International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 2, April 2024, pp. 1948~1958
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i2.pp1948-1958

Efficient intelligent crawler for hamming distance based on prioritization of web documents

Amol Subhash Dange[1,5], Manjunath Swamy Byranahalli Eraiah[1], Manju More Eshwar Rao[2], Asha Kethaganahalli Hanumanthaiah[3], Sunil Kumar Ganganayaka[4]
[1]Department of Computer Science and Engineering, Don Bosco Institute of Technology, Bengaluru, India
[2]Department of Computer Science and Engineering, PES University, Bengaluru, India
[3]Department of Information Science and Engineering, Global Academy of Technology (Affiliate to Visvesvaraya Technological University Belagavi), Bengaluru, India
[4]Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bengaluru, India
[5]Department of Computer Science and Engineering, Visvesvaraya Technological University, Belagavi, India

Article Info

Article history:

Received Jul 29, 2023
Revised Sep 26, 2023
Accepted Nov 29, 2023

Keywords:

Hamming distance
Information retrieval
Intelligent crawling
Search engine
User preferences
Web crawling

ABSTRACT

Search engines play a crucial role in today's Internet landscape, especially with the exponential increase in data storage. Ranking models are used in search engines to locate relevant pages and rank them in decreasing order of relevance. They are an integral component of a search engine. The offline gathering of the document is crucial for providing the user with more accurate and pertinent findings. With the web's ongoing expansions, the number of documents that need to be crawled has grown enormously. It is crucial to wisely prioritize the documents that need to be crawled in each iteration for any academic or mid-level organization because the resources for continuous crawling are fixed. The advantages of prioritization are implemented by algorithms designed to operate with the existing crawling pipeline. To avoid becoming the bottleneck in pipeline, these algorithms must be fast and efficient. A highly efficient and intelligent web crawler has been developed, which employs the hamming distance method for prioritizing the pages to be downloaded in each iteration. This cutting-edge search engine is specifically designed to make the crawling process more streamlined and effective. When compared with other existing methods, the implemented hamming distance method achieves a high value of 99.8% accuracy.

Corresponding Author:

Amol Subhash Dange
Department of Computer Science and Engineering, Don Bosco Institute of Technology
Bengaluru, India
Email: amoldange_cse@adcet.in

1. INTRODUCTION

The technology that automatically collects desired data from websites is called web crawling, which analyses web browsing by using reinforcement learning and a large amount of data [1]. The gathering element for search engines is a web crawler, commonly referred to as a robot or spider. It is responsible for gathering information from the web. Crawling is the process of acquiring useful web pages with an interconnected structure link in a methodical and automated way [2]. The percentage of the entire number of online pages in the takeaway evaluation that have been crawled to the total number of web pages connected to its content is known as crawling completeness, also referred to as recall [3]. The uniform resource locator (URL) for these resources allows connections to other resources and serves as a form of resource identification. As a result, there is now a requirement for effective web systems that can track URLs and their contents for indexing [4]. Due to the development of data and the enormous number of documents that could be found to match a particular query string, indexing, and ranking algorithms are used to discover the best-matching documents [5]. Web data is becoming an increasingly valuable source of information due to the quality and quantity of data that is available for automated retrieval [6]. Huge amounts of objective data, such as the data that practitioners or researchers need, are collected and indexed by web crawlers. They construct massive fields of the gathered data by automatically collecting specified content from several websites. Web crawlers are becoming more crucial as big data is utilized in a wider range of industries and the amount of data available grows considerably every year [7]. Additionally, complete access to the functions of the public administrations and their associated data/documents via the web is offered to lower the cost of administrative tasks within the public sectors [8]. It is easy to spot ethical crawlers that follow the norms and guidelines of crawlers since they engage with web servers and consume online content at a controlled speed. However, there are still some unethical crawlers who attempt to deceive web servers and management to conceal their actual identities. Complex and compound procedures should be used to find unethical crawlers [9], [10]. A web crawler that automatically finds and retrieves every website page is used to gather web pages. A portion of the data collected from web pages can be used to enhance the crawling procedure. By examining the information on the pages, a focused crawler grabs pertinent web pages [11]. There is a lot of research on this problem that leverages data from online pages to enhance this task. Similar features and hierarchies can be found on several web pages of a website [12]. Three additional pieces of data analyzed from earlier web pages of a website are added to the information extraction method in this study to improve it [13]. The targeted crawling strategy uses machine learning (ML). The majority of these methods use classification algorithms to create new models from training data. The focused crawler and the current model are used to assess the relevance of the unvisited URLs. The effectiveness of the classifier-based focused crawler depends on the training data employed. Training data must be comprehensive enough to cover all aspects of the subject. With support vector machine SVM, specific text classification techniques are frequently utilized. As a consequence, these services enable cloud users to build custom crawlers to gather particular documents, topics, or information. Therefore, be offered as software as a service layer or more particularly as a crawler as a service.

