iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** V **Month of publication:** May 2021

DOI: https://doi.org/10.22214/ijraset.2021.34561

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

WEB APP: String Similarity Search - A Hash-based Approach

Snehal Bobhate[1], Snehali Shendre[2], Nidhi Joshi[3], Sneha Ghodeswar[4], Dr. (Mrs.) Y. A. Nafde[5]
[1, 2, 3, 4]*Student,* [5]*Guide, Department of Electronics & Telecommunication, Priyadarshini College of Engineering, Nagpur, Maharashtra.*

Abstract: ***During this Project, we study string similarity search based on edit distance that is supported by many database management systems like Oracle and PostgreSQL. Given the edit distance, ed(s, t), between two strings, s and t, the string similarity search is to search out each string t in a string database D which is almost like a query string s such that ed(s, t) = t for a given threshold t. Within the literature, most existing work takes a filter-and-verify approach, where the filter step is introduced to reduce the high verification cost of 2 strings by utilizing an index engineered offline for D. The two up-to-date approaches are prefix filtering and native filtering. We have a tendency to propose 2 new hash- primarily based labeling techniques, named OX label and XX label, for string similarity search. We have a tendency to assign a hash-label, H s , to a string s, and prune the dissimilar strings by comparing 2 hash-labels, H s and H t , for two strings s and t within the filter step. The key idea is to take the dissimilar bit- patterns between 2 hash-labels.Our hash-based mostly approaches achieve high efficiency, and keep its index size and index construction time one order of magnitude smaller than the present approaches in our experiment at the same time.***

I. INTRODUCTION

1) SiSelection and join are two important operations in tradional databased to help users query the data.The underlying data, however, are rather dirty in real world, due to the typographical errors and data inconsistencies.
2) The exact selection and joining operations cannot return answers if the data is not exactly matched.
3) To address this problem, approximate selection and join are proposed to extend exact selection and exact join by tolerating errors and inconsistencies.
4) And they have many real world applications ,including data cleaning , data integration and data clustering.
5) Given a set of objects and query objects, similarity search aims to find objects to the query. String Similarity Search has many real world applications, such as data cleaning and spell checking .
6) For example, in spell checking, given a set of words and a query word ,string similarity search wants to find similar words to the query.
7) Many information's system utilize this feature to enhance the usability and provide user-friendly functionalities. e.g., Microsoft word, Gmail, Google search engine.

String similarity search is a fundamental query that has been widely used for DNA sequencing, error-tolerant query auto completion, and data cleaning needed in database, data warehouse, and data mining. In this paper, we study string similarity search based on edit distance that is supported by many database management systems such as Oracle and PostgreSQL. Given the edit distance, between two strings, s and t, the string similarity search is to find every string t in a string database D which is similar to a query string s such that t for a given threshold t. In the literature, most existing work takes a filter-and-verify approach, where the filter step is introduced to reduce the high verification cost of two strings by utilizing an index built offline for D. The two up-to-date approaches are prefix filtering and local filtering. In this paper, we study string similarity search where strings can be either short or long. Our approach can support long strings, which are not well supported by the existing approaches due to the size of the index built and the time to build such index. We propose two new hash- based labeling techniques, named OX label and XX label, for string similarity search. We assign a hash-label, Hs, to a string s, and prune the dissimilar strings by comparing two hash-labels, Hs and Ht, for two strings s and t in the filter step. The key idea is to take the dissimilar bit-patterns between two hash-labels. We discuss our hash-based approaches, address their pruning power, and give the algorithms. Our hash-based approaches achieve high efficiency, and keep its index size and index construction one order of magnitude smaller than the existing approaches in our experiment at the same time.

A. What is Similarity?

A word frequently used when discussing Approximate Matching is similarity. Investigators may use Approximate Matching to discover the presence of files similar to something we already know. So what is similarity and how do we measure it? There are essentially two different ways in which two files can be similar: syntactic and semantic. Syntactic similarity is from the perspective of a computer, and semantic similarity is from the perspective of a human. Two documents are semantically identical if they communicate the same information. For example, a Microsoft PowerPoint presentation is semantically identical to an exported PDF document containing the same pages. Their cryptographic hashes6 will not be identical, though we can still argue that the documents are the same. A similar concept applies to media, like pictures and videos. Most people would consider two pictures to be identical even though they are stored in two different formats. The meaning of two documents can also be identical even though the contents are presented in two different formats, e.g., the same set of numbers represented as a table or as list.
Semantically similar documents do not need to be similarly represented on a hard drive, but when presented they will appear similar to a human. Syntactically identical documents are represented identically on a hard drive and h(A) == h(B), where A and B are the two documents, and h is a crystallographic hash function.

