Analysis Of Pattern Matching Algorithms Used For Searching The Patterns In MLIR Framework

Chayapathi A R[1], G Sunil Kumar[2], Manjunath Swamy B E[3], Thriveni J[4], Venugopal K R[5]

[1]Information Science Department, Visvesvaraya Technological University, Acharya Institute of Technology Bengaluru, Karnataka, India, archayapathi@gmail.com
[2]Computer Science Department, Banglore University, UVCE Bengaluru, Karnataka, India
[3]Computer Science Department,Don Bosco Institute of Technology, Bengaluru, Karnataka
[4]Computer Science Department, Banglore University, UVCE Bengaluru, Karnataka, India,
[5]Banglore University Bengaluru, Karnataka, India

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 April 2021

Abstract. There is a solid requirement for the real-time ordering of enormous measures of information streaming at the rate of 10GB/s or more. This information should be scanned for designs and the query items are time-basic in fields as different as security reconnaissance, monetary services including stock exchanging, checking the basic well-being states of patients, atmosphere notice frameworks, etc. Here, the file will be required to age-off in a little time and thus will be of limited size. Notwithstanding, such situations can't endure any infringement of indexing latency and severe pursuit reaction times. Likewise, future greatly parallel (multicore) structures with capacity class recollections will empower rapid in-memory constant indexing, where the index can be totally put away in a high limit storage class memory. As the web is growing, it means a number of documents on the web are also growing so index size will also grow. The approach would be to partition a single large search index into smaller partitions and assigned to each node in the cluster. Later when a search request comes each node in the cluster will perform a search on the local index for each query term and send the result back to the central machine, which will later combine search results from all nodes in the cluster and then send it to the user. In this paper also comparative analysis of rabin-karp and Knuth-Morris-Pratt and Boyer-Moore algorithm has been done based on the execution time of the search pattern. An experimental result of the algorithm is presented in this paper, based on the result it has been concluded that KMP algorithm is better to use in MLIR framework.

Keywords: Indexing, Rabin Karp, Knuth-Morris-Pratt, Boyer-Moore,MLIR.

1 Introduction

Multilingual Information Retrieval (MLIR) System as the name denotes exhibits capabilities of managing inquiry utilization with a single language and accomplishes report-recovery through various dialects. MLIR ensures translation of queries is carried out with utmost care. This work relies on language-independent indexing technology to work on the text chunks of three languages: Kannada, Hindi, and English. In the present interconnected world, it is found that archives and different wellsprings of data arrive in a variety of dialects. This represents an issue for some pursuit applications. We have to comprehend the language of these archives admirably well to dissect them appropriately and give the most ideal search experience.[3] The work employs a multilingual dictionary-based word-to-word query translation. The experimental results is used for evaluation, analysis, and comparison of the performance of three different algorithms used for searching patterns. The existing works fail to satisfy the needs of Multilingual search through keywords, not only issues of establishing semantic importance of different starting points have to be considered, Additionally ability to gather swift access, deduplicating, rectifying, and creating joins are handled. We drew motivation from these factors designed a system displaying results for the keyword search through multi-languages with user preferences as the pivotal point. The system is envisioned for retrieving multiple languages-based results simultaneously. So, on carrying out a search for an English keyword the results are obtained in multiple languages. The plan is to employ Hindi and Kannada languages to procure the results. As our team is comfortable conversing in these two languages along with English we preferred to use them for result generation. As future work, it can be extended for all Indian languages with a script.

The rationale behind the usage of the framework is to ensure the users can view documents written in various languages on executing a query. Though contrasts are bound to emerge with monolingual and multilingual situations provided the users have knowledge of more than one language. The focus is to create UI possessing differential showcasing capabilities. In a situation where more than one user encounters outcomes Interpreting them to few more dialects is essential.

Based on the multifaceted nature of clients interpreting the components at different stages is accomplished for various data-access requirements. Different types of interpretation used are: Title interpretation, Abstract interpretation, Keyword interpretation, Term interpretation, Explicit interpretation, caption interpretation, full document interpretation are various types. The goal is to construct a search engine capable of giving results in multiple languages at a given point in time. As all Indians can converse in more than one language on average, it becomes necessary to cater to the information needs of people in multiple languages. Also, the Information will be available in multiple languages and the objective is to ensure all the users have access to information in more than one language. The intent of our proposed system are:

- Document Indexing, retrieving, filtering.
- Prepare Summary and Present the information
- Procuring metadata in multilingual format; information retrieval with cross-language capabilities
- The capability of Semantic Parsing and Morphological analysis
- Ensure disambiguation techniques and document segmentation process

The focus of the research work is constructing an indexing framework employed for indexing multilingual documents. Also developing algorithms for effective storage and posting of multilingual documents is a key objective. Such a process ensures an efficient reduction in the index size of vocabulary matches for multiple languages is in place. The current day generic search engines like Google produce search results in the suitable language: for instance if the search query is in English the information is retrieved English only. For procuring results in Multiple languages like Kannada, Hindi, and so on the keywords have to be keyed in those respective languages.

2 Design and Implementation of the Architecture

The framework for structural design comprises a system to handle objects called frameworks. They also support the thinking-process involved in the fundamental property of these elements. Fig.1 elucidates the architecture of search engines. In the Admin section File Indexing module is used to allow the administrator to create indexing for each file/document uploaded. The Inverted indexing facilitates setting up an index on the Admin section. On the User section provision to enter the query for search is given. The Language Translation module determines the language used and also determines its comparable languages. The string searching algorithm employed in this work can detect a set of pattern strings in a queried text. It also enables speed testing of the equality between patterns and the substrings. The Index Mapping module facilitates the determination of the index of the file based on the search query and eventually exhibits the results in chosen destination language[1].