The study expands on the web crawling methodology, which can automatically find and gather a large number of constructions from the internet [14]. The use of web crawler technology may lead to the development of novel environment survey techniques and approaches while obviating the need for significant amounts of labor, money, and time [15]. Kaur *et al.* [16] implemented an intelligent hidden web crawler (IHCW) for harvesting data in urban domains to handle the relevant issues, including prioritizing URLs, domain classification, and avoiding exhaustive searching. The crawler operates well while collecting data on pollutants. By using rejection rules, the crawler selects the pertinent websites and ignores the unimportant ones. The implemented framework was accurate and had a high harvest rate. It effectively collects hidden web interfaces from large-scale sites and outperforms other crawlers in terms of rates. However, IHWCs had trouble in effectively capturing and processing this real-time data because it necessitates continuous page refreshing. Hosseini *et al.* [17] implemented an SVM-based classifier for detecting crawlers, It categorizes numerous sorts of recognized text and non-text crawlers, as well as unknown and harmful ones to protect patient privacy and web server security. SVM was extremely effective at binary classification tasks, which makes them suited for accurately identifying trustworthy and harmful web crawlers, and the crawlers were detected with high precision. However, SVM-based classifier’s heterogeneous behavior and insufficient labeled data, compiling a representative sample of malicious web crawlers was difficult. Murugudu and Reddy [18] implemented a novel and efficient two-phase deep learning data crawler framework (NTPDCF), for intelligent crawling of search data to produce a large variety of effectively matched content. The two-phase data crawler architecture improves data harvesting and makes it possible to successfully extract pertinent information from a variety of web sources by effectively collecting targeted data from deep web interfaces. However, data crawlers must constantly adapt to the deep web interface’s dynamic nature to efficiently harvest data and avoid erroneous or incomplete extraction. Capuano *et al.* [19] implemented an ontology-driven approach to concentrate on crawling based on the use of both multimedia and textual material on online pages. The implemented framework was employed in a system to improve crawling activities by fusing the outcomes with novel technologies like linked open data and convolutional neural networks. A high degree of generalization in various conceptual domains was also made possible by the use of formal frameworks to express knowledge as ontologies. However, the crawler needs to operate with a lot of online pages, so this strategy necessitates manually labeling a sub-graph of the web, which would involve a lot of labeling work.

Hosseinkhani *et al.* [20] implemented the ANTON framework based on a semantic-focused crawler to support web crime mining using SVM. The enhanced criminal ontology employed in the implemented framework, which took elements from biological studies of ant foraging behavior, was developed using an ant-miner-focused crawler. The ANTON framework employs SVM for the classification of content that relates to crime, increasing precision and decreasing false positives to produce findings in crime mining.

However, due to pre-defined semantic information sources and frequent changes, the ANTON framework has trouble responding to changing crime patterns. Kaur and Geetha [21] implemented a SIMHAR crawler, based on hybrid technology, Sim+Hash, and hash-maps of Redis were utilized to detect duplication. The distributed web crawler for the hidden web detects the crawler accurately and submits the searchable forms. The Sim+Hash technique uses similarity-based crawling, which aims to fetch pages that are similar to those that have already been crawled. By minimizing duplicate crawling of similar pages, this method helps highlight important information. However, the surface web, which consists of publicly accessible web pages, is not optimized for crawling. This restriction limits its applicability to a particular portion of the internet. Sharma *et al.* [22] implemented an experimental performance analysis of web crawlers utilizing multi and single-threaded web crawling and indexing algorithms for the smart web contents application. The simulation work addresses the key parameters for the hierarchical, single, and multithreaded clustering strategy, including execution time and harvest ratio. The result of the implemented method provides the system's better performance than existing methods of self-adaptive, artificial neural networks (ANN), and probabilistic and it takes less time to investigate. However, the capacity of algorithms was not fully captured by experiments because the performance was varied with concurrent threads and web dataset sizes. Attia *et al.* [23] implemented a multi-criteria indexing and ranking model (MICR) based on weighted documents and pages which depend on one or more ranking factors. The MICR model was used in three distinct trials to compare the rankings of documents and pages based on one or more user preference criteria. The aim was to create a model that performs well, produces more relevant pages, and can index and rank articles that are both offline and online. However, the implemented MICR model was the model’s inability the search key phrases that were less/more important to understand and it consumes time. From the overall analysis, the existing method has some limitations in efficient intelligent crawler for prioritization of documents, that are mentioned above such as difficulty in compiling a representative collection of malicious web crawlers, the ANTON framework has trouble responding to changing crime patterns, and the model’s incapacity to distinguish between search key phrases that are less or more important. To overcome these issues, the paper implemented a hamming distance method for prioritizing the pages to be downloaded in each iteration. This cutting-edge search engine is specifically designed to make the crawling process more streamlined and effective. The major contributions of this method are given below: i) In this research the implemented hamming distance method is utilized for the efficient intelligent crawler based on prioritization of web documents; ii) This study’s main contribution is to enhance the time efficiency of the extraction process; iii) Both get and post methods can be used with this strategy. It finds and submits searchable web pages automatically in addition to detecting and locating them; and iv) The crawler can grow with the hidden web’s expanding size because it is scalable. It can be expanded to include more external elements like indexing.

This research paper is structured as follows: Section 2 describes the methodology of this research. Section 3 explains the hamming distance method. The result and discussion are described in section 4, and the conclusion of this research paper is given in section 5.

