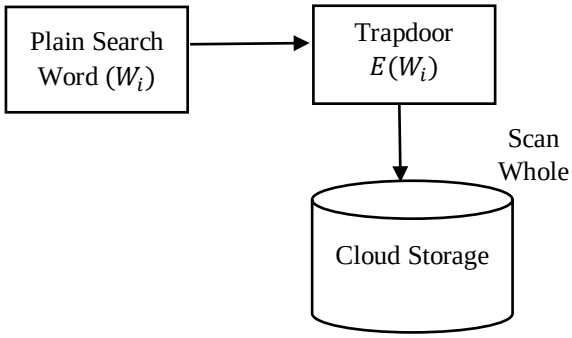

Fig. 1 Encryption and protection of sensitive data

E-J Goh [5] introduces a secure index search scheme Figure 2, which resolves the issue of linear search but depends on bloom filters to construct an index; it doesn't preserve trapdoor keyword privacy. Bloom filter is an array used to build the indexed keyword of the corresponding document, which is later used to find the document. For example, suppose a user submits an 'n' number of queries in which two queries with the word 'xy' and other 'n-2' queries do not contain the 'xy' word. Based on the output results of all queries, intruders can analyze which search word produces more documents.

R. Curtmola et al. [6] provided a new scheme where indexed keywords are hashed to secure privacy. However, these schemes only support exact match keywords, not typo-errored ones. To resolve the misspelling keyword problem, Li et al. [7] developed a wild card scheme based on the edit

distance concept shown in Figures 3 & 4. Wild card Fuzzy Set Construction (WFSC) maintains a typos database with a predefined edit distance. Edit distance value increases typos database size and degrades performance while searching. If edit distance = 1, it means one character mistake or missing, then * represents a wild card in the typos database for the word "Security". The size of the database is increased to 18 terms.

Similarly, when the two characters are misspelt, edit distance = 2, the typo database size is increased to 122 words.