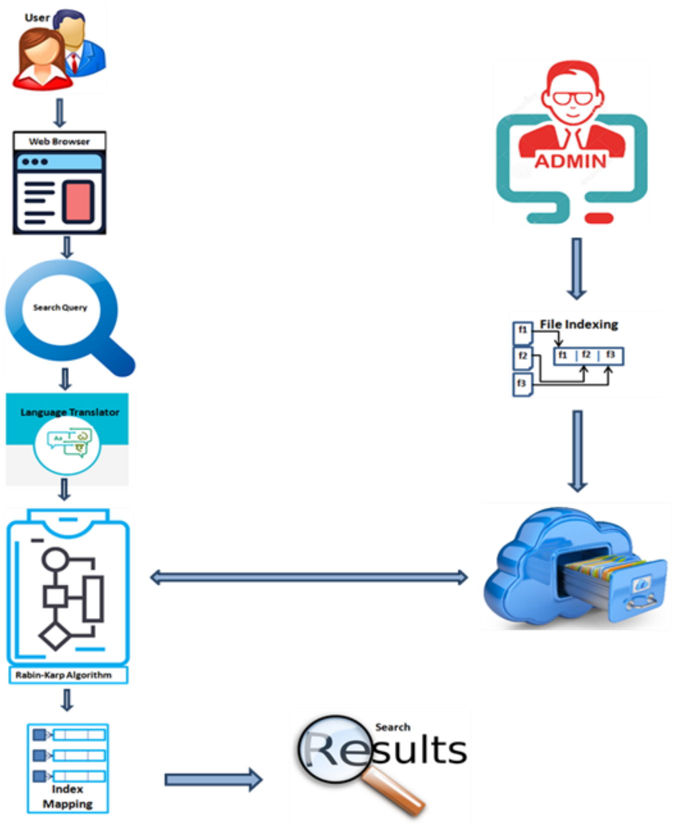


Fig.1. The Search Engine Architecture

Fig.2 elucidates the blow-up image of the context-level design module planned for implementation in the MLIR framework. This also depicts a whole data-system and signifies the interaction taking place with external entities.

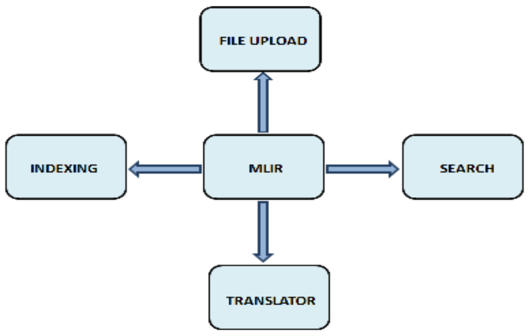


Fig.2. MLIR Framework Contextual Design

Steps followed at Admin side of the framework:

1. Accumulate all indexing documents.
2. Initiate the Tokenization process for the text, transforming each document as a list of tokens.
3. Accomplish linguistic preprocessing to obtain normalized tokens list, that has indexing terms.
4. Create an inverted index by the process of document Indexing which comprises of a dictionary and postings.

Steps followed at Client-side for displaying the searched result:

1. Key in the Query for searching.
2. Translating searched query into supported Language.
3. Selection of pattern matching Algorithm.
4. Displays the searched result.
5. The graph is displayed for Analysing different pattern matching algorithms based on the execution time.

2.1 Inverted Indexing

Indexing is a crucial part of any information retrieval system. It is a challenging task requiring paying attention to any practical issues. The main goal of hoarding an index is to make faster queries and improved performance in detecting correct documents for a given search query. If the indexing method doesn’t exist significant amount of time and computing power would have required for the search engine as it would scan every document in the collection. For example, if an index of 1,00,000 documents can be inquired in milliseconds, a consecutive examination of every word in 1,00,000 huge documents requires hours of computation. Further bigger system storage is expected to store the index, a moreover a substantial increase in the time for updating is evaluated by comparing with the saved information retrieval time.

Inverted indexing is the most popular index structure in search engines [2]. Inverted Indexing is an indexing method used for storing a mapping from content that includes words or numbers to the location where it belongs like a table or document or a group of documents. The main reason for using inverted indexing is because it provides faster full-text searches which come at a cost of increased CPU processing when a document is stored in a db.

There are different types of indexing techniques like forward indexing, inverted indexing, and Suffix trees. Its concurred inverted indexing is better compared with the forward indexing and Suffix trees because it allows complete pattern matches instead of simple word-based search, transforming the functionalities as compressed suffix array structures. Pattern location through inverted files is more efficient and faster during the usage of equivalent memory-based structures.

2.2 Experimental Result

The execution background is all set through the installation of the necessary IDE for developments through the java programming language. The algorithm provided in the design and implementation ensures meeting the objectives. Since our experiment is in progress, we have included screenshots in Fig.3, Fig.4, and Fig.5. These screenshots lead to achieving the desired outcome. [1] The system built can upload and process the inputs from different languages employed. The Fig.3. Depicts LOGIN to the MLIR Search Engine.

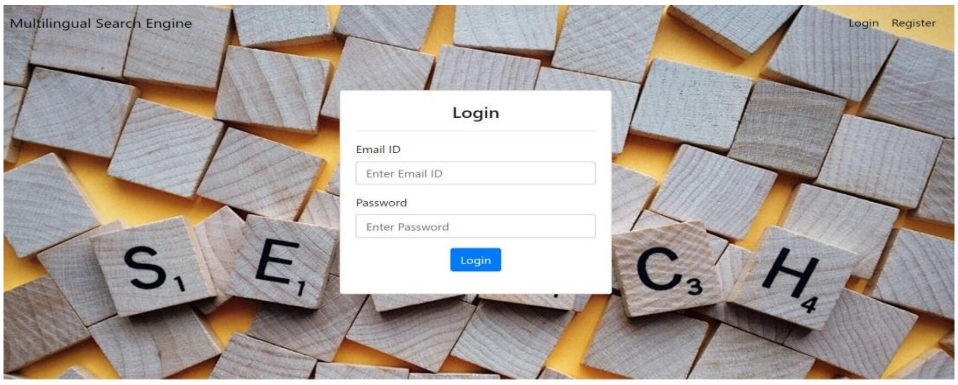


Fig.3. Screenshot for log in to the MLIR Framework

The Fig.4. Depicts entering the keyword to be searched related to the document, in the proposed system three different algorithms have been used to search the content based on the entered keyword and these algorithms are analyzed based on the execution time.