2. METHOD

Figure 1 shows the architecture of the proposed method. The given URLs must first be checked to see if we have already crawled the exact one. To do this, it will use a well-indexed database that contains the information on the URLs that have already been crawled. If not, it can move on to the next step, dump the data from the URL into the Kafka topic URL-data [24], and then continue to crawl the new URL that is produced in the Kafka topic URL topic, and extract the database structure form.

```
{_ id:'''',
        "URL":'''',
       "Website":''''}
```

The fact that a website may appear more than once will cause the URL and website to both be indexed together. Since each website’s URL will be unique when in search for a specific URL, the index will bring both the website and the URL, which will take less time than simply storing the URL value. When a crawler pulls information from a URL, the data can be separated into the website and URL for database storage. The second step involves taking the subject URL-data-topic, eliminating the script files, and styling components, and then extracting the heading from the bulleted points, H1, H2, H3, H3, H4, H5, and H6 tags, as well as any links to images or other links that may be there. In the topic similarity theme, the extracted topic is subsequently pasted. The grammar will be removed from the text in the third step information extracted from crawled data so that it is not optimized for search engines. Instead, the information will be compared to the data already present in the database, and if there is a similarity of more than 50%, it will be given a unique ID.

```
{"_ID":'''',
     "URL":'''',
Headings:[{
              "Headingname":'''',
          "Value":'''' }]
         Images:{"URL1","URL2"}
                     bulletedPoints:["bulleting points 1"]
                     similarity index:["_ID1","_ID2"]
}
```

As the search engine might not get an exact string to compare to, grouping all similar objects will be important as it will drastically cut down on the time required for the process of traveling through the entire database and extracting the *particular_id* when it appears. This process will be repeated for every message found on the topic. If the message is similar to another message, it will be stored and compared with that message.

The introduction of the scheduler, which will reintroduce the URLs to the URL topic before removing the details from the database so that it can repeat the initial phase, can be disabled for static websites that do not change frequently. To crawl just one static page using this will be helpful. The schedule time can be set dynamically for dynamic websites. Therefore, if the URLs are updated within 1 minute, the original time will be 1 minute. If it is updated, the time for a certain website will be increased by 2 minutes, just as it will if it is set to run again after 2 minutes. With this specific architecture, it can raise consumption when the message in the subject increases and decrease it when the message in the topic lowers. Both vertical and horizontal scalability will be aided by this event architecture. One consumer or producer's failures will have an impact on the entire system. The outage of either of them will not affect the system because there will be other customers present. After all, the process is asynchronous and requires less time to handle than a synchronous operation.

The two primary phases that form the indexing and ranking model's implemented architecture and each define a specific step in the ranking and indexing process for either offline web pages or documents [25]. The model can be utilized for storing documents offline or web pages online. Following is the description of the model's two primary phases,

Phase 1: The first phase of the proposed model starts with acquiring the users to enter the search query as input for the model. The methodology allows users to choose between one and display seven criteria for the results in the final ranking, giving them flexibility in determining their preferences and ordering priority based on their needs. To their needs and preferences, each user has the option of choosing any set of criteria, or all of them.

The model begins processing user queries after assessing the performances and needs of the user. Then it uses stemming and lemmatization to reduce inflectional forms and occasionally derivatively related forms of a word in the query to a common base form. The model then moves on to the next step, where it begins to crawl online and offline documents and pages, and then analyses the material by user requests. The model engine page crawls to determine what is on it, which is referred to as the process of indexing after learning a page's URL or document path, and the results are then indexed. The model begins with the keyword criterion of matching user queries in three places: page URL, domain name, and page content. The model engine also looks up the page's creation data in its metadata. These user preferences include keyword criterion (KYC) and creation data criterion (PD). Phase 2 of the model then delivers the ranking module of the data.

Phase 2: The page handler module initiates the second phase. The model begins to identify the types of pages or documents. In addition, the page's contents load in the builder page which is in charge of keeping track of every document or page's content so that the search query can be compared to it. The following is a description of the page handler module's pseudocode.

After using module 1 to load the pages and document information, the model begins the process of rank computation according to user criteria through the rank calculator. Phase one findings establish the preferences of the user before loading page content in the page handler. The rank calculator module, which forms the core of the MICR model, is where the content is first sent at this stage.

By processing the loaded material by the module of the page handler, the first stage discovers a pattern to determine the creation and modification dates of pages or documents. It also calculates the number of votes for each page by comparing the user's search query to the content of other pages to count the number of links on those other pages that are relevant to the search query. It also estimates the keyboard's number by counting the page terms, which is used to compare search queries used to the sites or documents' content. Finally, it delivers the number of total votes cast on the page in the current. The rank calculator module then shows additional criteria.

The weight module successfully determines the weight initial for every criterion according to the preferences of a user after Module 2 with that feature enabled. Finally, computes the ultimate score for every document and page by passing these values to the rank calculator. Additionally, the module of rank statistics, which is in charge of the final values displayed for every page, receives the obtained page scores.