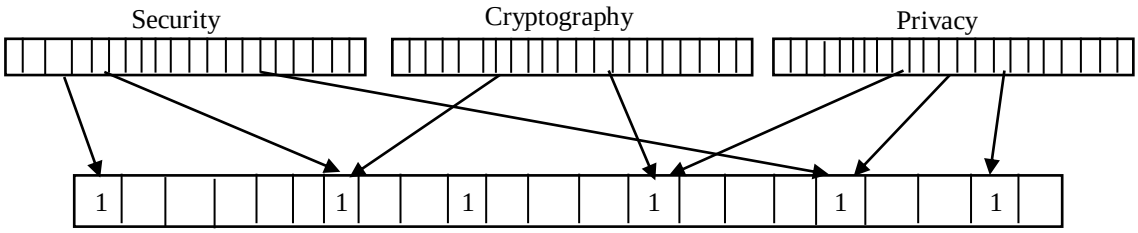

Fig. 2 Bloom filters to avoid linear search

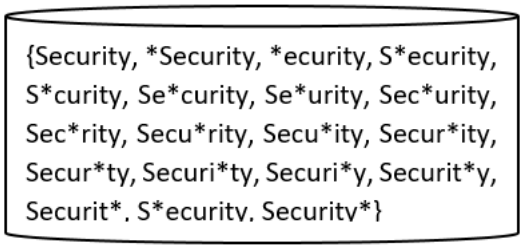

Fig. 3 Typos database of the word "security" with one character misspelt

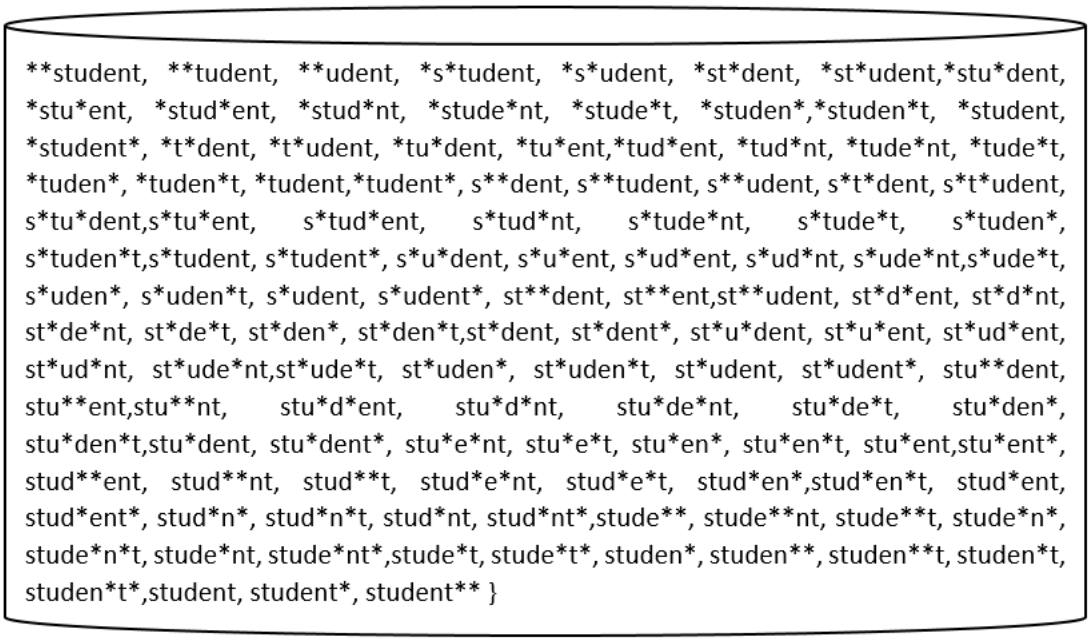

Fig. 4 Typos database of the word "student" with two characters misspell

Increasing edit distances increases the database's size, which leads to the problem of maintaining a separate database. The system must scan the complete database to find the correct word and edit the distance value if a misspelt word occurs. Liu et al. [8] proposed a solution by replacing it with a dictionary. Introducing Dictionary-based Fuzzy Set Construction (DFSC) may resolve the problem of edit distance database, but this scheme doesn't work with misspelt keywords. Dictionary maintains suffices of words like (Security, Secure, Secured, Secures, Securing) but not misspelt word again this scheme supports exact match. Later, Kuzu et al. [9], Lu et al. [10], and Kam Ho-ho [11] proposed efficient keyword searches to provide better results but were limited to boolean keyword searches.

Orencik et al. [12] proposed a blinding technique which improves security and resolves the limitations of Boolean keyword search by introducing a multi-keyword query search but produces randomized results [3]. Due to randomized effects, users face problems in finding their required files. Wang et al. [14] and Cao et al. [15] proposed the rank-based-keyword search method to overcome the issue of randomization result with the ranking result. Li, Xinghua, et al. [18] propose a verifiable ranked fuzzy single and multi-keyword search scheme with the highest accuracy but using a bi-gram approach containing the base index for every two characters of indexed keywords, which may increase the computation time.

Multi Keyword-Based Search (MKWBS) involves searching for multiple keywords or phrases in a given document or dataset to find matches containing all or most keywords [16]. This technique helps retrieve more precise and relevant results to the user's search query. However, MKWBS may not be effective when the user is unsure of the exact keywords or phrases to use in their search.

On the other hand, Fuzzy Keyword-Based Search (FKWBS) is a search technique that uses fuzzy logic to search for results that are similar or related to the user's search query, even if they do not contain the exact keywords or phrases entered by the user. This technique is helpful in retrieving results that may be relevant, even if the user is not sure of the exact keywords or phrases to use in their search. However, FKWBS may retrieve more irrelevant results than MKWBS.

The previous article proposed replacing the typo error word with the correct word instead of maintaining a typo database, generating a base index of characters, or constructing the dictionary-based fuzzy set.

Some of these schemes work on exact matches, and some of them take more computation time. These issues motivate designing Unigram Computing based on a Probabilistic (UCP) approach for MCC to find the misspelt character position in a word and compute an accurate and efficient result.

All the above schemes are categorized and explained as follows;

2.1. Secure Index Search Scheme Using Bloom Filters [4-6]

This scheme resolves the issue of linear search by using bloom filters to construct an index. However, it does not preserve trapdoor keyword privacy. A bloom filter is an array used to build the indexed keyword of the corresponding document, which is later used to find the document. The scheme uses exact-match keywords and is limited to this type of search. The main advantage of this method is that it is efficient and can be used for fast keyword searches. However, it has limited privacy protection and is unsuitable for fuzzy keyword searches.

2.2. Hashing Indexed Keywords Scheme [7]

This method protects privacy by hashing indexed keywords. However, it only supports exact match keywords and is unsuitable for fuzzy keyword searches. The main advantage of this method is that it provides better privacy protection than the previous scheme. However, it is limited to exact match keywords only.

2.3. Wildcard Fuzzy Set Construction (WFSC) [8]

This scheme resolves misspelt keywords by constructing a typos database with a predefined edit distance. The edit distance value determines the size of the database and affects the system's performance while searching. This method is helpful for fuzzy keyword searches and can retrieve relevant results even if the search query contains typos. However, the database size increases with the edit distance, which may affect the system's performance. Additionally, maintaining a separate database can be difficult.

2.4. Dictionary-Based Fuzzy Set Construction (DFSC) [9]

This scheme is similar to WFSC but uses a dictionary instead of a typos database. The dictionary maintains the suffices of words but does not work with misspelt keywords. The main advantage of this method is that it is more efficient than WFSC since it does not require a separate database. However, it is limited to exact match keywords and cannot retrieve relevant results for misspelt keywords.

2.5. Blinding Technique for Multi-Keyword Query Search [10-15]

This method improves security and supports multi-keyword query search, but it produces randomized results. The randomization effect makes it difficult for users to find their required files. However, the rank-based-keyword search method proposed by Wang et al. and Cao et al. resolve this issue. This method ranks the results, improving users' search experience. However, it may take more computation time, and the order may not always be accurate.

Overall, each of these methods has its benefits and limitations. The choice of method depends on the specific requirements of the search application. Computing techniques like multi-keyword query search and fuzzy keyword-based search have restrictions, such as producing randomized results and irrelevant matches. These limitations can hinder users' ability to efficiently retrieve and access information from the cloud, as Keyword search and data retrieval are crucial components of any efficient computing environment.