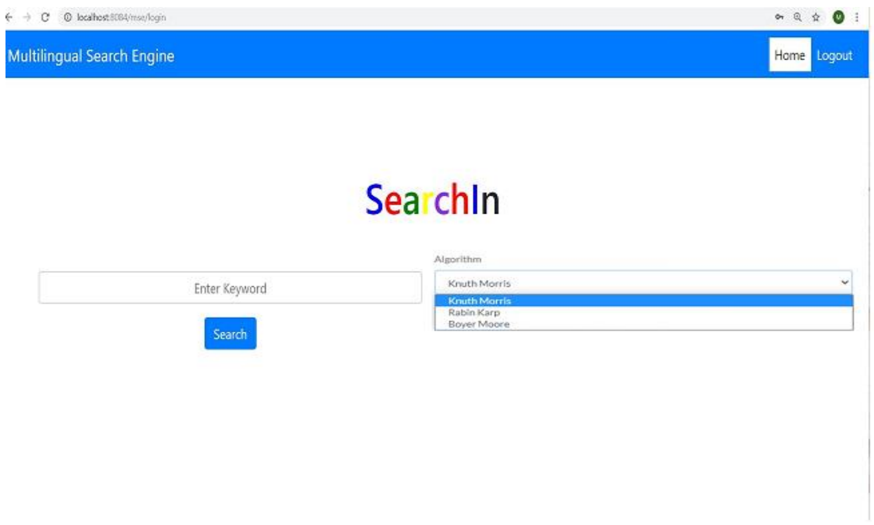
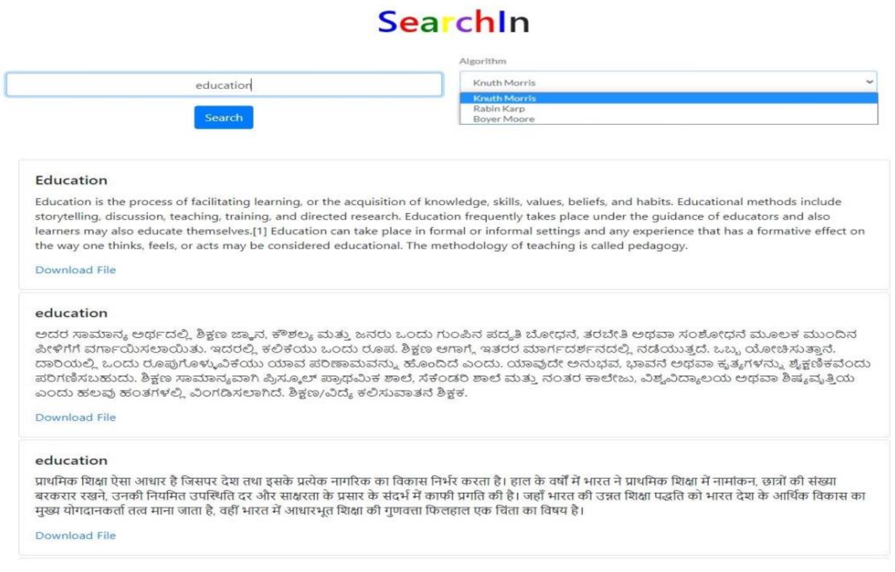


Fig.4. Searching the content based on the keywords and Algorithm selection

The system is capable of translating all the search queries into different languages we have deployed in the design. The system translates all search queries into the various languages based on the user's preferences as illustrated in Fig .5 and Fig .6. Our MLIR aims is designed to work on three languages English, Hindi, and Kannada.



Education

Education is the process of facilitating learning, or the acquisition of knowledge, skills, values, beliefs, and habits. Educational methods include storytelling, discussion, teaching, training, and directed research. Education frequently takes place under the guidance of educators and also learners may also educate themselves.[1] Education can take place in formal or informal settings and any experience that has a formative effect on the way one thinks, feels, or acts may be considered educational. The methodology of teaching is called pedagogy.

Download File

education

ಅದರ ಸಾಮಾನ್ಯ ಅರ್ಥದಲ್ಲಿ, ಶಿಕ್ಷಣ ಜ್ಞಾನ, ಕೌಶಲ್ಯ ಮತ್ತು ಜನರು ಒಂದು ಗುಂಪಿನ ಪದ್ಧತಿ ಬೋಧನೆ, ತರಬೇತಿ ಅಥವಾ ಸಂಶೋಧನೆ ಮೂಲಕ ಮುಂದಿನ ಪೀಳಿಗೆಗೆ ವರ್ಗಾಯಿಸಲಾಯಿತು. ಇದರಲ್ಲಿ, ಕಲಿಕೆಯು ಒಂದು ರೂಪ. ಶಿಕ್ಷಣ ಆಗಾಗ್ಗೆ ಇತರರ ಮಾರ್ಗದರ್ಶನದಲ್ಲಿ ನಡೆಯುತ್ತದೆ. ಒಬ್ಬ ಯೋಚಿಸುತ್ತಾನೆ. ದಾರಿಯಲ್ಲಿ, ಒಂದು ರೂಪುಗೊಳ್ಳುವಿಕೆಯು ಯಾವ ಪರಿಣಾಮವನ್ನು ಹೊಂದಿದೆ ಎಂದು. ಯಾವುದೇ ಅನುಭವ, ಭಾವನೆ ಅಥವಾ ಕೃತ್ಯಗಳನ್ನು ಶೈಕ್ಷಣಿಕವೆಂದು ಪರಿಗಣಿಸಬಹುದು. ಶಿಕ್ಷಣ ಸಾಮಾನ್ಯವಾಗಿ ಪ್ರಿಸ್ಕೂಲ್ ಪ್ರಾಥಮಿಕ ಶಾಲೆ, ಸೆಕೆಂಡರಿ ಶಾಲೆ ಮತ್ತು ನಂತರ ಕಾಲೇಜು, ವಿಶ್ವವಿದ್ಯಾಲಯ ಅಥವಾ ಶಿಷ್ಯವೃತ್ತಿಯ ಎಂದು ಹಲವು ಹಂತಗಳಲ್ಲಿ ವಿಂಗಡಿಸಲಾಗಿದೆ. ಶಿಕ್ಷಣ/ವಿದ್ಯೆ ಕಲಿಸುವಾತನೆ ಶಿಕ್ಷಕ.