3. HAMMING DISTANCE METHOD

In this research semantic data include textual information, such as labels, descriptions, or annotations associated with resources or predicates. hamming distance might be used as a measure of text similarity between two binary-encoded text strings, but this is a simplistic approach to textual comparison in the semantic web. In ontology matching or alignment tasks, where the goal is to align different ontologies or knowledge graphs, distance metrics or similarity measures are used to assess the similarity of concepts, properties, or entities in different ontologies. These metrics help identify mappings between ontology elements that are semantically similar. When there are several words, the hamming distance method [26] is used to determine which words are similar to one another. If more than one word is present, the word with the highest degree of similarity will be accepted. If the spring length does not match *hammingDist(str1, str)*, padding is used:

```
i=0
  count=0
 while(str1[i]!='\0'):
    if(str1[i]!=str2[i]):
      Count++
  [End of if condition]
    i++
  [End of the while loop]
  return count
```

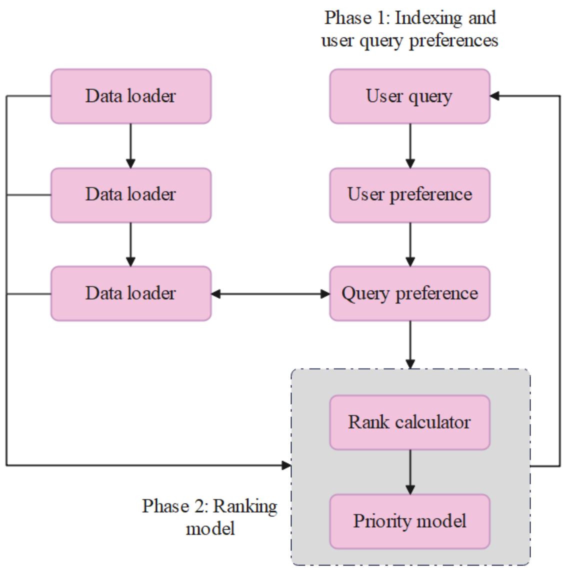

Figure 1. The architecture of the proposed method

The value of every heading and bullet point is determined. To ensure that each type has a priority value, as shown in Table 1, the similarity between H1 and H2 comparisons should not be greater than the comparison between H1 and H6. It will take less time for the data to display because the cache will be memory-sensitive rather than time-sensitive, as records will be deleted when the cache memory is full until that record is kept in the cache. When a specific text is searched and given, the 100% similarity index will be stored in the cache so that when the same text is searched again, the 100% similarity index is crawled data will be presented as output. When compared to searching through the complete database, the likelihood of finding a match for a given record is too great, thus the records with more similarity index that is cached will assist us in quickly identifying the record. When possible, records with a higher similarity index are discovered, and the old records will be replaced. During search engine optimization, as shown in Figure 2, a search engine for crawled data is introduced.

Some websites will need to give their content more attention, and this can be done by adding another topic prioritization to the architecture of the website in question. The results will be sorted by priority topic once we have crawled the site the most times, making it easier for power users to handle this request. The changed architecture will be represented in Figure 3.