This paper proposes using Unigram Computing based on a Probabilistic approach (UCP) for efficient keyword search

and data retrieval in MCC to address these challenges. The UCP approach aims to locate misspelt words and produce accurate and efficient keyword search and data retrieval results.

The proposed approach significantly improves the accuracy and speed of keyword search and data retrieval, providing a reliable computing environment for MCC. By combining MCC and UCP, this paper aims to create an efficient computing environment that meets the demands of modern mobile communication while providing users with the benefits of cloud computing.

3. Proposed Approach

Typo errors are pretty common when the user types a keyword word for searching. Cannot extract the user-required file if the word is not matched. The objective of the Unigram Computing based on a Probabilistic (UCP) technique is to repair misspelt words by computing the probabilities of each letter in a term. This is done to achieve accuracy and enhance efficiency by minimizing the amount of time spent on computation. Set up a platform so that a client is responsible for uploading the user's sensitive information to the cloud using storage as a service in an encrypted form to protect against unauthenticated users and create an index containing the keywords of all uploaded files for future extraction.

In this concept, the platform consists of (i) a cloud user, (ii) a client and, (iii) a CSP. A cloud user contributes to uploading a file to a client and searching files through the cloud. A client contributes to uploading a file to the cloud and creating an index by generating keywords before uploading a file using a frequent keyword algorithm.

Cloud users enter search terms through a trapdoor, and the trapdoor calculates the likelihood that a particular search term, w_i, will be found using indexed terms. Compare w_i with the probability of the indexed keyword after producing the possibility of a search phrase. If the search term w_i matches one of the keywords in the index, encrypt it before sending it to CSP. CSP will then return the associated files in encrypted form. Finally, the files are decrypted by approved cloud users.

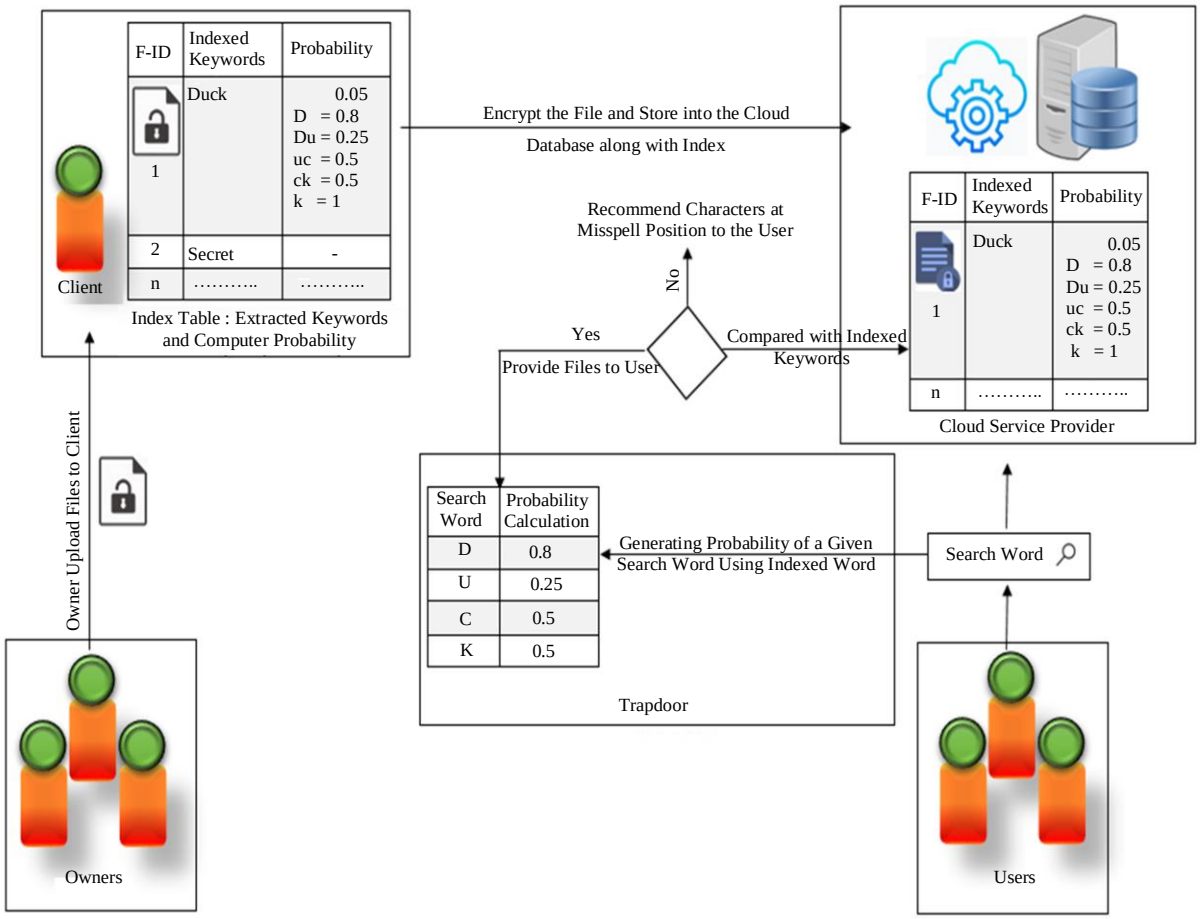

Fig. 5 Architecture of UCP

3.1. UCP Architecture

Figure 5 shows cloud users input a search term, and algorithm 1 computes the probability of each word character based on indexed keywords. Compared with an indexed keyword, if a search term matches an index of keywords, then return associated files or compare the search term with the probability of each character of an indexed keyword. If the result of the likelihood of a character is zero, then based on the previous character probability, algorithm 1 provides recommendations and corrects the misspelt search term. If the search term generates a probability result of a character zero, it indicates that the misspelt character position is identified. The search term is corrected based on recommendations, and the user-required file is extracted.