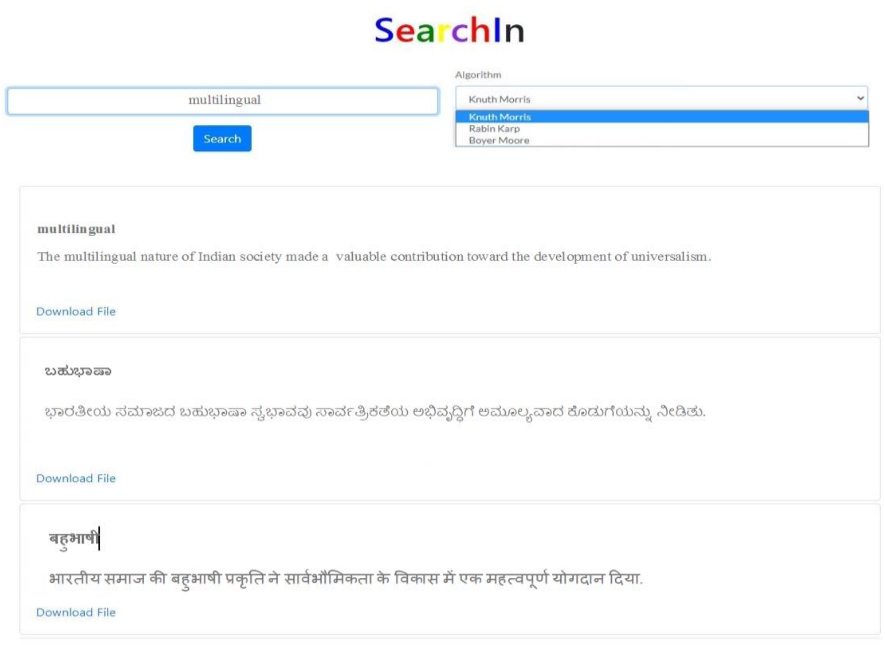
Download File

education

प्राथमिक शिक्षा ऐसा आधार है जिसपर देश तथा इसके प्रत्येक नागरिक का विकास निर्भर करता है। हाल के वर्षों में भारत ने प्राथमिक शिक्षा में नामांकन, छात्रों की संख्या बरकरार रखने, उनकी नियमित उपस्थिति दर और साक्षरता के प्रसार के संदर्भ में काफी प्रगति की है। जहाँ भारत की उन्नत शिक्षा पद्धति को भारत देश के आर्थिक विकास का मुख्य योगदानकर्ता तत्व माना जाता है, वहीं भारत में आधारभूत शिक्षा की गुणवत्ता फिलहाल एक चिंता का विषय है।

Download File

Fig.5. Results for the searched keywords



multilingual

multilingual

The multilingual nature of Indian society made a valuable contribution toward the development of universalism.

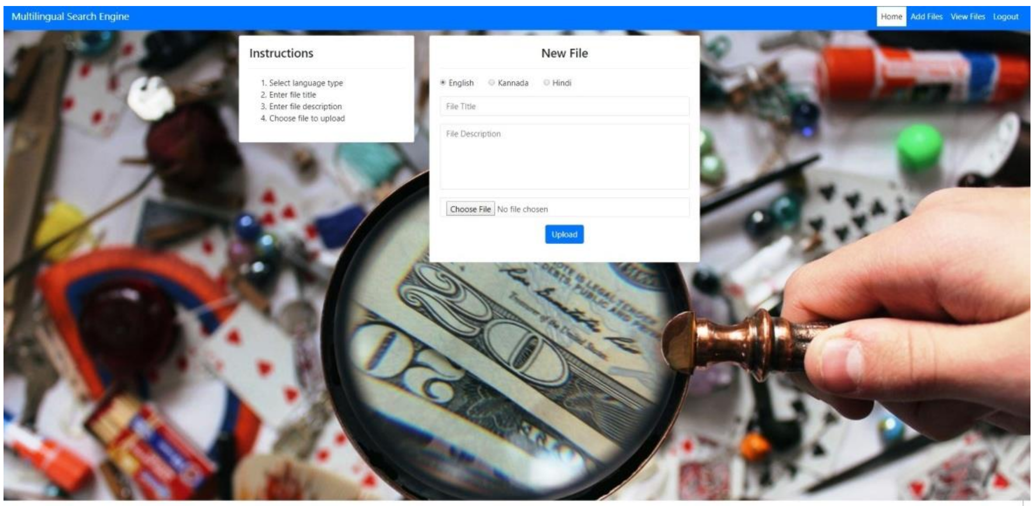
ಬಹುಭಾಷಾ

ಭಾರತೀಯ ಸಮಾಜದ ಬಹುಭಾಷಾ ಸ್ವಭಾವವು ಸಾರ್ವತ್ರಿಕತೆಯ ಅಭಿವೃದ್ಧಿಗೆ ಅಮೂಲ್ಯವಾದ ಕೊಡುಗೆಯನ್ನು ನೀಡಿತು.