B. Approximate Matching

In order to understand how Approximate Matching can be applied during digital investigations, there is a need to review some theory on the types of Approximate Matching.

C. Hash Based Similarity

Hash based matching measures the syntactical similarity between two files, not by interpreting the perceptual similarity, but by evaluating byte level commonalities in data. Due to the fact that two pictures can look identical but have different encoding and therefore be very different on byte level, AHBM is not suited for the task of measuring similarity between pictures. The benefit of measuring similarity on byte level is that it enables measurement of unknown content types and therefore allows Approximate Matching of complex and unstructured data such as documents, memory images and network packets.
Although encoding makes hash based matching unsuited for measuring similarity in images and videos, it can still provide great value when analyzing fragments from memory or deleted files. A fragment of a picture recovered from memory or unallocated space may be identical to the corresponding fragment in a picture found elsewhere. The same concept applies when looking for traces of known objects in a memory or disk image. An investigator may for example suspect that a certain software has been installed and later deleted from an hard drive. Approximate hash based algorithms may then be used to create a reference set from this software and match the disk image against this reference set. The reference set will usually contain executables, libraries and other resource used by the target software.
The most significant algorithms for computing hash-based similarity are currently sd hash (Roussev, 2010) and ss deep (Kornblum, 2006). Other algorithms have been proposed, but none have yet gained as much scrutiny in the digital forensics community. Comparisons between the tools are given in Roussev, 2011, Breitinger et al., 2013. Both algorithms look for statistical improbable chunks of bytes within the files. These improbable chunks are called features and are used to compute the similarity between two files. A highly simplified figure of hash based matching algorithms are shown in Fig. 2. In this figure, File A has five features and File B has four features. As File B has the least features, and three of its four features are identical to features in File A, the two files do therefore have similarity score (3/4)*100 = 75. Common output of Approximate Matching tools are similarity scores in a range from 1 to 100.

II. LITERETURE REVIEW

A. Paper[1]

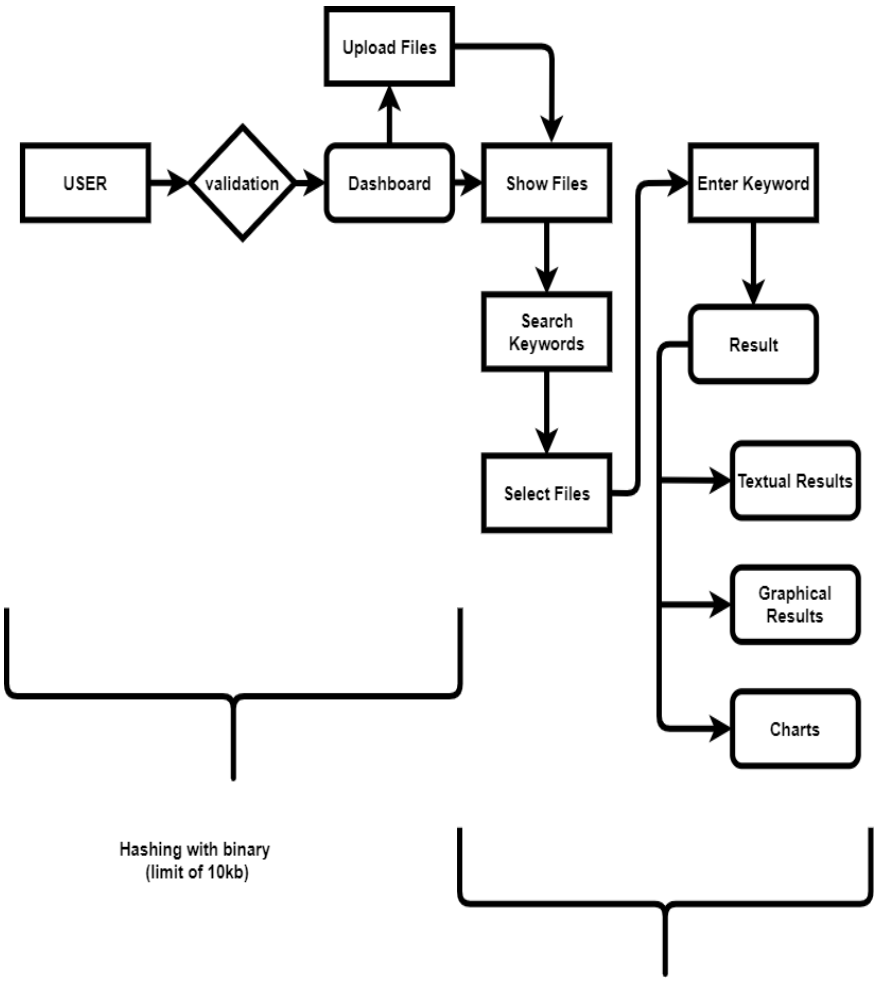
Kholoud Al-Khamaiseh*, Shadi ALShagarin**
(Department of Communication and Electronics and Computer Engineering, Tafila Technical University, 66110, Tafila, Jordan)
** (Computer and Information Technology Center, Tafila Technical University, 66110, Tafila, Jordan)
In order to this survey of string matching algorithm, we observe the means used to answer two types of search models: (a) is a word (depends on the language); (b) is any sequence starting in an index point. In order to these models, the answer models are: Exact match and approximate match respectively. In the remainder of this section, we review the recent updated and hybrid algorithms.