3.2. The Efficient Search Using UCP

The proposed UCP approach achieves efficiency by the following algorithms:

- Keyword Extraction
- Index Construction
- Probabilistic Calculation
- Ranking

3.2.1. Keyword Extraction

The frequent keyword generation algorithm extracts the top k frequent words when the file is uploaded to the cloud. Term frequency-inverse document frequency indicates how important a word is to a document. It computes a weight to each word, which signifies the importance of the word in the document. Algorithm 1 discusses the step-to-step process of how to extract a keyword from uploaded files and index.

Algorithm: Frequent Keyword Generation and Indexing

Step 1: Outsourced the files to a cloud.

Step 2: Remove stop words from each file by using NLTK.

Step 3: Extract all frequent words from a file.

Step 4: Apply the stemming algorithm to identify the root word.

Step 5: Calculate term frequency for each word, i.e.,

$$TF = \frac{number\ of\ times\ 'x'\ word\ appeared}{Total\ number\ of\ words\ in\ a\ files}$$

Step 6: Calculate Inverse document frequency for each word, i.e.,

$$IDF = \frac{Total\ files}{Total\ number\ of\ files\ 'x'\ word\ appear}$$

Step 7: Calculate TF-IDF

$$TF - IDF = TF * IDF$$

Step 8: The word frequently appears, and the count of that word w_i is more significant than other words w_j in the file and has more weight.

$$Weightage\ of\ a\ indexing\ Word = \sum_{i=0}^{n} w_i > \sum_{j=0}^{n} w_j$$

Step 9: The probability of each character of the keyword is

$$Probability_{k_i}(Char_n|Char_{n-1}) = \sum \frac{(Char_{n-1},\ Char_n)}{Char_{n-1}}$$

Step 10: The probability of the keyword is

$$Probability_{k_i} = \prod_{i=1}^{n} Probability_{k_i}(Char_n|Char_{n-1})$$

Each keyword is associated with its related file ID. Each indexed keyword has calculated the probability for efficient and accurate retrieval of the files.

3.2.2. Index Creation

The client is responsible for uploading a file in the cloud; whenever the user wants to upload a file in the cloud, the user redirects their file to the client, and the client extracts keywords from a file using algorithm.1. The Client constructs an index to place all the keywords which are extracted from files. These keywords will be used for file extraction by cloud users in future.

While uploading a file f_i by the client, the algorithm.1 fetches multiple frequent keywords k_i from k_1-k_n to efficiently retrieve all files of interest. These keywords k_i from k_1-k_n are placed in an index by calculating the most frequent keyword of a particular file using equation (1).

$$frequent\ keyword_{file_i} = \sum_{i=1}^{n} \frac{freq.keyword\ k_{i_{file_i}}}{Keyword\ k_{i\ file_i}} * 100 \quad (1)$$

$$i = 1, 2 \ldots \ldots \ldots n$$

$freq.keyword_{file_i}$=Calculate the $frequent\ keyword\ k_{i_{file_i}}$ in the file i,

$|\ Keyword_{file_i}\ |$ = Total $Keyword\ k_{i_{file_i}}$ of a file i

Suppose the file's content consists of n words. In that case, the procedure uses equation (1) to count the number of times each word appears before sorting the words into ascending order, extracting the top n words for quick

computation and placing them into an index. The terms may have a post and prefix, like {legally, illegal,} to the root word "legal" to determine the root word of the frequent keyword $k_{i_{file_i}}$ of a file i, use the stemming algorithm [17].

The frequent keyword $k_{i_{file_i}}$ of a file i, is placed into an index and calculate the probability of these indexed keywords for efficient extraction of a file from the cloud by the users.

3.2.3. Probability Calculation

The probability of each character can be calculated as follows:

Table 1. Probability calculation of a keyword DUCK

	D	**U**	**C**	**K**	**/**
/	**0.4**	0	0	0	0
D	0	**0.5**	0	0	0
U	0	0	**1**	0	0
C	0	0	0	**0.33**	0
K	0	0	0	0	**1**

$$Probability_{k_i}(Char_n|Char_{n-1}) = \sum \frac{(Char_{n-1}, Char_n)}{Char_{n-1}} \quad (2)$$

The likelihood of the following character can calculate each character's probability depends only on the preceding k characters of k_i as shown in Table 1. $Char_n$ is a current character and $Char_{n-1}$is the preceding character of a $keyword\ k_i$. After calculating all the characters' probabilities, obtain the total probability of a $keyword\ k_i$:

$$Probability_{k_i} = \prod_{i=1}^{n} Probability_{k_i}(Char_n|Char_{n-1}) \quad (3)$$

Let the frequent keyword $k_{i_{file_i}}$ of a file, i be indexed as:

Table 2. Frequent keyword k_i of all files

File_ID	Frequent Keywords
101	Secret
102	Computation
103	Probability
104	Doctor
105	Duck

Let the probability of each character of the indexed keyword duck be calculated using equation (2),

$$p(D,\_) = \frac{(\_,D)}{Count(\_)} = 2/5 = 0.4$$

$$p(u,D) = \frac{Count(D,u)}{Count(D)} = 1/2 = 0.5$$

$$p(c,u) = \frac{Count(u,c)}{Count(u)} = 1/1 = 1$$

$$p(k,c) = \frac{Count(c,k)}{Count(c)} = 1/3 = 0.33$$

$$p(/,k) = \frac{Count(k,/)}{Count(k)} = 1/1 = 1$$

The probability of the keyword DUCK is computed by using equation (3),

$$p(duck) = 0.4 * 0.5 * 1 * 0.33 * 1 = 0.066$$

Calculate and compare the likelihood of the search term, k_i., with the probability of the keyword that has been indexed. If the search term k_i was misspelt, then find the closest likelihood probability of each keyword character and replace it with that word.

Table 3. Probability of each character of a misspelt keyword

	D	**A**	**C**	**K**	**/**
/	**0.4**	0	0	0	0
D	0	**0**	0	0	0
A	0	0	**0**	0	0
C	0	0	0	**0.33**	0
K	0	0	0	0	**1**

The probability calculation of a misspelled keyword using equation (3) is as follows:

The probability of each character of the keyword dack is calculated as,

$$p(d,\_) = \frac{Count(\_,d)}{Count(\_)} = 2/5 = 0.4$$

$$p(i,d) = \frac{Count(d,a)}{Count(d)} = 0/2 = 0$$

$$p(c,a) = \frac{Count(a,c)}{Count(a)} = 0/2 = 0$$

$$p(k,c) = \frac{Count(c,k)}{Count(c)} = 1/3 = 0.33$$

$$p(/,k) = \frac{Count(k,/)}{Count(k)} = 1/1 = 1$$

The probability of a word dack keyword is $p(dack) = 0.4 * 0 * 0 * 0.33 * 1 = 0$

Word dack yields 0 results. To replace the misspelt keyword with the closest probability keyword, the misspelt search word (w_i) compares character by character with the indexed keywords probabilities. Figure 6 displays the flowchart for service requests, services like file uploading and file searching. While uploading a file, frequent keywords are extracted and indexed for efficient and accurate extraction of a file. When a user uploads a file, they act as the owner; when they search for a file using a search term, they act as the user.