Table 1. Priority types

Type	Priority
H1	1
H2	2
H3	3
H4	4
H5	5
H6	6
Bullet Points	7

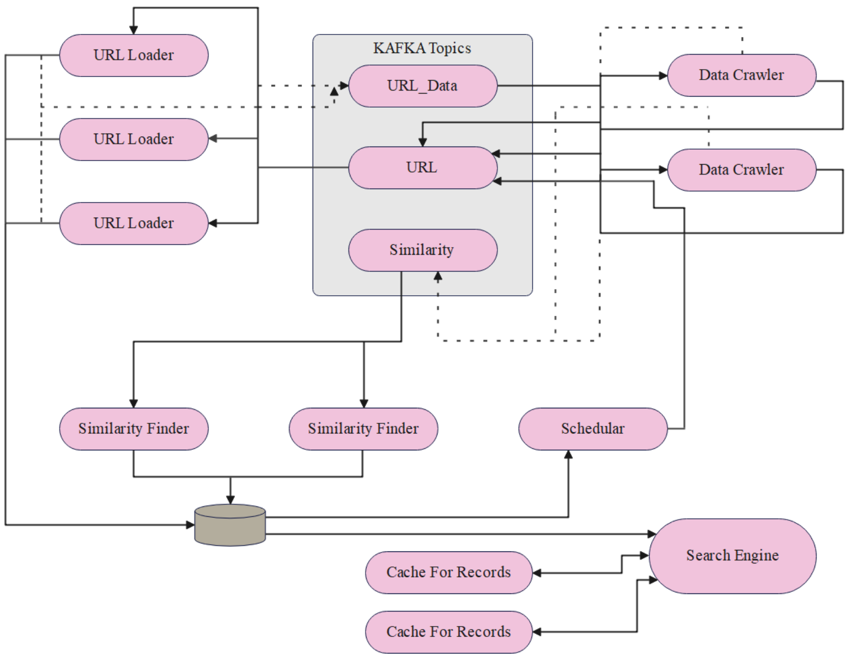

Figure 2. Introduction of search engine for crawled data

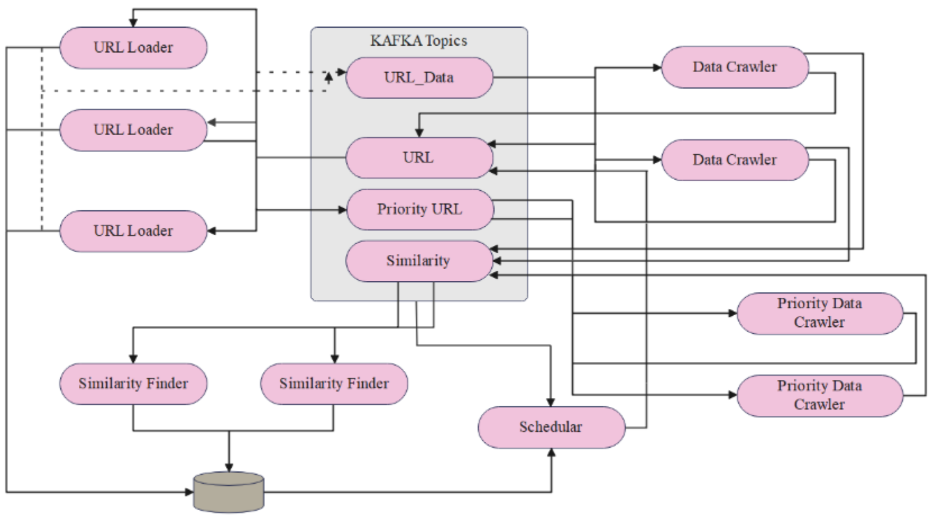

Figure 3. Web crawler architecture with induction of prioritization of URL

Based on a statistical learning theory approach, SVM pattern recognition [27] has important applications in computer pattern recognition. The research of SVM could not be ideal or address pattern recognition difficulties effectively. The new machine learning techniques, however, encounter some significant difficulties when trying to use statistical learning theory or neural networks. It may have improved the SVM classification algorithms [28] by determining the network structure difficulties using learning challenges caused by local minimal issues. The SVM quickly grows as a result of this, and it has been successfully applied to various classes of machine learning tasks involving text categorization. At last, the priority data crawlers will classify with the help of the SVM classifier.

4. RESULTS AND DISCUSSION

The experimental setup includes the Ubuntu 20.04 operating system and 32 GB of RAM. The technologies utilized to implement the concept of Docker, Docker-compose, Apache Kafka (Docker Image instead of actual software), and Python programming language. The implemented concept is powered by an Intel 17 octal-core processor. Wikipedia, Udemy, Medium, and Geeks for Geeks are among the initial URLs that are injected. Web crawlers will concentrate on headings and titles rather than the complete website's data. If a website adopts the implemented methodology, the primary material can always be made up of headers and titles, allowing for the provision of just restricted and essential content.

To identify genuine traveling, URLs with and without similarity were added. Table 2 lists the number of records that were explored both using and without a similarity index. Figure 4 shows the graphical illustration with and without a similarity index.

Table 2. Number of records against similarity vs without similarity index

Total No. of records	Without similarity index	Similarity index
250	250	150
500	500	350
1000	1000	800
1500	1500	1300

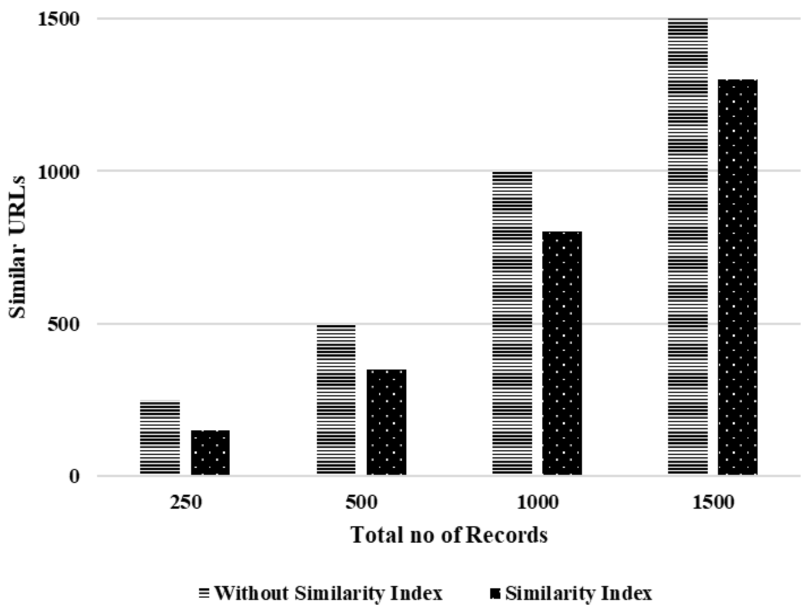

Figure 4. Graphical illustration with and without similarity index

Four consumer data crawlers, two similarity finders, and two URL loader generators were used in the experiment. Table 3 shows how much time the convolutional, event-driven, and topic-driven architecture each takes up. Additionally, Figure 5 displays a graphical study of the same. The graph makes it evident that the implemented event-driven architecture is more responsive and takes up less time as the number of records grows than the topic-driven design.

Table 3. Time consumed by architectures

No of URLs	Types of architectures in Time (s)		
	Regular Architecture	Topic Driven Architecture	Event-Driven Architecture
1	7	7	8
2	15	14	8
4	33	30	9
10	85	75	25