B. Paper[2]

Literature Review of Attribute LEVEL AND Structure Level data Linkage Techniques, Mohammed Gollapalli, College of Computer Science & Information Technology, University of Dammam, Dammam, Kingdom of Saudi Arabia

Data Linkage is an important step that can provide valuable insights for evidence-based decision making, especially for crucial events. Performing sensible queries across heterogeneous databases containing millions of records is a complex task that requires a complete understanding of each contributing database's schema to define the structure of its information. The key aim is to approximate the structure and content of the induced data into a concise synopsis in order to extract and link meaningful data-driven facts.

III. BLOCK DIAGRAM

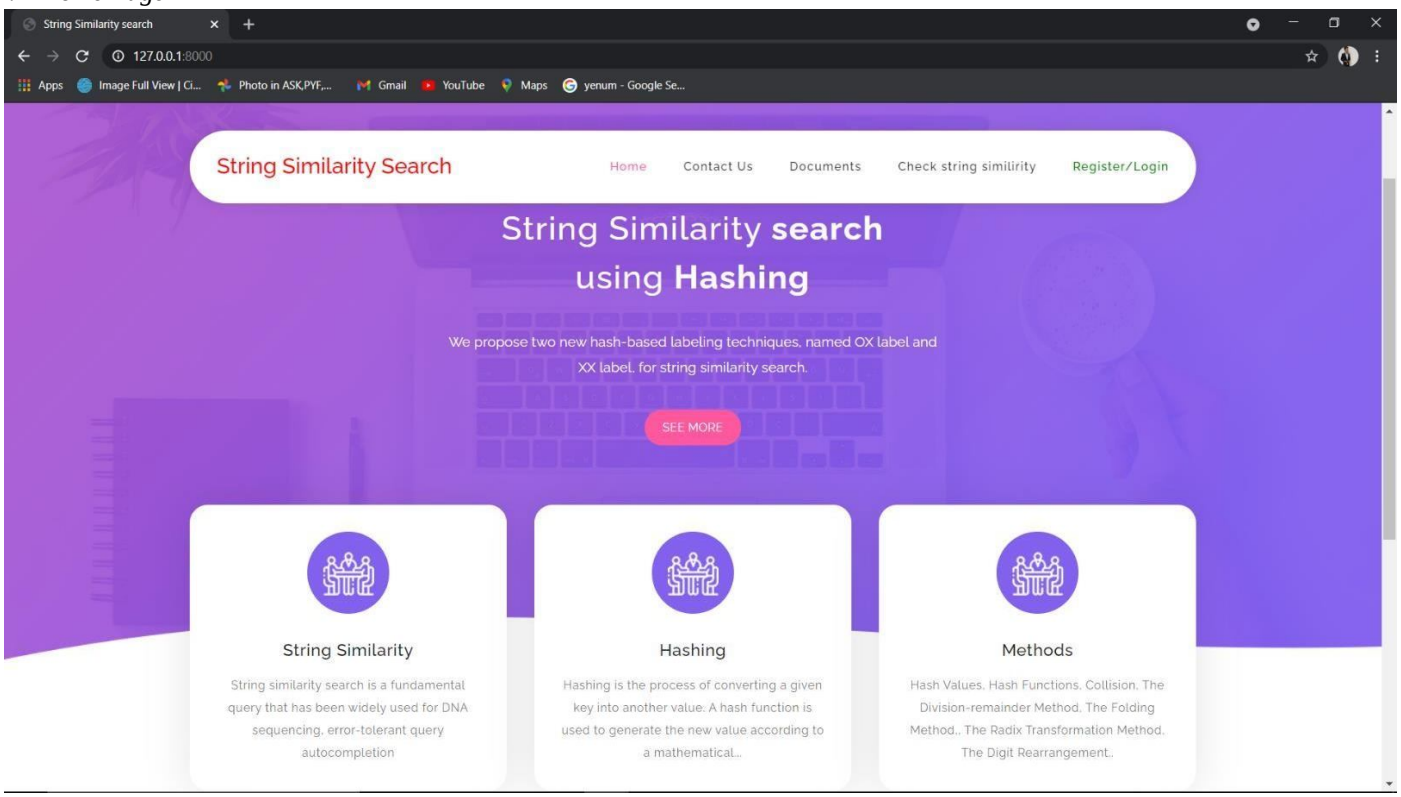


User will register first and then by using credentials user can login

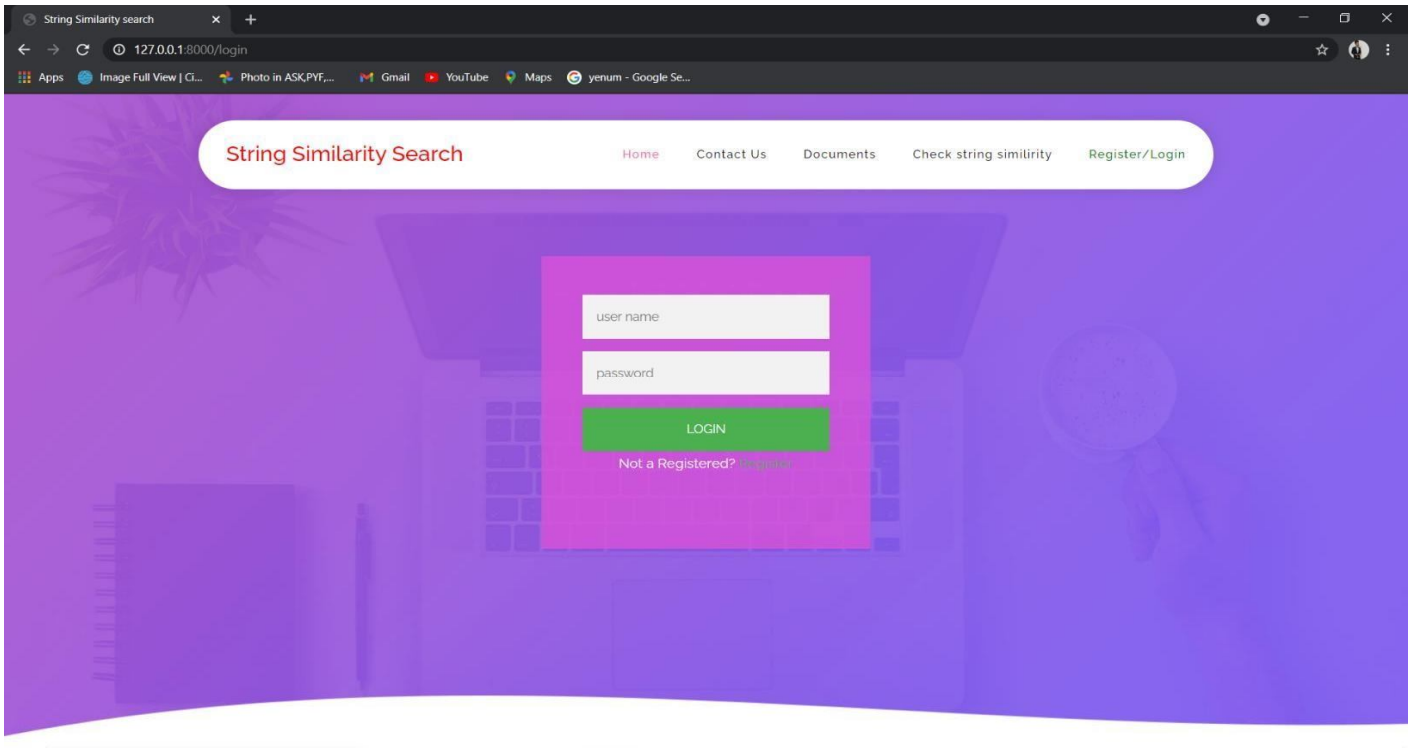
A. In dashboard the user can upload his text file for analysis of similar strings or keywords
B. The file uploaded should be less than 10kb
C. In show file the system will show all the files uploaded by the user
D. The user have to select the files from which user have to make analysis.
E. The user have to enter a keyword or a string for matching
F. By using clustering a machine learning algorithm the typed data will match the similar data in text file
G. The result will be in textual, in graphical analysis and charts