बहुभाषी

भारतीय समाज की बहुभाषी प्रकृति ने सार्वभौमिकता के विकास में एक महत्वपूर्ण योगदान दिया.

Fig.6. Results for the searched keywords



Instructions

1. Select language type
2. Enter file title
3. Enter file description
4. Choose file to upload

Fig.7. Screenshot for uploading the Files into the corpus by the Admin.

Fig.6 depicts uploading the document to the corpus.This page, provides the option to selecting the language, giving title to the document, description of the document, and also provides the instruction for uploading the document to the corpus.

3. Pattern matching Algorithms

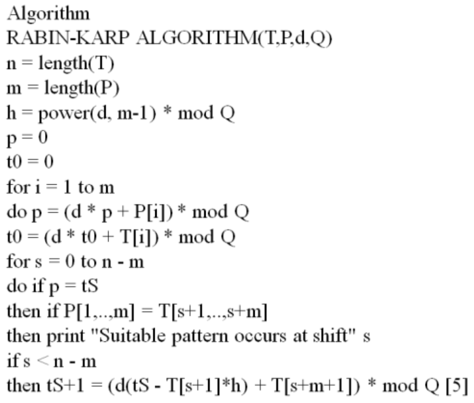
String algorithms are effective in detecting appropriate content within a minimum time. Various research teams are deployed to come up with software and hardware designs to ensure quick pattern searching. By applying various algorithms in different applications approximate best algorithm is determined.[4] Pattern matching is a methodology to verify the apparent sequence of strings to detect the occurrence of the elements of some patterns. Similar to pattern recognition, the matching should be identical. The patterns have the form sequences of pattern matching providing the occurrence of a pattern within a sequence of string. So that output is obtained as some component of the matched pattern[7].

In the proposed Architectural design following algorithm has been used for searching requested queries.

1) Rabin-Karp Algorithm
2) Knuth–Morris–Pratt algorithm
3) Boyer–Moore string search algorithm

Rabin-Karp Algorithm

Rabin-Karp Algorithm works based on the hashing technique. R-K algorithm utilizes hashing value for matching and compares from left to right. Rabin-karp string matching algorithm works using modular hashing and is the first to introduce the concept of hashing in the string matching process. [10] It is similar to brute force comparison except it improves the speed of comparison. The first step is to calculate the hash value of the given pattern. It makes the window of size length of the pattern, and this window is made movement right to the text each time when hash values become unequal. The second step is to compute the hash value of characters inside the window. Then the algorithm iterates through the text string. If hash values of pattern and window become equal then only it starts comparison of each character in the window with each character of pattern and if all the characters of window matches with the characters of the pattern then return the position of the pattern in the text. If characters mismatch then it stops comparison and moves the window to the right by one character and continues the above process. [5]



```
Algorithm
RABIN-KARP ALGORITHM(T,P,d,Q)
n = length(T)
m = length(P)
h = power(d, m-1) * mod Q
p = 0
t0 = 0
for i = 1 to m
do p = (d * p + P[i]) * mod Q
t0 = (d * t0 + T[i]) * mod Q
for s = 0 to n - m
do if p = tS
then if P[1,..,m] = T[s+1,..,s+m]
then print "Suitable pattern occurs at shift" s
if s < n - m
then tS+1 = (d(tS - T[s+1]*h) + T[s+m+1]) * mod Q [5]
```

The pseudocode employed in the RK algorithm receives the text T, Search pattern P, Q, and d, where d denotes |Σ| and is given in the program. The function power(d,m-1) represents d m−1 and mod denotes the regular modulo function. The initial for loop is used to preprocess and consumes Θ(m) time for the first process. The second for loop is used for matching strings and consumes Θ((n−m+1)m) time in the worst case. If P = am and T = an, then the verifications take time Θ((n − m + 1)m), as the possible verifications have a valid shift.[5]

Knuth–Morris–Pratt algorithm

Knuth-Morris algorithm matches the characters of pattern and text from left to right. In Comparison with BF algorithm, the KMP algorithm exhibits improved pattern matching characteristics. The algorithm was developed by Donald Knuth and Vaughan Pratt, and further by James H. Morris, thus the name KnuthMorris-Pratt operation. [11]. It works based on prefix and suffix match within the given pattern. Compare each character of text with each character of the pattern, if all the characters of pattern matched with the text substring of length pattern, then return the starting position of text string where a pattern exists. If there is no match of a particular character then find a substring in the pattern which must be a suffix as well as a prefix in that substring. If no found then compare the next character of text with starting character of the pattern and continue the process. If suffix and prefix found then compare the next character of text with the next character immediately after the prefix substring and continue the process.

This method avoids backward movement for comparison and also reduces time complexity. To summarize, a pattern denoted by P[1, . . . , m], the prefix function of the pattern P is given by function $\pi : \{1, 2, . . . , m\} = \{0, 1, . . . , m - 1\}$ such that $\pi[q] = \max\{k : k \leq q$ and Pk is a proper suffix of Pq$\}$. Unofficially, $\pi[q]$ is the length of the longest prefix of P which is Pk and q is the current character in the text T. Here is a description of the pseudocode.[5]

```
Algorithm
KNUTH-MORRIS-PRATT ALGORITHM (P,T)
m=length(P)
n=length(T)
pi=compute prefix function(P)
q=0
for i=1 to n
do while q>0 and P[q+1] != T[i]
do q=pi[q]
if P[q+1] = T[i]
then q=q+1
if q=m
then print "The pattern occurs with shift" i-m
q=pi[q]
The prefix function in pseudocode is:
COMPUTE PREFIX FUNCTION(P)
m = length(P)
pi[1] = 0
k = 0
for q = 2 to m
do while k > 0 and P[k+1] != P[q]
do k = pi[k]
if P[k+1] = P[q]
then k = k+1
pi[q] = k
return pi[5]
```

Boyer–Moore: String searching algorithm

Boyer-Moore algorithm compares the letters from right to the left side for the patterns against the text in the same direction as like-pattern, starting with the index equal to the length of pattern-1. It matches the tail of the pattern rather than the head. It gives better results in less time when the alphabet is reasonably sized and the pattern is reasonably lengthy.
The algorithm is based on some shift mappings or functions that establish the mapping of the shifting index according to matching characters or occurrence of the mismatch.[7]

The pseudocode of the algorithm comprises of 2 functions and the main function for computation of the good suffix and a function for final occurrence computations. [5] The algorithm monitors the pattern and characters in it from right to the left starting from the rightmost one. For a deviation or a complete-match involved for a whole pattern, two pre-computed functions are used for shifting the position of the window to the right.