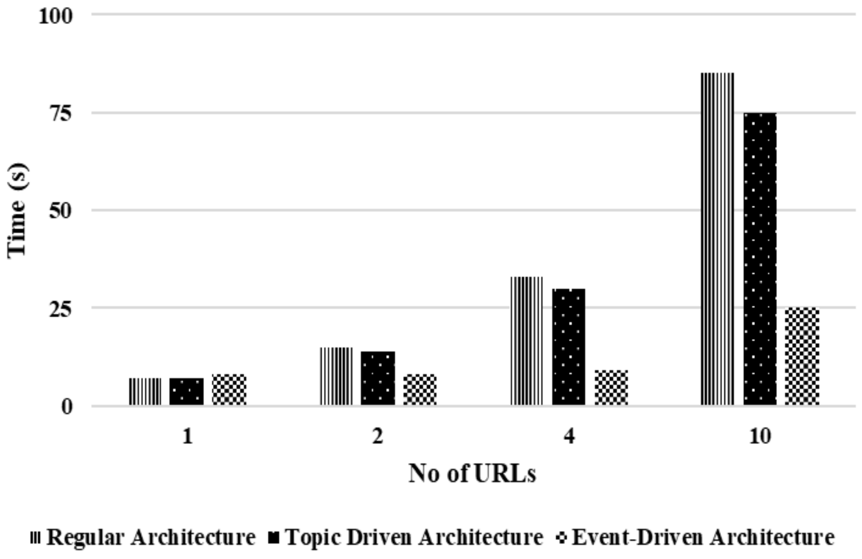

Figure 5. Analysis of time taken by regular vs. event-driven vs. topic-driven

4.1. Comparative analysis

In this section, there are some advantages and limitations for the efficient intelligent crawler for prioritization of documents as the implemented method was found accurate and had a high harvest rate, making them suited for accurately identifying trustworthy and harmful web crawlers and the crawlers were detected with high precision. However, the method had trouble effectively capturing and processing this real-time data because it necessitates continuous page refreshing, and the model's inability the search key phrases that were less/more important to understand and it consumes time. The implemented method compares with the existing methods of IHCM, SVM, NTPDCF, and SIMHAR concerning accuracy, precision, recall, and F-measure. Compared with other existing methods, the implemented hamming distance method achieves a high accuracy of 99.8%, a precision of 99.9%, a recall of 98%, and an F-measure of 99% are shown in Table 4.

Table 4. The existing and implemented models' comparative analysis

Author	Method	Accuracy (%)	Precision (%)	Recall (%)	F-measure (%)
Kaur *et al.* [16]	IHCW	90	84.62	81.06%	-
Hosseini *et al.* [17]	SVM	99.08	99.8	96	97.79
Murugudu and Reddy [18]	NTPDCF	95	91	72	-
Kaur and Geetha [21]	SIMHAR	99.4	99	97	98
Proposed method	Hamming Distance	99.8	99.9	98	99

4.2. Discussion

This section provides a discussion of the implemented hamming distance and compares those results with existing methods such as IHCM [16], SVM [17], NTPDCF [18], and SIMHAR [21] in the comparative analysis section. The main goal of the hamming distance is to avoid becoming the bottleneck in the pipeline, these algorithms must be fast and efficient. To allow for parallel processing, the entire task has been separated into smaller tasks. For each sub-task, the required work can be accomplished by a varied number of entities. It will be accomplished by utilizing Kafka in between the sub-tasks. This implemented method is used to determine which forms are not searchable and use rejection rules. This avoids laborious form crawling and also saves time and resources. First, the websites that might be interesting are found. The online pages that fall under the hidden web category are then found by applying rules of rejection. The establishment of guidelines for halting standards to prevent crawlers from stepping into spider traps. Both get and post methods can be used with this strategy. It finds and submits searchable web pages automatically in addition to detecting and locating them. The crawler can grow with the hidden web's expanding size because it is scalable. It can be expanded to include more external elements like indexing. The first subtask will be a procedure of Kafka, while the second will be a consumer of Kafka. The first subtask is distinct from the second, and the data in the next higher queue can now be processed. In comparison to the usual design, the implemented technique crawls, allowing for more equitable data distribution through plug-and-plug. When compared with the existing methods such as IHCM [16], SVM [17], NTPDCF [18], and SIMHAR [21], the implemented hamming distance method achieves better performance values of 99.8%, 99.9%, 98%, and 99% in terms of accuracy, precision, recall, and f-measure.

5. CONCLUSION

In this paper, the developed approach relies on scenarios where data processing is necessary, but complete resources are still unavailable. To enable parallel processing, the full task has been divided into different tasks. The needed work can be completed by a different number of entities for each sub-task. Utilizing Kafka in between the sub-tasks will be achieved. A producer of Kafka will be the first subtask, and a consumer of Kafka will be second. The first subtask is separate from the second subtask. The processing of the data in the next higher queue can begin. Compared to the standard architecture, the implemented method crawls. With the plug-and-play, the data can be distributed more equally. The resource will not be used until it is necessary. The implemented solution's pivotal aspect lies not in the code itself but rather in the robust infrastructure designed to handle diverse URLs and facilitate both vertical and horizontal scaling. Compared with other existing methods, the implemented hamming distance method achieves a high accuracy of 99.8%. A collection of services that can be offered in the future and that are based on online content also implements a more flexible web crawler for a particular use.

REFERENCES

[1] Y. Gao *et al.*, "Reinforcement learning based web crawler detection for diversity and dynamics," *Neurocomputing*, vol. 520, pp. 115–128, Feb. 2023, doi: 10.1016/j.neucom.2022.11.059.

[2] S. Neelakandan, A. Arun, R. R. Bhukya, B. M. Hardas, T. C. Anil Kumar, and M. Ashok, "An automated word embedding with parameter tuned model for web crawling," *Intelligent Automation and Soft Computing*, vol. 32, no. 3, pp. 1617–1632, 2022, doi: 10.32604/IASC.2022.022209.

[3] S. M. Cheema, S. Tariq, and I. M. Pires, "A natural language interface for automatic generation of data flow diagram using web extraction techniques," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 626–640, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.006.