IV. SYSTEM FLOW

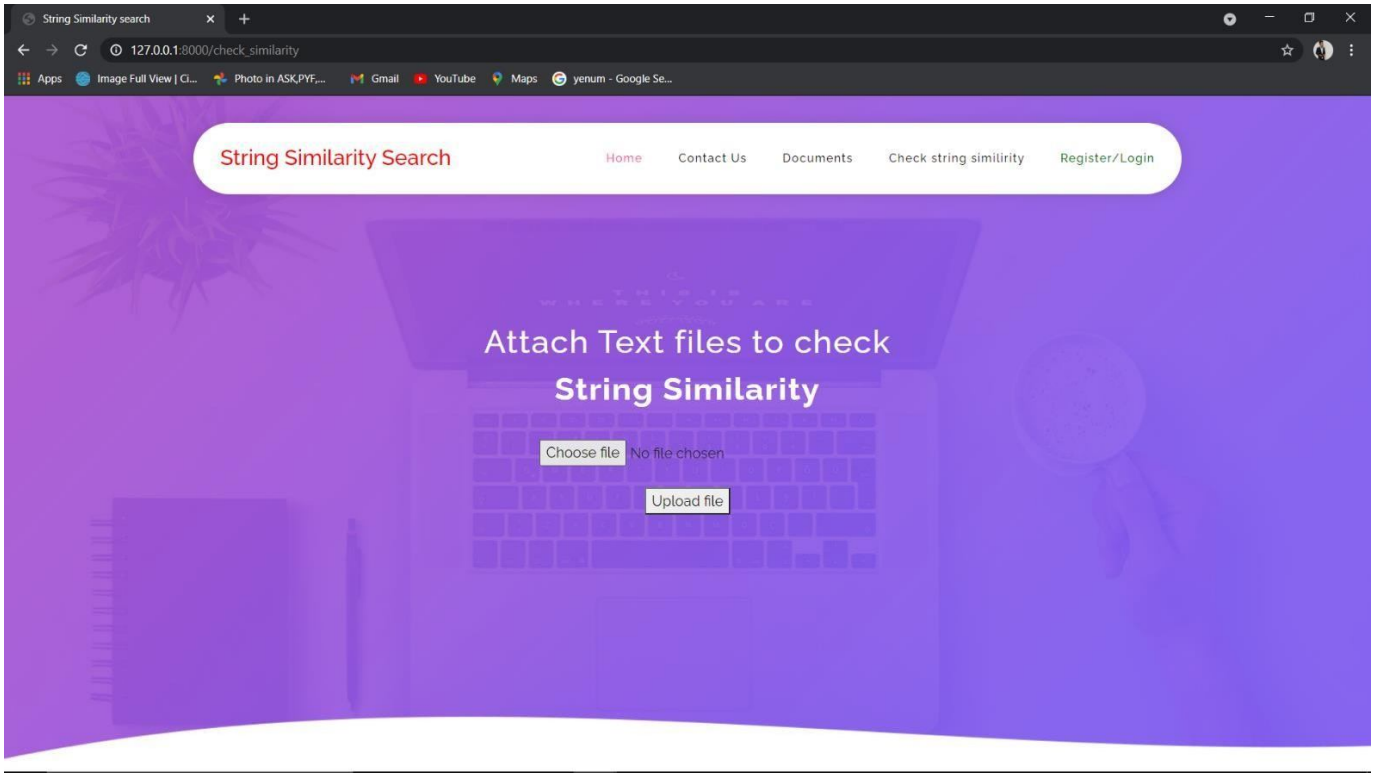
A. *Home Page*



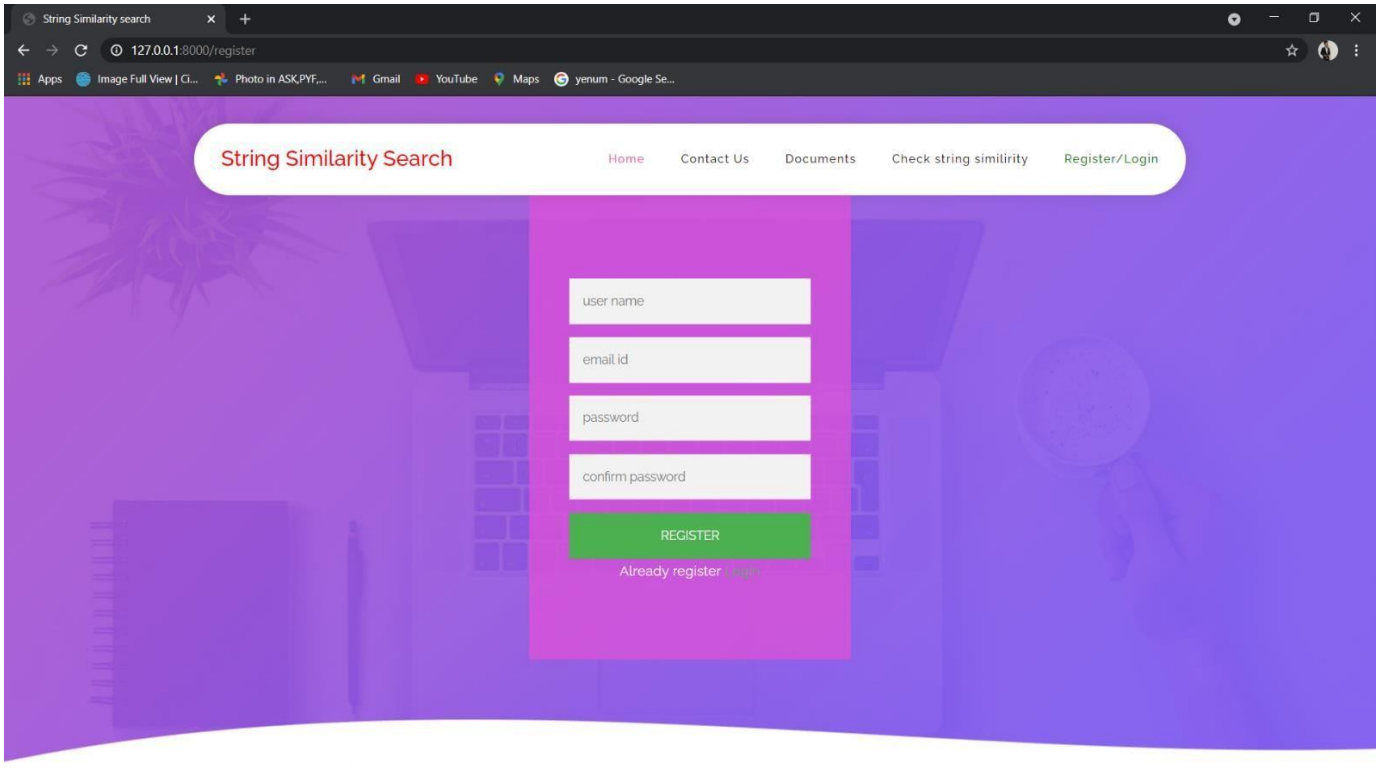
B. *Login Page*



C. Upload Document Page



D. Registration Page



1) Basic Requirements Required

a) Pentium 2 266 MHz processor.20 GB Hard Drive having 3 GB Free Space.
b) 128 MB RAM
c) A Windows Device.