Here is a description of the two shifts functions-

• The good suffix shift or matching shift: This ensures alignment of the pattern characters matched against the target characters which are already been matched.

• The bad character shift or occurrence shift: This avoids the repetition of ineffective comparisons with destination characters. [8]

```
Algorithm
BOYER-MOORE ALGORITHM(P,T,Sigma)
m = length(P)
n = length(T)
lambda = compute last occurrence(P,m,Sigma)
gamma = compute good suffix(P,m)
s = 0
while s <= n-m
do j = m
while j>0 and P[j] = T[s+j]
do j = j-1
if j = 0
then print "The pattern occurs at shift" s
s = s + gamma[0]
else s = s + max(gamma[j],lambda[T[s+j]])
COMPUTE LAST OCCURRENCE(P,m,Sigma)
for each a in Sigma
do lambda[a] = 0
for j = 1 to m
do lambda[P[j]] = j
return
COMPUTE GOOD SUFFIX(P,m)
pi = compute prefix function(P)
P' = reverse(P)
pi' = compute prefix function(P')
for j = 0 to m
do gamma[j] = m- pi[m]
for l = 1 to m
do j = m - pi'[l]
if gamma[j] > l - pi'[l]
then gamma[j] = l - pi'[l]
return.[5]
```

3.1 Analysis of pattern matching Algorithm

In this paper, we have selected three different String matching algorithms and analyzed them based on time complexity and Execution time. When the time complexity for the Rabin Karp algorithm is analyzed O(n +m) represents the best case, but the worst-case time complexity of Rabin Karp algorithm is O(nm). Finding the best-time complexity of the Boyre Moore Algorithm is O(n/m) and the worst-case time complexity is O(nm). The best time complexity of the Knuth Morris Algorithm is O(n) and the worst-case time complexity is O(n).[6]

The best algorithm for different applications is determined by using them in different scenarios. Most applications use Boyer Moore, RK or KMP algorithms for their effective functionality and other applications employ the fundamentals of all aforementioned algorithms to determine their functions.[9]

Sl.No	Execution Time			
	Pattren	**Rabinn Karp**	**Knuth Morris**	**Boyer Moore**
1	Education	2013ms	830ms	2949 ms
2	acquisition	968ms	275 ms	458 ms
3	storytelling	601ms	353 ms	1030 ms
4	knowledge	5113ms	378 ms	936 ms
5	learning	2302ms	435 ms	564 ms
6	research	707ms	303 ms	551 ms

Table.1: Execution time of all the three Algorithms used for searching small patterns in MLIR Framework

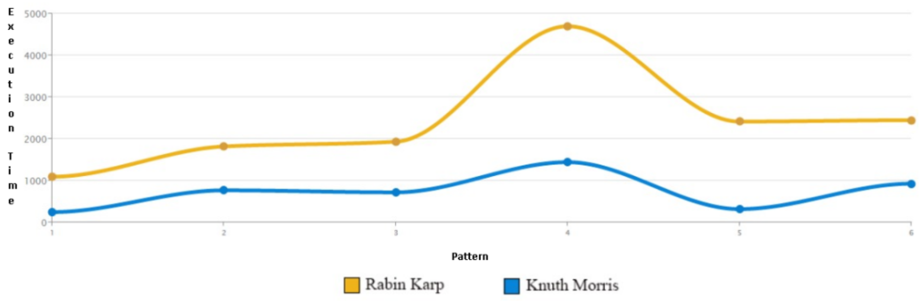


Fig.8. Screenshot to compare the Execution time of R-K and KMP Algorithm.

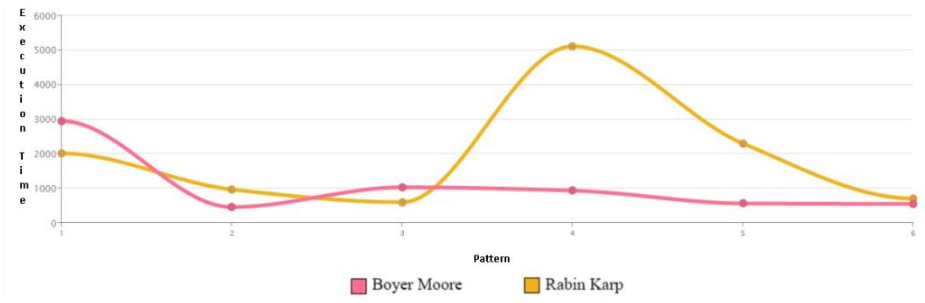


Fig.9. Screenshot to compare the Execution time of R-K and BM Algorithm.