[4] M. E. ElAraby and M. Y. Shams, "Face retrieval system based on elastic web crawler over cloud computing," *Multimedia Tools and Applications*, vol. 80, no. 8, pp. 11723–11738, Jan. 2021, doi: 10.1007/s11042-020-10271-3.

[5] P. S. Ang *et al.*, "Augmenting product defect surveillance through web crawling and machine learning in Singapore," *Drug Safety*, vol. 44, no. 9, pp. 939–948, Jun. 2021, doi: 10.1007/s40264-021-01084-w.

[6] J. Schedlbauer, G. Raptis, and B. Ludwig, "Medical informatics labor market analysis using web crawling, web scraping, and text mining," *International Journal of Medical Informatics*, vol. 150, Jun. 2021, doi: 10.1016/j.ijmedinf.2021.104453.

[7] N. Kumar, M. Gupta, D. Sharma, and I. Ofori, "Technical job recommendation system using APIs and web crawling," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–11, Jun. 2022, doi: 10.1155/2022/7797548.

[8] I. Bifulco, S. Cirillo, C. Esposito, R. Guadagni, and G. Polese, "An intelligent system for focused crawling from big data sources," *Expert Systems with Applications*, vol. 184, Art. no. 115560, Dec. 2021, doi: 10.1016/j.eswa.2021.115560.

[9] S. Eswaran, V. Rani, D. Daniel, J. Ramakrishnan, and S. Selvakumar, "An enhanced network intrusion detection system for malicious crawler detection and security event correlations in ubiquitous banking infrastructure," *International Journal of Pervasive Computing and Communications*, vol. 18, no. 1, pp. 59–78, Sep. 2022, doi: 10.1108/IJPCC-04-2021-0102.

[10] P. Verma, A. Goyal, and Y. Gigras, "Email phishing: text classification using natural language processing," *Computer Science and Information Technologies*, vol. 1, no. 1, pp. 1–12, May 2020, doi: 10.11591/csit.v1i1.p1-12.

[11] A. A. Ojugo and A. O. Eboka, "Memetic algorithm for short messaging service spam filter using text normalization and semantic approach," *International Journal of Informatics and Communication Technology (IJ-ICT)*, vol. 9, no. 1, pp. 9-18, Apr. 2020, doi: 10.11591/ijict.v9i1.pp9-18.

[12] Y. Lee and J. Cho, "Web document classification using topic modeling based document ranking," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2386–2392, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2386-2392.

[13] N. Campos Macias, W. Düggelin, Y. Ruf, and T. Hanne, "Building a technology recommender system using web crawling and natural language processing technology," *Algorithms*, vol. 15, no. 8, Art. no. 272, Aug. 2022, doi: 10.3390/a15080272.

[14] J. Hwang, J. Kim, S. Chi, and J. O. Seo, "Development of training image database using web crawling for vision-based site monitoring," *Automation in Construction*, vol. 135, Art. no. 104141, Mar. 2022, doi: 10.1016/j.autcon.2022.104141.

[15] J. Zhang, T. Zou, and Y. Lai, "Novel method for industrial sewage outfall detection: Water pollution monitoring based on web crawler and remote sensing interpretation techniques," *Journal of Cleaner Production*, vol. 312, p. 127640, Aug. 2021, doi: 10.1016/j.jclepro.2021.127640.

[16] S. Kaur, A. Singh, G. Geetha, and X. Cheng, "IHWC: intelligent hidden web crawler for harvesting data in urban domains," *Complex and Intelligent Systems*, vol. 9, no. 4, pp. 3635–3653, Jul. 2023, doi: 10.1007/s40747-021-00471-1.