2) Software Required

a) Python(3.6 or higher)
b) Visual Studio Code
c) Djanog Web Framework
d) HTML/CSS, Javascript

V. ADVANTAGES

This project have the advantages of small indexing space, freeness of verification, and computation sharing among strings with common prefixes. The method proposed is a simple adaptation of trie-based error-tolerant prefix matching [30]. Existing trie-based methods process a query by incrementally traversing the trie and maintaining a set of trie nodes (called active nodes) for each prefix of the query. One common drawback is that they have to maintain a large number of active nodes. Instead, Team 8 records only a small number of potentially feasible nodes as ”active nodes” during query processing, which reduces the overhead of maintaining nodes and reporting results. In addition, Team 8 characterizes the essence of edit distance computation by a novel data structure named edit vector automaton, which substantially accelerates the state transition of active nodes, and therefore, improves the total query performance. Naive parallelization is added to exploit multi-core CPUs

A. String similarity search and its variants are fundamental problems with many applications in areas such as data integration, data quality, computational linguistics, or bio-informatics,Low storage space for new retrieved data, Low maintenance of data, Get binary files size wise, Exact matched binary files will be retrieved, Avoid inconsistency data, Reduce data redundancy

B. Approximate search and join operations over large collections of strings are fundamental problems with many applications. String similarity search is used, for instance, to identify entities in natural language texts , to align DNA sequences produced in modern DNA sequencing with sub strings of a reference genome , or to perform pattern matching in time series represented as sequences of symbols .

C. There are many applications in Multilingual and Multi-modal Information Access, the database comprises of data objects with multiple (conditionally independent) views and there is a need for similarity search across the views

D. Also used in Password Security

E. String similarity search and its variants are fundamental problems with many applications in areas such as data integration, data quality, computational linguistics, or bio informatics

VI. FUTURE SCOPE

We addressed the problem of matching two time series for similarity. The similarity model describes a fast search technique to discover all similar subsequences in a set of sequences. Searching a target subsequence requires O(logn) comparisons on an average, where n is the number of subsequences. Thus the search technique is very effective. The highlight of this method is the reduced number of comparisons required for matching source subsequences with the 6 target. It also does no t store the subsequences more than once if they are repeated, thus reducing the storage space. We have been able to demonstrate that this model works well with a variety of time series data. The indexing structure created using kd- tree is stored in the main memory. It is done considering the fact that the cost of the memory is decreasing rapidly. If the data is large and the index structure spills over to secondary memory the performance of the system suffers. We leave a suitable modification to the system that stores the index in the secondary memory as future work.:

VII. C O N CLUSION

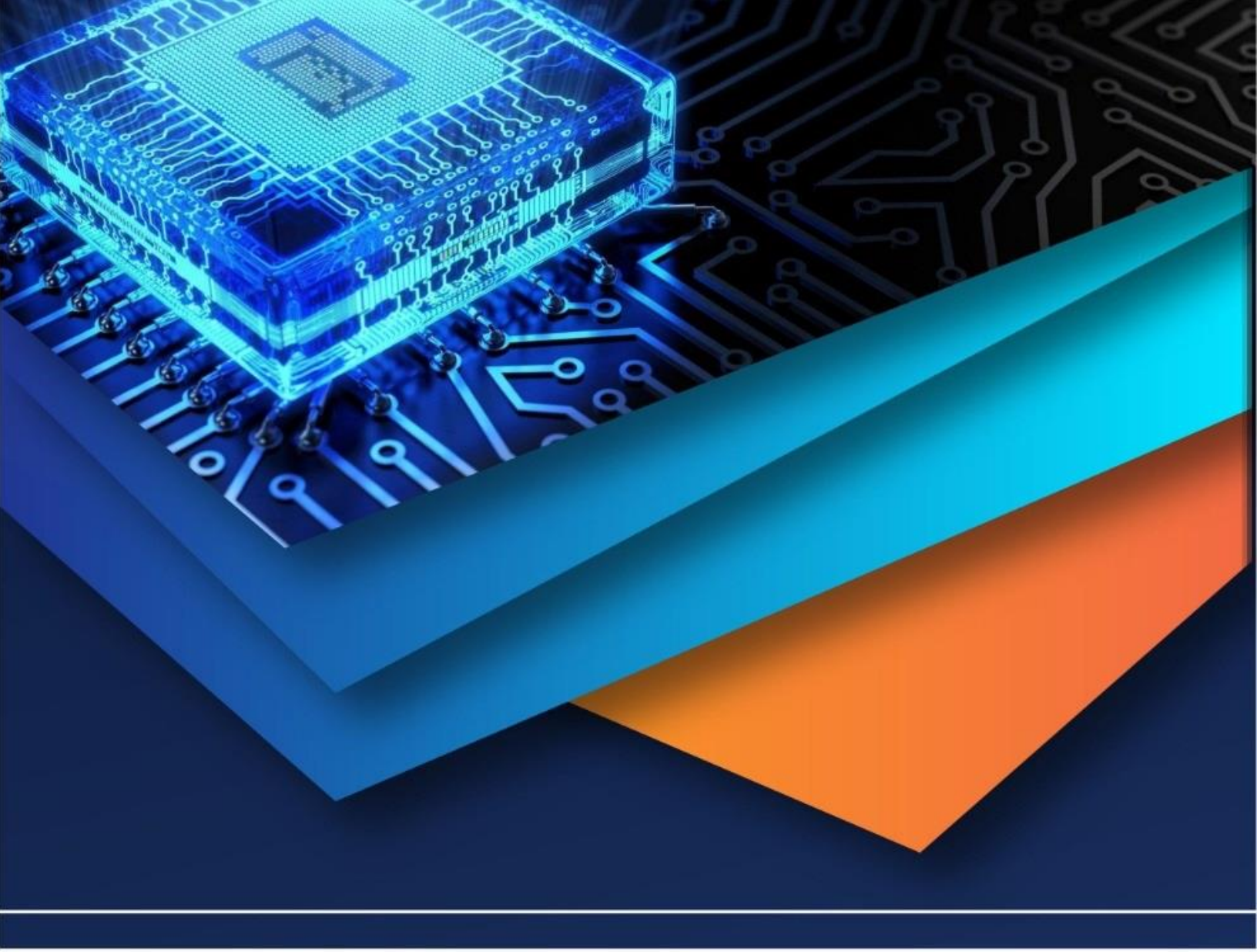
In this paper, we provide a comprehensive survey on string similarity search and join. We formalize the problems of string similarity search and join and other variants. For the string similarity search and join problems, we introduce the filtering- and-verification framework. For similarity search, we introduce the list-merge algorithms. For similarity join, we introduce the prefix filtering technique. We also discuss other effective techniques. For the other variants including type-ahead search, approximate entity extraction and approximate substring matching, we also discuss recent studies to these problems..