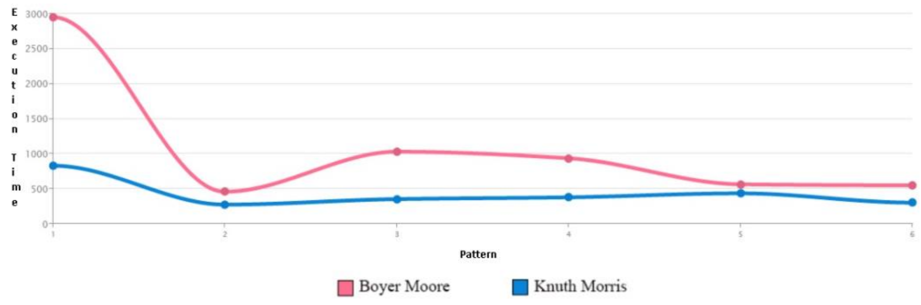


Fig.10. . Screenshot to compare the Execution time of BM and KMP Algorithm.

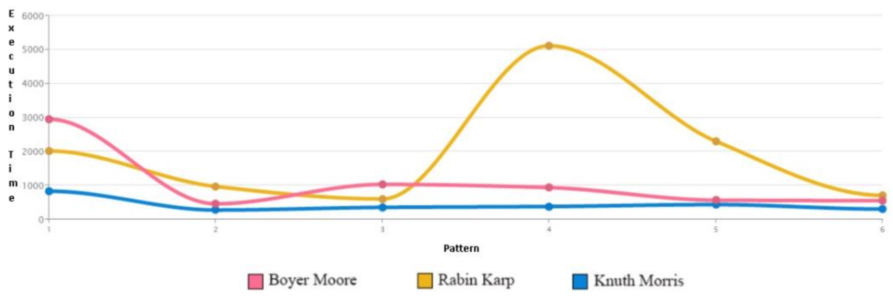


Fig.11. . Screenshot to compare all the three Algorithms

. Table.1 Shows the Execution time of R-k, KMP, and BM algorithms for smaller patterns. Fig.8 shows the chart for the execution time of the KMP algorithm and R-k algorithm and in this chart, we can easily analyze that KMP executes faster than R-K algorithm in searching the pattern. Fig.9 shows the chart for the execution time of BM algorithm and R-k algorithm and this graph shows that BM executes faster than R-K algorithm in searching the pattern. Fig.10 shows the chart for the execution time of BM algorithm and KMP algorithm and this graph shows that BM executes faster than KMP algorithm. Fig.11 chart shows the comparison of all three algorithms. By analyzing above mentioned graphs inferred that KMP algorithm is faster in searching the smaller patterns compare to the R-K and BM Algorithm.

Sl.No	Execution Time			
	Pattern	Rabinn Karp	Knuth Morris	Boyer Moore
1	Education is the Process	1093ms	245ms	435ms
2	process of facilitating	1817ms	766ms	295ms
3	facilitating learning	1913ms	719ms	806ms
4	acquisition of knowledge	4695ms	1440ms	1355ms
5	Educational methods	2413ms	317ms	539ms
6	directed research	2446ms	926ms	563ms

Table.2: Execution time of all the three Algorithms used for searching long patterns in MLIR Framework

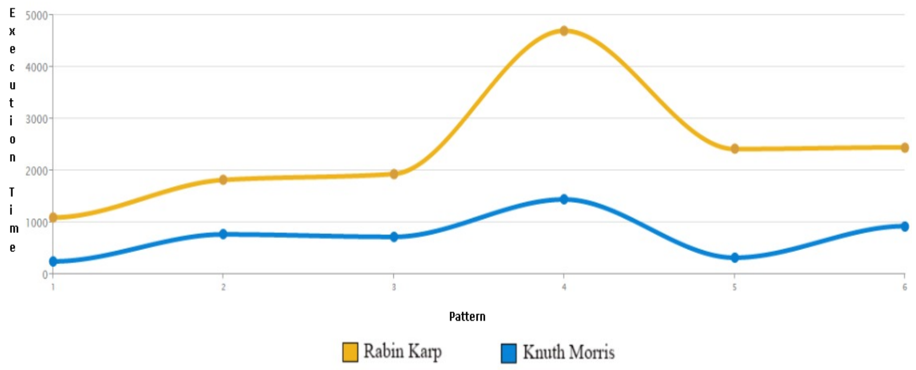


Fig.12. Screenshot of the chart for comparing the Execution time of R-K and KMP Algorithm.

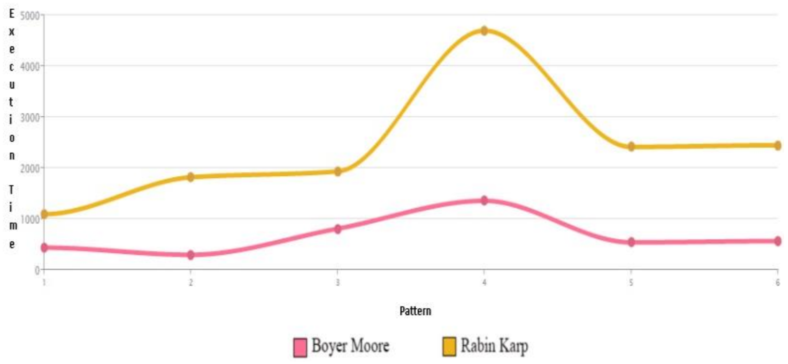


Fig.13. Screenshot of the chart for comparing the Execution time of R-K and BM Algorithm

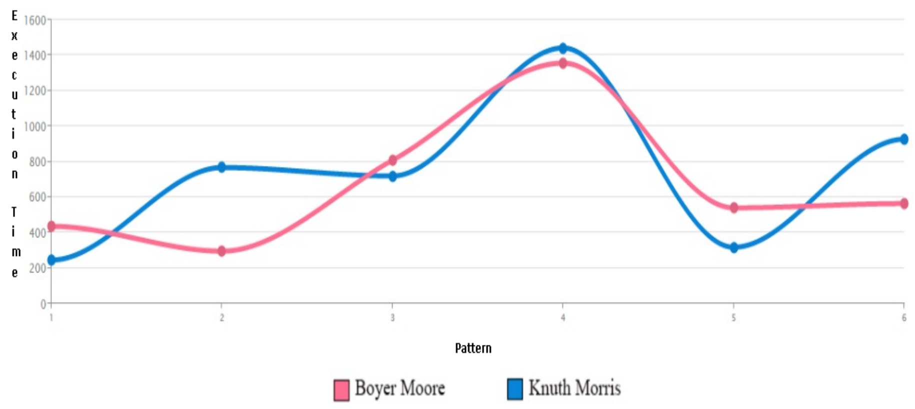


Fig.14. Screenshot of the chart for comparing the Execution time of BM and KMP Algorithm.

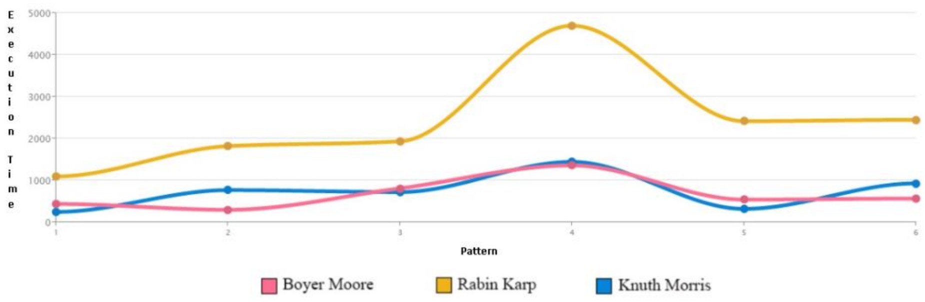


Fig.15. Screenshot of the chart for comparing all the three Algorithms.

Table.2 Shows the Execution time of R-k, KMP, and BM algorithms for bigger patterns. Fig.12 shows the chart for the execution time of KMP algorithm and R-k algorithm and Fig.13 shows the graph for the execution time of BM algorithm and R-k algorithm and these charts show that R-K algorithm is slower in searching the pattern compared to the BM and KMP. Fig.14 illustrates the chart for the execution time of BM algorithm and KMP algorithm and this graph shows that BM algorithm is slightly faster than KMP algorithm. Fig.15 graph shows the comparison of all three algorithms. By analyzing above mentioned graphs inferred the BM algorithm is faster in searching for the bigger patterns compare to the KMP and RK Algorithm. But, the R-K algorithm has limitations, a distinct set of characters with identical hash values is one among them, accordingly, a fake hit occurs, leading to a disparity, [7] and it is slower in searching patterns compare to the other two algorithms.

4 Conclusion

The research work is based on the principles of the semantic-based methodology. Particularly it works for multi-lingual information retrieval. Three languages used in the work are English, Kannada, and Hindi. A Multilingual Information Retrieval System handles inquiry in a solo language and retrieves the records in different dialects. The system is developed using multilingual lexicon-based word-by-word translation of queries. Languages like Hindi and Kannada are used for producing the outcome. The algorithms allow us to examine keywords in the documents of diverse dialects. In the proposed work three different algorithms are used for scanning the entered keywords in the corpus and compared all the three algorithms by considering the execution time for smaller patterns and longer patterns, by analyzing these graphs of algorithms it can be concluded that KMP algorithm efficient for smaller strings and BM works slightly better for long search patterns compare to KMP algorithm. The work is envisioned to cater to the needs of documents in different Indian dialects and also constituting big-word orientations.

References

1. Chayapathi, A.R., Kumar, G.S., Thriveni, J. and Venugopal, K.R., 2021. Usage of Multilingual Indexing for Retrieving the Information in Multiple Language. In *Advances in Machine Learning and Computational Intelligence* (pp. 255-264). Springer, Singapore.
2. Xu, X., Pan, S. and Wan, J., 2010, May. Compression of the inverted index for comprehensive performance evaluation in Lucene. In *2010 Third International Joint Conference on Computational Science and Optimization* (Vol. 1, pp. 382-386). IEEE.

3. Chayapathi A R, G Sunilkumar, Manjunathswamy B E, Thriveni J, Venugopal KR, 2020, April, NLP-based Stemming and Lemmatization approaches for Multilingual Search Indexing. International Journal of Advanced Science and Technology, Vol. 29, No. 6, (2020), pp. 9121 - 9134.
4. Singla, N. and Garg, D., 2012. String matching algorithms and their applicability in various applications. *International journal of soft computing and engineering*, *1*(6), pp.218-222.
5. Lumburovska, L., 2018. *Time-Efficient String Matching Algorithms and the Brute-Force Method* (Doctoral dissertation, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko).
6. Rasool, A., Tiwari, A., Singla, G. and Khare, N., 2012. String matching methodologies: A comparative analysis. *REM (Text)*, *234567*(11), p.3.
7. Sheikh, N.U., Rahman, H. and Al-Qahtani, H., 2019. An Optimized Pattern Recognition Algorithm for Anomaly Detection in IoT Environment. *arXiv preprint arXiv:1901.08729*.
8. SS.Swapna, Yashdeep Jha, Syed Zaheed, Keertik Dewangan,Sayyed Mujahid Pasha, A Survey on Different Pattern Matching Algorithms of Various Search Engines. International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 6, Issue 10, October 2019.
9. Minal Suthar, Amit Patel, Shivali Shah,, 2015. A Survey Paper on String Matching. International Journal for Scientific Research & Development, Vol. 3, Issue 05, 2015 , ISSN (online): 2321-0613
10. Hakak, S.I., Kamsin, A., Shivakumara, P., Gilkar, G.A., Khan, W.Z. and Imran, M., 2019. Exact string matching algorithms: Survey, issues, and future research directions. *IEEE Access*, *7*, pp.69614-69637.
11. Zhou, Y. and Pang, R., 2019, December. Research of Pattern Matching Algorithm Based on KMP and BMHS2. In *2019 IEEE 5th International Conference on Computer and Communications (ICCC)* (pp. 193-197). IEEE.