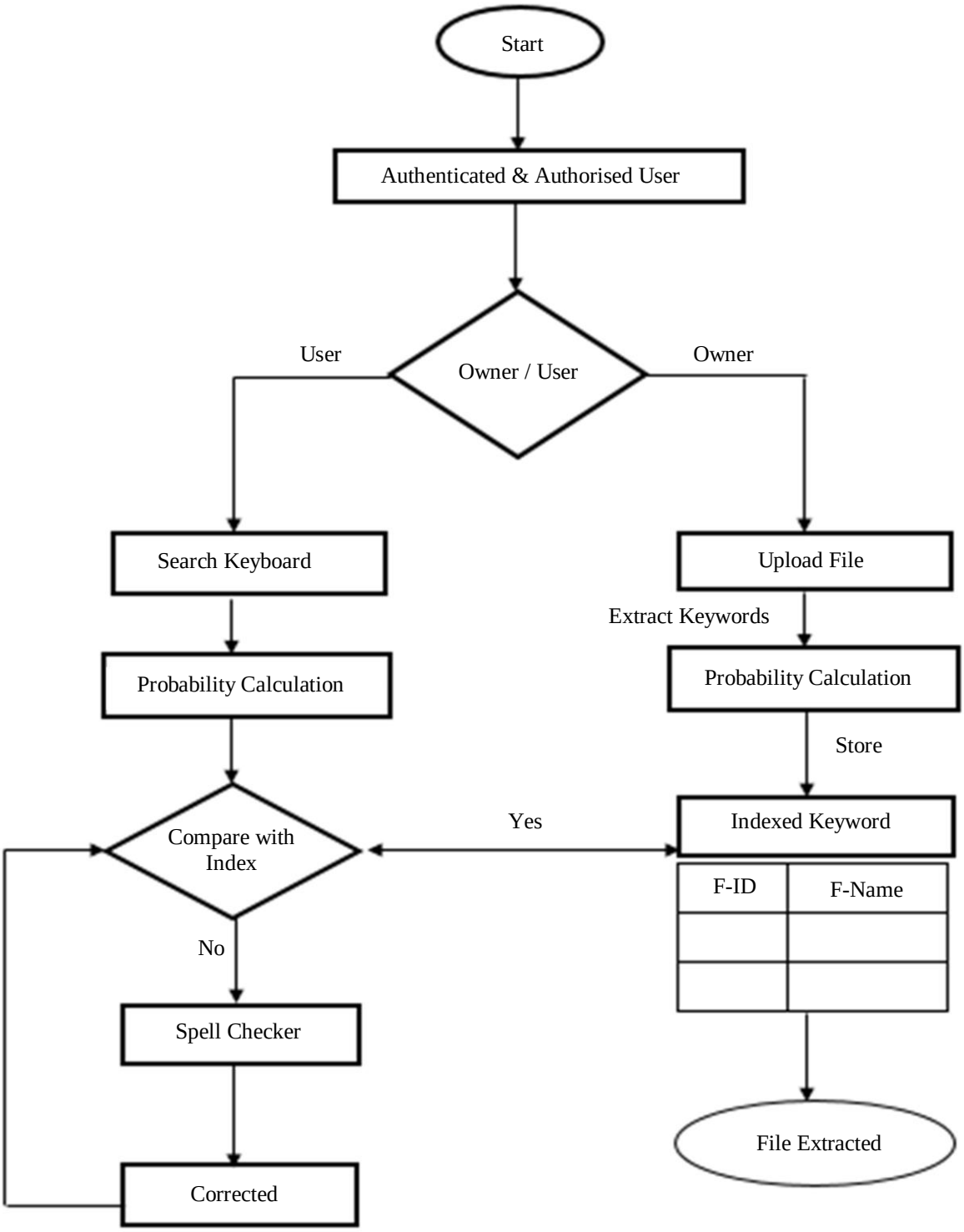

Fig. 6 Working flow of a file upload to extraction

3.2.4. Files Rank Calculation

Through the trapdoor, the user enters a search term, and the trapdoor uses indexed terms to calculate the likelihood of the search term, w_i. Compare (w_i) with the indexed keyword's probability after calculating the search term's probability. If the search term (w_i) maps with the indexed keywords, figure out each file's score and provide the user.

$$Score\left(w_i, F_j\right) = S + (f_j, w)/|F_j| \quad (4)$$

S = number of searching keywords,
(f_j, w) = Keyword w 's frequency in the file F_j,
$|F_j|$ = Total keywords frequency of files

To extract file F1, the queried words are term1 and term2; their frequencies are 6 and 7, and the file's overall keyword frequency is 15. For file F2, the queried words are w1 and w2; their frequencies are 6 and 8, respectively, and the frequency of file F2 is 20. The following formula (4) calculates a search word's score:

$Score\ (w, \text{F1}) = 2+(6+7/15) =2.87$

$Score\ (w, \text{F2}) = 2+ (6+8/20) = 2.7$

$Score\ (w, \text{F1}) > Score\ (w, \text{F2})$

The search term algorithm's step-by-step workflow is presented in algorithm 2. The user enters a search term to determine its likelihood and contrasts it with the indexed phrase's probability. Depending on the result, the user is shown the files. The files with the highest scores are at the top level, while those with the lowest are at the bottom. Based on the search word's occurrence in the file, scores can be computed for that file.

Algorithm: Search Term

Input: $search\ words, total\ no.of\ search\ words = S$
Output: $weighted\ keyword\ based\ ranked\ files$
for each search w_i from w_1 to w_n
Stem w_i
Mapping w_i with a probability of indexed keyword
if $w_i = true$ then
for j = 1 to n, do
$Score\left(w_i, F_j\right) = S + (f_j, w/|F_j|)$
end for
Sort $(Score\left(w_i, F_j\right))$
end if
Sorted $(Score\left(w_i, F_j\right))$
end for

4. Performance Analysis

We implemented our newly developed probabilistic (UCP) keyword search technique for unigram computing in Java on a Windows 11 server that has an IntelI CoreI i7-10610U CPU running at 1.80GHz and 2.30GHz and a 64-bit operating system.

We employed two performance metrics-calculation time vs. the number of keywords and computation time vs. the number of documents-to assess the effectiveness and precision of our technique. Additionally, we contrasted our probabilistic unigram keyword search mechanism's findings with those of pre-existing Multi-Keyword-Based Search (MKWBS) and Fuzzy Keyword-Based Search (FKWBS) systems. Comparing our novel mechanism to existing well-known search mechanisms, our objective was to determine if it was more efficient and accurate at producing search results. By doing this, we can assess the possibility of our strategy for improving search results and giving users more pertinent results.

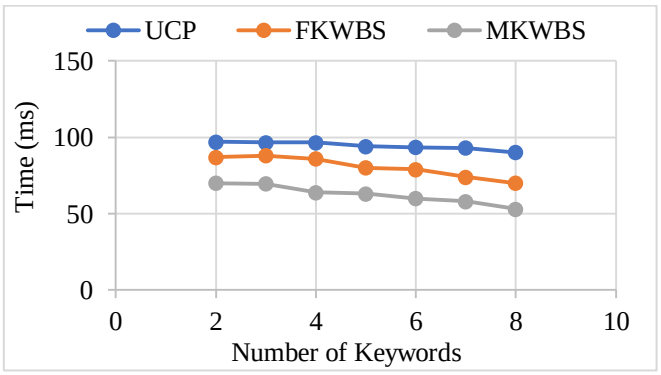

Fig. 7 Computation time vs The number of keywords

Figure 7 depicts the calculation duration of the computing algorithm during the query search with varying keyword counts. The outcome shows that the performance of the suggested mechanism, or UCP, is superior to that of the two currently used computer algorithms, FKWBS and MKWBS.

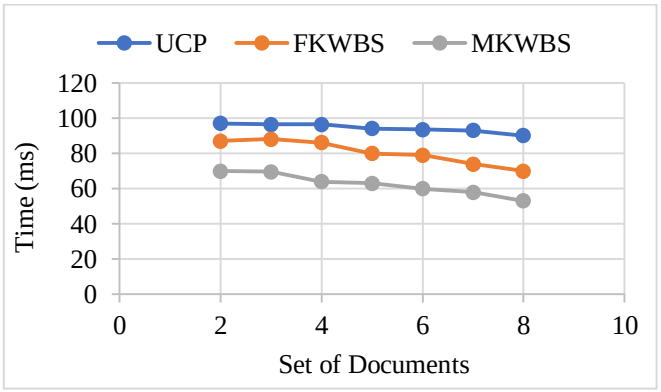

Fig. 8 Computation time vs Size of a document

Figure 8 depicts the computation times for the various size variations of the cloud-based data upload and download operations. The outcome shows that the performance of the suggested mechanism, or UCP, outperforms that of the two currently used computing algorithms, FKWBS and MKWBS.

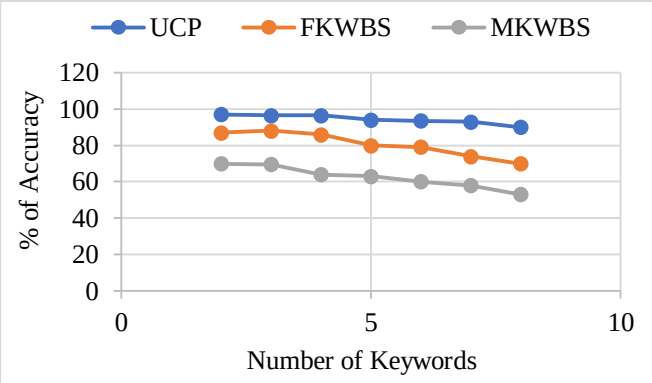

Fig. 9 Computing algorithm performance with variation in the number of keywords

The accuracy of the computing method throughout the query search with varying keyword counts is shown in Figure 9. Since the suggested work was based on the probabilistic unigram computation, the result shows that the proposed mechanism, i.e., UCP performance, outperforms the already-existing computing algorithms FKWBS and MKWBS.

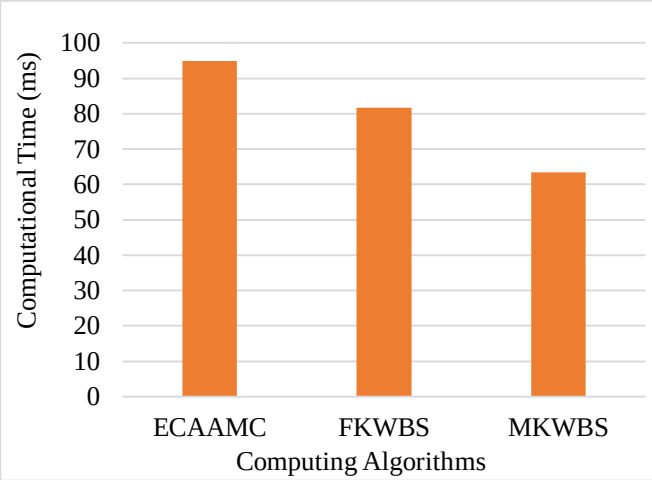

Fig. 10 Average computation time during the keyword search in the MCC environment (maximum keywords the query is up to 15)

The average computation time for the computing method throughout the query search with varying keyword counts is shown in Figure 10. Since the suggested work was based on the probabilistic unigram computation, the result shows that the proposed mechanism, i.e., UCP performance, outperforms the already-existing computing algorithms FKWBS and MKWBS.

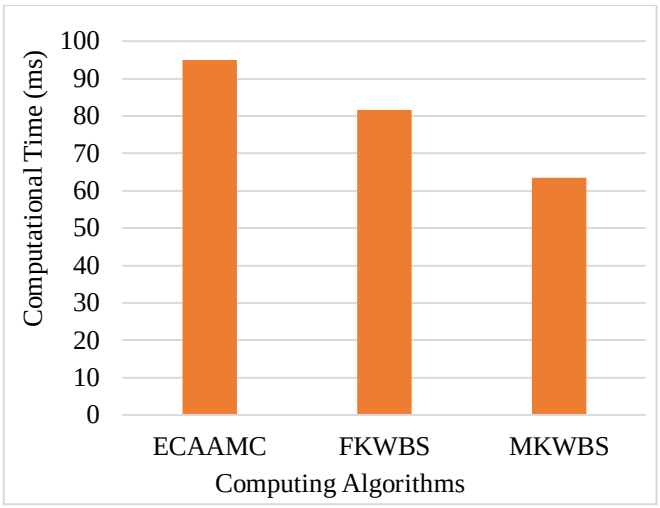

Fig. 11 Average computation time during the upload/download document in the MCC environment (maximum downloaded document size is 3000 Mb)

The average calculation time for the computing method in size fluctuation throughout the upload and download of data from the cloud is shown in Figure 11. The outcome indicates that the suggested mechanism, or UCP performance, outperforms the already-in-use computer algorithms FKWBS and MKWBS due to the proposed algorithm's quicker and more precise searching process.

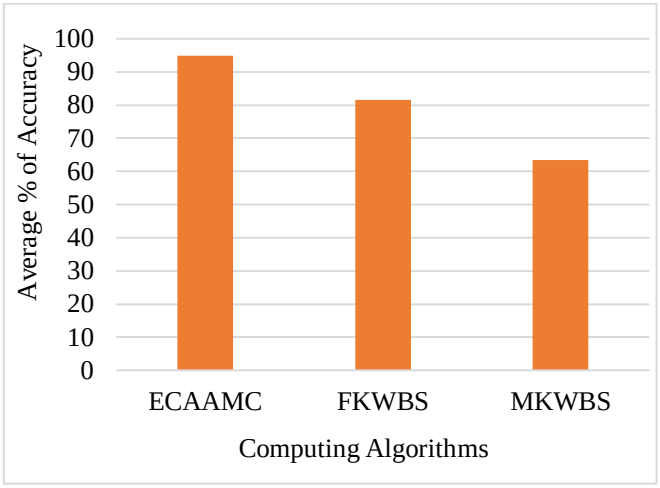

Fig. 12 Average percentage of accuracy of the computing algorithms during the document search with keywords

The average performance of the computing algorithms in terms of how precisely they extract the data for the given query with a change in the number of keywords is shown in Figure 12. The outcome indicates that because the proposed technique is based on indexed keyword likelihood, it performs better than current computing algorithms FKWBS and MKWBS, i.e., UCP accuracy. The results shown in Figures 7 to 12 indicate that the proposed scheme, i.e., UCP, produces a better result.

5. Conclusion

Mobile Cloud Computing (MCC) has revolutionized how we use mobile devices and interact with the cloud. However, current computing techniques for keyword searches in MCC have limitations that affect the accuracy and efficiency of data retrieval. This paper proposed a new approach, Unigram Computing based on a probabilistic (UCP) system, which addresses these issues by locating misspelt words and producing accurate and efficient results. The proposed approach significantly improves the speed and accuracy of data retrieval in MCC, providing users with a more efficient computing environment that meets the demands of modern mobile communication while offering the benefits of cloud computing.

References

[1] Mohammad Yamin et al., "An Innovative Method for Preserving Privacy in Internet of Things," *Sensors*, vol. 19, no. 15, pp. 1-24, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Yu Ishimaki, Hiroki Imabayashi, and Hayato Yamana, "Private Substring Search on Homomorphically Encrypted Data," *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, Hong Kong, China, pp. 1-6, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[3] I. Lakshmi, "A Review on Security in Mobile Cloud Computing," *SSRG International Journal of Mobile Computing and Application*, vol. 6, no. 2, pp. 4-11, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Dawn Xiaoding Song, David Wagner, and Adrian Perrig, "Practical Techniques for Searches on Encrypted Data," *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, Berkeley, CA, USA, pp. 44-55, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[5] Eu-Jin Goh, "Secure Indexes," *Cryptology ePrint Archive*, 2003. [Google Scholar] [Publisher Link]

[6] Reza Curtmola et al., "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *CCS '06: Proceedings of the 13th ACM conference on Computer and Communications Security*, pp. 79-88, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[7] Jin Li et al., "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," *2010 Proceedings IEEE INFOCOM*, San Diego, CA, USA, pp. 1-5, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[8] Qin Liu, Guojun Wang, and Jie Wu, "Secure and Privacy Preserving Keyword Searching for Cloud Storage Services," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 927-933, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[9] Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu, "Efficient Similarity Search over Encrypted Data," *2012 IEEE 28th International Conference on Data Engineering*, Arlington, VA, USA, pp. 1156-1167, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[10] Yanbin Lu, "Privacy-Preserving Logarithmic-Time Search on Encrypted Data in Cloud," *NDSS*, 2012. [Google Scholar] [Publisher Link]

[11] Kam Ho Ho, "*Semantic Search over Encrypted Data in Cloud Computing*," Masters Thesis, San Jose State University, 2013. [Google Scholar] [Publisher Link]

[12] Cengiz Örencik, and Erkay Savaş, "An Efficient Privacy-Preserving Multi-Keyword Search over Encrypted Cloud Data with Ranking," *Distributed and Parallel Databases*, vol. 32, no. 1, pp. 119-160, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[13] Saripalli Vinod Manikanta, and Kondapalli Varaprasad, "A Secure Privacy Preserving Information Retrieval Model in Cloud Computing," *International Journal of Computer and Organization Trends*, vol. 9, no. 1, pp. 16-19, 2019. [Publisher Link]

[14] Cong Wang et al., "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1467-1479, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[15] Ning Cao et al., "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222-233, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[16] Putchakayala Mahesh Reddy, and Mula Sudhakar, "An Efficient Cluster Based Searching Process for Finding Keyword Query Related Documents," *SSRG International Journal of Computer Science and Engineering*, vol. 5, no. 2, pp. 1-4, 2018. [Publisher Link]

[17] Julie Beth Lovins, "Development of a Stemming Algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, no. 1-2, pp. 22-31, 1968. [Google Scholar] [Publisher Link]

[18] Xinghua Li et al., "VRFMS: Verifiable Ranked Fuzzy Multi-Keyword Search over Encrypted Data," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 698-710, 2023. [CrossRef] [Google Scholar] [Publisher Link]