[17] N. Hosseini, F. Fakhar, B. Kiani, and S. Eslami, “Enhancing the security of patients’ portals and websites by detecting malicious web crawlers using machine learning techniques,” *International Journal of Medical Informatics*, vol. 132, Art. no. 103976, Dec. 2019, doi: 10.1016/j.ijmedinf.2019.103976.
[18] M. R. Murugudu and L. S. S. Reddy, “Efficiently harvesting deep web interfaces based on adaptive learning using two-phase data crawler framework,” *Soft Computing*, vol. 27, no. 1, pp. 505–515, May 2023, doi: 10.1007/s00500-021-05816-z.
[19] A. Capuano, A. M. Rinaldi, and C. Russo, “An ontology-driven multimedia focused crawler based on linked open data and deep learning techniques,” *Multimedia Tools and Applications*, vol. 79, no. 11–12, pp. 7577–7598, Dec. 2020, doi: 10.1007/s11042-019-08252-2.
[20] J. Hosseinkhani, H. Taherdoost, and S. Keikhaee, “ANTON Framework based on semantic focused crawler to support web crime mining using SVM,” *Annals of Data Science*, vol. 8, no. 2, pp. 227–240, Apr. 2021, doi: 10.1007/s40745-019-00208-5.
[21] S. Kaur and G. Geetha, “SIMHAR - smart distributed web crawler for the hidden web using Sim+Hash and Redis server,” *IEEE Access*, vol. 8, pp. 117582–117592, 2020, doi: 10.1109/ACCESS.2020.3004756.
[22] A. K. Sharma, V. Shrivastava, and H. Singh, “Experimental performance analysis of web crawlers using single and Multi-Threaded web crawling and indexing algorithm for the application of smart web contents,” *Materials Today: Proceedings*, vol. 37, no. Part 2, pp. 1403–1408, 2020, doi: 10.1016/j.matpr.2020.06.596.
[23] M. Attia, M. A. Abdel-Fattah, and A. E. Khedr, “A proposed multi criteria indexing and ranking model for documents and web pages on large scale data,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 8702–8715, Nov. 2022, doi: 10.1016/j.jksuci.2021.10.009.
[24] G. Calderon, G. del Campo, E. Saavedra, and A. Santamaría, “Monitoring framework for the performance evaluation of an iot platform with Elasticsearch and Apache Kafka,” *Information Systems Frontiers*, Jul. 2023, doi: 10.1007/s10796-023-10409-2.
[25] S. An and J. J. Jung, “A heuristic approach on metadata recommendation for search engine optimization,” *Concurrency and Computation: Practice and Experience*, vol. 33, no. 3, Jun. 2021, doi: 10.1002/cpe.5407.
[26] N. Sharkasi and S. Rezakhah, “A modified CRITIC with a reference point based on fuzzy logic and hamming distance,” *Knowledge-Based Systems*, vol. 255, p. 109768, Nov. 2022, doi: 10.1016/j.knosys.2022.109768.
[27] Z. Yin, J. Zheng, L. Huang, Y. Gao, H. Peng, and L. Yin, “Sa-SVM-based locomotion pattern recognition for exoskeleton robot,” *Applied Sciences (Switzerland)*, vol. 11, no. 12, p. 5573, Jun. 2021, doi: 10.3390/app11125573.
[28] D. Wahyudi, M. Niswar, and A. A. P. Alimuddin, “Website phising detection application using support vector machine (SVM),” *Journal of Information Technology and Its Utilization*, vol. 5, no. 1, pp. 18–24, Jun. 2022, doi: 10.56873/jitu.5.1.4836.

BIOGRAPHIES OF AUTHORS

Amol Subhash Dange has graduated in computer science and engineering from Shivaji University, Kolhapur, Maharshtra. He has obtained his Masters of Technology in computer engineering from Bharti Vidyapeeth Deemed University, College of Engineering, Pune and Pursuing Doctoral Degree in computer science and engineering from Visvesvaraya Technological Univrsity, Belgavi, He is research scholar of Don Bosco Institute of Technology, Banglore. He has 14 years of teaching experience. Currently he is working in Department of Computer Science and Engineering, of Annasaheb Dange College of Engineering and Technology, Ashta, Sangli, Maharshtra. He has over 25 research papers to his credit. He received fund from different schemes of University and AICTE. He has filed 1 US Patents. His area of interest includes software engineering, information retrieval and data science. He can be contacted at email: amoldange_cse@adcet.in/amol.dange@gmail.com.

Manjunath Swamy Byranahalli Eraiah has graduated from Visvesavaraya Technological University, Belgaum, He obtained his Masters of Engineering and Doctoral Degree from Bangalore University, Bengaluru. Currently he produced 2 Ph.D students from VTU, Belagavi. He has 15 years of teaching experience. Currently he is working in Department of Computer Science and Engineering, Don Bosco Institute of Technology. Bangalore. He has over 35 research papers to his credit. He received fund from AICTE, VGST and KSCST. He has filed 4 Indian patents and one Australian Patent. His area of interest includes image processing, signal processing, and network security, cloud computing, IoT, data science. He can be contacted at email: manjube2412@gmail.com.

Manju More Eshwar Rao currently working in Gitam Deemed to be University, Bangalore in the Department of Computer Science and Engineering. She has around 11 years of teaching Experience from Alpha College of Engineering and REVA University Bangalore. She has obtained her bachelor of engineering (B.E), master of technology (M.Tech) and doctor of philosophy (Ph.D) from Visvesvaraya Technological University(VTU), Belagavi. She has published various technical and research papers in national, international journals and conferences. Her areas of research include cloud computing, data mining, and machine learning. She can be contacted at email: manjumore13@gmail.com.

Asha Kethaganahalli Hanumanthaiah currently working in Don Bosco Institute of Technology in the Department of Information Science and Engineering. She has around 11 years of teaching experience from Alpha College of Engineering. She has obtained her bachelor of engineering (B.E), master of technology (M.Tech) and doctor of philosophy (Ph.D) from Visvesvaraya Technological University (VTU), Belagavi. She has published various technical and research papers in National, International journals and conferences. She can be contacted at email: asha.kh06@gmail.com.

Sunil Kumar Ganganayaka has completed bachelor of engineering from Visvesavaraya Technological University, Belgaum, masters of engineering and doctoral degree in computer science and engineering from University Visvesvaraya College of Engineering, Bangalore University, Bengaluru. He has over 15 years of teaching and research experience. Currently he is an assistant professor in the Department of CSE, University Visvesvaraya College of Engineering, Bengaluru. He has published over 35 research papers in refereed International Journals and Conferences. Currently he is guiding 4 Ph.D. students and supervised more than 30 post graduate dissertations in computer science and engineering. He has organized 10 International Conferences and more than 20 Workshops and FDP's. He has awarded 2 best paper awards and has filed 2 Patents. His research interests include cognitive networks, computer networks, wireless sensor networks, parallel and distributed systems, cloud computing and natural language processing. He can be contacted at email: sunil777g@gmail.com.