VIII. PROJECT OUTCOME

A. Undertake problem identification, formulation and solution.
B. Be able to identify and summarize an appropriate list of literature review,analyse previous researchers' work and relate them to current project.
C. Design engineering solutions to complex problems utilization.
D. Demonstrate the knowledge, skills and attitudes of a professional engineer through the implemented product.
E. Be able to present the project outlining the approach and expected results using good oral presentation skills.
F. Able to work in team and communicate with peers.
G. Develop skills required in the current situation.
H. Show correct attitude towards achieving the goals and objectives.
I. Be able to produce project outcome of good quality.
J. Be able to compile, analyse and present the output of project in the form of report.

REFRENCES

[1] Verma P., Kesswani N. Web Usage mining framework for Data Clean ing and IP address Identification. CoRR, abs/1408.5460, 2014
[2] Maccio V.J., Chiang F., Down D.G. Models for Distributed, Large Scale Data Cleaning. PAKDD, 2014, 369–380
[3] Suresh Reddy G., Rajinikanth T.V., Rao A.A. Design and analysis of novel similarity measure for clustering and classification of high dimensional text documents. CompSysTech, 2014, 194–201
[4] Chaudhuri S.,Ganjam K., Ganti V., Motwani R. Robust and Efficient Fuzzy Match for Online Data Cleaning. SIGMOD, 2003, 313–324 23. Wang J., Li G., Feng J. Fast-join: An efficient method for fuzzy token matching based string similarity join. ICDE, 2011, 458– 469 24. Wang J., Li G., Feng J. Extending string similarity join to tolerant fuzzy token matching. ACM Trans. Database Syst., 39(1), 2014:7 25. Nandi A., Jagadish H.V. Effective Phrase Prediction. VLDB, 2007, 219–230
[5] Dai Z., Sun A., Liu X. Crest: Cluster-based Representation Enrichment for Short Text Classification. PAKDD, 2013, 256–267
[6] P. Ciaccia, M. Patella, and P. Zezula. M- tree: An efficient access method for similarity search in metric spaces. In PVLDB, VLDB '97, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
[7] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and J. F. Reid. A fast and practical bit-vector algorithm for the Longest Common Subsequence problem. Information Processing Letters, 80(6), Dec. 2001.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY