Ingénierie des Systèmes d'Information
Vol. 30, No. 5, May, 2025, pp. 1147-1161
Journal homepage: http://iieta.org/journals/isi

Hybrid Contextual Ontology-Fuzzy Logic and LSTM Model for Efficient Web Crawler Prediction and Traffic Optimization

Suresh Ponnur Mani*, Raja Kothandaraman

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai 600089, India

Corresponding Author Email: sureshpm.networks@gmail.com

https://doi.org/10.18280/isi.300504

Received: 29 January 2025
Revised: 19 April 2025
Accepted: 29 April 2025
Available online: 31 May 2025

Keywords:
contextual ontology, fuzzy logic, LSTM, web crawler prediction, traffic optimization, parallel web crawlers, intelligent information retrieval, network traffic management, temporal dependencies

ABSTRACT

Efficient management of web crawlers and prediction of their behavior remain key challenges in the domain of intelligent information retrieval, particularly when multiple crawlers operate simultaneously in dynamic environments. This research presents a hybrid framework that integrates long short-term memory (LSTM) networks, fuzzy logic, and contextual ontology to enhance the accuracy and efficiency of web crawling and traffic optimization. The LSTM component is responsible for identifying temporal patterns in historical crawling behavior and predicting future crawler actions, while fuzzy logic deals with uncertain or imprecise web data, enabling smoother decision-making processes. To enrich the semantic understanding of web content and improve context-based data extraction, a contextual ontology is employed, allowing for intelligent interpretation and classification of retrieved web data. The proposed model was evaluated using two large-scale benchmark datasets: Common Crawl (250 GB) and ClueWeb09 (1 TB). These datasets were chosen for their diversity and representation of real-world web structures and content. Experimental results demonstrate that the proposed system outperforms conventional approaches, achieving an 18.3% improvement in prediction accuracy and a 15.7% reduction in network traffic, compared to baseline LSTM and rule-based models. These results confirm the model's capability to reduce redundant data retrieval, avoid crawler overlaps, and enhance resource efficiency. This study highlights the potential of combining machine learning with semantic technologies to improve web crawling in complex environments. The proposed hybrid system offers a scalable, intelligent solution for high-performance information retrieval and efficient network traffic management.

1. INTRODUCTION

The broad appeal of the World Wide Web (WWW) has presented challenges not only for society but also for the technological innovations that created and maintain it. The Web is expanding in society faster than we can comprehend its implications or establish guidelines for its use [1]. The widespread usage of the Internet has raised significant societal issues related to information availability, restrictions, and security. From a technical perspective, the Web's uniqueness and rapid growth have posed challenges in developing new applications that fully utilize its potential, as well as in building the infrastructure required to scale such applications to handle massive data loads and information sets [2].

Fields like data retrieval, computation, and decentralized networks have both benefited from and contributed to addressing these new challenges. These data repositories are now known as search engines, which index the entire WWW and present results to users based on their search queries [3]. Search engines employ web crawlers, also known as robots, to crawl the web and retrieve information from the WWW. These sources of knowledge include legal, social, educational, and food-related data [4].

A web crawler is an automated computer program that systematically searches the WWW. Web crawlers are also referred to by various names, including web robots, web spiders, web wanderers, bots, automated indexers, and spiders. Spidering is a common technique used by many websites, especially search engines, to gather current information [5]. Web crawlers are primarily employed to store copies of each web page they visit so that an internet search engine can later analyze and index the retrieved pages. Crawlers can also automate website maintenance tasks such as validating HTML code and checking links. They can be used to extract specific types of data, such as email addresses [6].

Generally, a web crawler starts with a list of URLs to visit, referred to as seed URLs. When the crawler visits these URLs, it identifies all the hyperlinks on the page and adds them to the crawl frontier—a list of URLs to visit in the future [7]. The crawler then continues to visit URLs from the frontier according to a set of policies. The effectiveness of the WWW can be attributed to two key factors: its vastness and the lack of centralized content management. These same factors also contribute to the challenges of finding information on the Web [8]. The Web presents a challenge to traditional Information Retrieval (IR) techniques because the sheer volume and rapid evolution of the Web exceed the capacity of modern search engines [9]. The quality of content on the Web is uneven, with

a small number of valuable web pages buried under a massive amount of less relevant content.

For the Internet to function correctly, certain protocols must be followed. A protocol is a well-defined format for connecting, communicating, and exchanging data between two machines on a network [10]. The Internet employs several important protocols, including Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS), and File Transfer Protocol (FTP). HTTPS is used by websites that request sensitive information such as passwords and other private data [11]. HTTP is used to transfer web pages over the WWW. Websites using HTTPS have a Secure Sockets Layer (SSL) certificate issued by a trusted third party, which can be examined when a webpage is accessed using HTTPS. FTP is another common protocol used for transferring files across the Internet [12]. Web developers use FTP to upload new versions of their websites, while HTTP is used to display files in a web browser. FTP allows files to be transferred from one machine to another, such as from a home computer to a remote web server [13]. Every search engine maintains a core database of websites. The search engine retrieves results from this database based on the user's query. A web crawler, also known as a spider or bot, searches the Internet for web content. The web crawler begins by downloading online documents from an initial set of URLs [14].

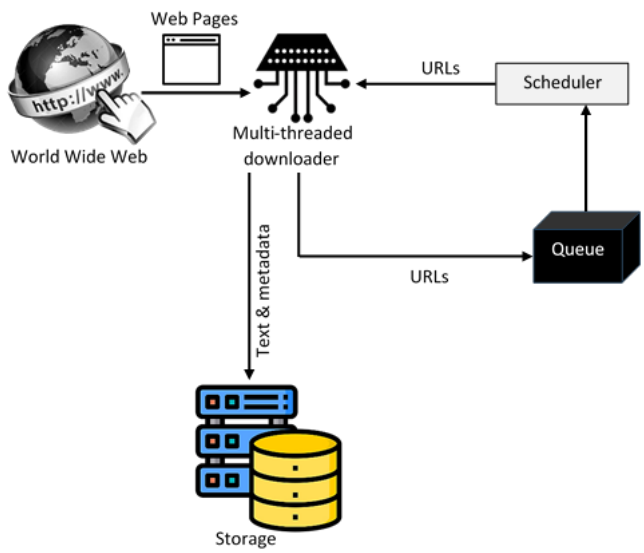


Figure 1. Web crawler architecture

The search engine's information is kept up to date by the web crawler architecture integrates updates from the internet. To overcome computational congestion, multiple instances of this component operate in parallel and communicate at high frequencies. Using the HTTP protocol, web crawlers retrieve and manipulate web pages from the internet [15]. The web crawler architecture consists of two main components: the processor and the downloader. The downloader retrieves websites from the World Wide Web and sends them to the processor for further analysis. The HTTP protocol is used to download these web pages. Various efficiency techniques such as TCP reuse, pipelining, conditional GETs, variable refresh rates, and DNS caching are employed to overcome network constraints [16]. Simultaneous web crawlers operate concurrently to optimize download speed and cover a large portion of the internet, as shown in Figure 1. Concurrent crawlers face challenges such as webpage overlap, bandwidth consumption, and reliability issues [17]. Since crawlers might not be aware of each other's previous searches, the same web pages may be retrieved multiple times, resulting in overlap. Minimizing this overlap conserves network bandwidth, allows for more unique downloads, and improves the overall efficiency of web crawlers [18].

The goal of a web crawler is to download as many significant web pages as possible. When multiple crawlers operate simultaneously, they may lack knowledge of the complete set of pages retrieved collectively. Consequently, crawling decisions based solely on local data can lead to poor performance [19]. To reduce redundancy and maintain the quality of retrieved content, crawlers must communicate with each other. However, this communication consumes both bandwidth and time.

Crawlers assign priority to web pages based on their relevance score, which is calculated using techniques such as Term Frequency–Inverse Document Frequency (TF-IDF) and cosine similarity [20].

2. RELATED WORKS

One of the core concerns in modern information retrieval systems is the accuracy and relevance of responses to user queries. Query expansion has emerged as a widely adopted strategy to improve retrieval effectiveness [21]. In particular, keyword-based query expansion enhances user satisfaction and recall by semantically enriching the initial user query before it is submitted to a search engine. This enrichment is achieved by incorporating synonyms or related terms, thereby capturing a broader scope of relevant documents [22]. Two main types of query expansion strategies are commonly employed: local and global analysis. In local analysis, expansion terms are extracted from the top-ranked documents retrieved in response to the initial query, effectively refining the query based on its own results. In contrast, global analysis adds expansion terms based on a broader corpus or thesauri, independent of the initial result set. The vector space model is typically used to support such retrieval processes, enabling partial matching rather than requiring exact keyword matches, thus enhancing the search's flexibility [23].

Ontologies play a vital role in improving semantic understanding in information retrieval. An ontology provides a structured representation of domain knowledge, capturing entities, relationships, and shared concepts. It enables the semantic enrichment of queries and helps manage shared access and communication in distributed environments [24]. For instance, OntoCrawler integrates an ontology-based academic database to effectively crawl scholarly webpages by leveraging structured metadata and associated conceptual hierarchies [25]. Hypertext systems, foundational to the web's architecture consist of interconnected documents linked by embedded hyperlinks. The concept of hypertext allows users to navigate through related content seamlessly, enhancing the exploration of large information spaces. This user-centric model inspired the design of the World Wide Web, where millions of interconnected documents allow for non-linear browsing and dynamic information discovery [26]. The vast and unstructured nature of the Web poses significant challenges for information retrieval. Parallel and distributed web crawlers have emerged to address the scalability of data collection across diverse web domains [27].

Web search and classical information retrieval differ

notably in terms of scale and user expectations. Web search engines operate on extremely large-scale, heterogeneous data with a wide user base exhibiting diverse search behaviors. Information Retrieval (IR) systems aim to return a relevant subset of documents from a much smaller, structured repository [28]. Key components of IR systems include document indexing, query processing, ranking algorithms, and user interfaces for relevance feedback [29]. Precision and recall are commonly used to assess the effectiveness of these systems. Precision measures the proportion of relevant documents among retrieved ones, while recall evaluates the proportion of all relevant documents successfully retrieved [30].

3. MATERIALS AND METHODS

Focuses on enhancing the efficiency and accuracy of web crawling in dynamic and complex web environments. Existing web crawlers often struggle with predicting changes in web content and managing network traffic, which can lead to inefficiencies in data retrieval and indexing. This model combines the strengths of contextual ontology, fuzzy logic and LSTM networks to address these challenges shown in Figure 2. The contextual ontology component is designed to capture and organize the relevant context from web pages, improving the system's ability to retrieve meaningful and precise information. Fuzzy logic is integrated to manage uncertainty and imprecision in the web data, enabling more adaptive and efficient decision-making in web crawling processes. LSTM networks are utilized to predict web crawler behavior by learning from past interactions allowing the system to anticipate changes in web content and optimize its crawling strategies. This hybrid proposed system aims to reduce unnecessary network traffic, enhance prediction accuracy, and improve the overall performance of web crawlers. By addressing key issues such as traffic optimization and crawler prediction, this model offers a more intelligent and scalable solution for web indexing and information retrieval in the fast-evolving digital landscape.

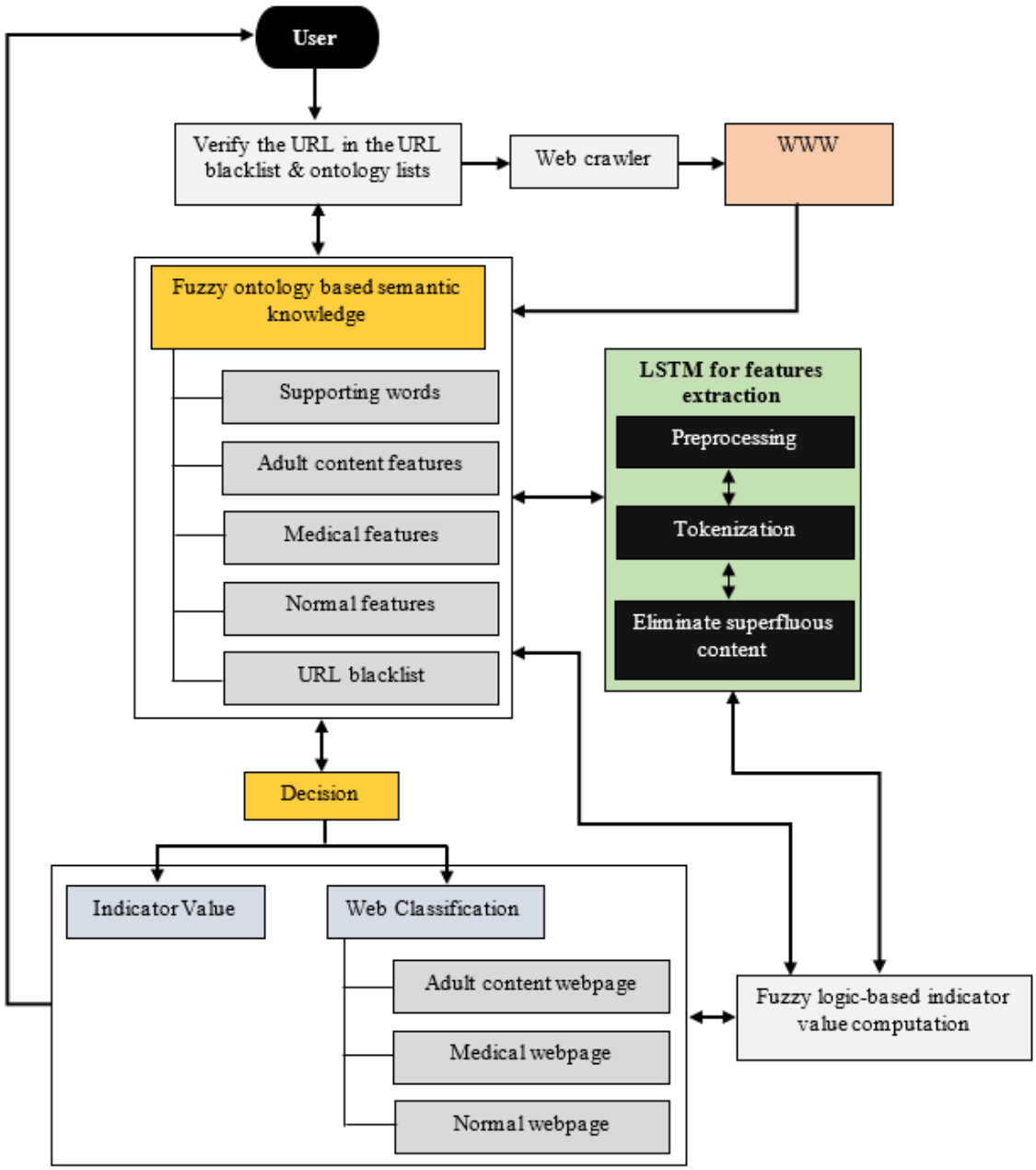


Figure 2. Proposed architecture

3.1 Dataset description

Large-scale web page material such as HTML, text, and metadata included in the Web Crawl Data 1 collection was obtained through Common Crawl. The algorithm's capacity to efficiently gather and evaluate material from a variety of web pages is trained using this dataset. The history of web page modifications may be found in the Web Page Change Logs dataset, which was retrieved from Archive.org. The information in the dataset is utilized for training the LSTM network allowing the algorithm to forecast upcoming shifts in online content according to historical trends. Variables such as URLs, time stamps, and modification frequency are included in the dataset. Comprehensive information on network traffic such as bandwidth utilization, number of requests and frequency latency available in the Network Traffic Logs dataset was gathered using custom crawlers. Using fuzzy reasoning to maximize network traffic and cut down on wasteful bandwidth usage requires this set of data.

Contextual connections and semantic linkages among the organized information provided by the Ontology-Based Dataset found in resources such as DBpedia and Wikipedia. Through a contextual and ontological framework, this collection of data offers a comprehensive contextual foundation that enhances web indexing. Finally, contextual information such as phrases and context labels are included in the Web Page Context Information collected by specialized crawlers. This collection of data uses context-based insight to better analyze and index online material helps to improve information retrieval shown in Table 1. These datasets aid in the learning, optimization, and validation of many facets of the approach, guaranteeing thorough development and efficient operation in the areas of handling traffic and web crawler predictions.

The information that is contained throughout each dataset is exemplified by these examples include web page pleased, change history, structured knowledge, network traffic statistics derived from ontologies, and contextual data taken from websites shown in Tables 2-6.

3.2 Pre-processing

For the development of a contextual ontology-based fuzzy logic and LSTM model for web crawler prediction, data preprocessing involves several steps to clean and prepare web data.

Table 1. Dataset description

Dataset Name	Source	Data type	Size	Features	Purpose
Web Crawl Data 1	Common Crawl	Web page content (HTML, text)	1TB	URL, HTML tags. metadata, page content	Training the model for content extraction
Web Page Change Logs	Archive.org	Change history of web pages	500GB	URL, timestamps. change frequency	LSTM training for predicting crawler behavior
Network Traffic Logs Custom	Crawler Data Collection	Network traffic statistics	300GB	Bandwidth usage, number of requests, latency	Optimizing network traffic using fuzzy logic
Ontology- Based Dataset	DBpedia, Wikipedia	Structured knowledge	200GB	Contextual relationships. semantic links	Enhancing web indexing with contextual ontology
Web Page Context Data	Custom Crawler Data Collection	Contextual data	150GB	Keywords, context tags, extracted entities	Context-based information retrieval

Table 2. Web crawl data 1

URL	HTML Content	Metadata
http://ex.com/page1	<html><head><title>Ex Page 1</title></head> <body> This is an ex page. </body></html>	<meta name="description" content-"Ex Page 1">

Table 3. Web page change logs

URL	Timestamp	Type	Details
http://ex.com/pagel	2024-09-16 11:00:00	Content Update	Changed paragraph text from "This is..."to "Updated content..."

Table 4. Network traffic logs

Timestamp	URL	Bandwidth Usage (MB)	Number of Requests	Latency (ms)
2024-09-16 11:00:00	http://ex.com/pagel	5.5	120	152

Table 5. Ontology-based dataset

Entity	Contextual Relationship	Semantic Link
Ex Page 1	Has keyword "ex"	Related to "tutorial"
Ex Page 2	Contains "guide"	Part of "documentation"

Table 6. Web page context data

URL	Keyword	Context Tag	Extracted Entities
http://ex.com/pagel	example, tutorial	educational resource	"ex page", "tutorial guide"

3.2.1 HTML tag removal

Extract raw text from HTML content to focus on the actual textual information described using a regular expression (regex):

Regex Pattern: <[^>]+>

This pattern matches any sequence of characters that starts < with and ends with >, effectively identifying and removing HTML tags.

Example: Original HTML: <html><head><title>Ex</title></head><body> Sample text. </body></html>

Cleaned Text: Sample text.

HTML Tag Removal: Utilize regex <[^>]+> to strip HTML tags from content.

3.2.2 Regular expression pattern matching

Extract specific patterns or clean up the text further based on defined patterns.

The regex pattern itself is used for text processing.

Example Patterns: Email Extraction: [\w\.-]+@[\w\.-]+

Matches email addresses. URL Extraction: http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+

Matches URLS.

Example:

Text: Contact us at example@ex.com or visit http://ex.com

Extracted Information: ex@ex.com, http://ex.com

Regular Expression Pattern Matching: Apply regex patterns to extract or clean specific information (e.g., emails, URLs).

3.2.3 Stop word removal

Remove common words (stop words) that do not contribute significant meaning to the text, thereby reducing dimensionality and noise.

Stop Words List Example: ['a', 'an', 'the', 'is', 'in', 'and']

Process: Tokenize the text into words. Remove words that are in the stop words list.

Example:

Original Text: The quick brown fox jumps over the lazy dog.

Tokenized: [The', 'quick', 'brown', 'fox', 'Jumps', 'over', 'the', 'lazy', 'dog']

After Stop Word Removal: ['quick', 'brown', 'fox', 'jumps', 'lazy', 'dog]

Stop Word Removal: Filter out common words from the tokenized text using a predefined list of stop words.

These preprocessing steps ensure that the data fed into the contextual ontology-based fuzzy logic and LSTM model is clean, relevant, and ready for further analysis and modeling.

3.3 Feature extraction

The goal is to maximize the probability of observed word-context pairs while minimizing the probability of non-observed pairs. This can be mathematically represented in the objective function:

$$Objective\ Function = \sum_{(w,c)\epsilon D} logp\big(Z=1\big|(w,c)\big) + \sum_{(w,c)\epsilon D,} logp\big(Z=0\big|(w,c)\big) \quad (1)$$

To model these probabilities, we often use a sigmoid function for $p\big(Z=1\big|(w,c)\big)$:

$$p\big(Z=1\big|(w,c)\big) = \sigma(v_w^T v_c) \quad (2)$$

where, σ denotes the sigmoid function:

$$\sigma(i) = \frac{1}{1+e^{-i}} \quad (3)$$

Similarly, $p\big(Z=1\big|(w,c)\big)$ can be computed as:

$$p\big(Z=1\big|(w,c)\big) = 1 - \sigma(v_w^T v_c) \quad (4)$$

The optimization process involves adjusting the vector representations v_w and v_c to maximize the objective function, thereby learning meaningful word embeddings based on their contexts. Figure 3 explains the preprocessing and feature extraction in step by step procedure. For the Development of a contextual ontology-based fuzzy logic and LSTM Model for web crawler prediction, feature extraction using LSTM networks involves transforming raw data into meaningful features that capture temporal dependencies and contextual information.

Step 1: Data Preparation: Transform raw web data into a sequence suitable for LSTM input.

Steps:

Tokenization: Convert raw text into sequences of tokens (words or subwords).

Embedding: Use word embeddings (e.g., Word2vec, Glove) to represent tokens as dense vectors.

$$E(w) = v_w \quad (5)$$

where, E(w) is the embedding of word w. v_w is the pre-trained vector representation of w.

Step 2: LSTM Feature Extraction: LSTM networks are designed to capture temporal dependencies in sequential data. The LSTM unit includes several components to process the sequence data:

$$\text{Forget Gate: } f_t = \sigma(W_f.[h_{t-1}, i_t] + b_f) \quad (6)$$

$$\text{Input Gate: } x_t = \sigma(W_x.[h_{t-1}, i_t] + b_x) \quad (7)$$

$$\bar{C}_t = tanh(W_C.[h_{t-1}, i_t] + b_C) \quad (8)$$

$$\text{Cell State Update: } C_t = f_t.C_{t-1} + x_t.\bar{C}_t \quad (9)$$

$$\text{Output Gate: } o_t = \sigma(W_o.[h_{t-1}, i_t] + b_o) \quad (10)$$

$$h_t = o_t.\tanh(C_t) \quad (11)$$

where, σ is the sigmoid activation function; tanh is the hyperbolic tangent activation function; W_f, W_C, W_x, W_o are weight matrices for the forget, input, cell, and output gates, respectively; b_f, b_C, b_x, b_o are bias terms; i_t is the input vector at time t; h_{t-1} is the hidden state from the previous time step; C_t is the cell state at time f.

Step 3: Contextual Features Extraction: Integrate contextual information extracted from a contextual ontology into the LSTM features.

Contextual Embedding: Contextual embeddings represent words in the context of their surrounding words, derived from a contextual ontology.

$$E_c(w) = v_w + c_w \quad (12)$$

where, $E_c(w)$ is the contextual embedding of word w; c_w is the vector representing contextual information from the ontology.

Feature Vector for LSTM: Combine contextual embeddings with LSTM outputs to form the final feature vector.

$$F_t = Concat(h_t, E_c(w_t)) \quad (13)$$

where, F_t is the feature vector at time t. Concat denotes concatenation of LSTM hidden state h_t and contextual embedding $E_c(w_t)$.

Step 4: Feature Aggregation: Aggregate features from multiple time steps to represent the entire sequence.

Pooling: Apply pooling techniques (e.g., max pooling, average pooling) to aggregate features over time.

$$F_{agg} = Pooling(\{F_t\}_{t=1}^{T}) \quad (14)$$

where, F_{agg} is the aggregated feature vector. $\{F_t\}_{t=1}^{T}$ represents the feature vectors from all time steps.

Final Feature Representation: The aggregated features can be used for further processing, such as classification or prediction.

Feature extraction involves preparing data through tokenization and embedding, processing sequences with LSTM to capture temporal dependencies, integrating contextual information from ontology-based embeddings, and aggregating features for final representation. The resulting features are then utilized for web crawler prediction and other analyses shown in Figure 4.

Input Data: This is the input data for the model can be text, images, or other sequential data.

Pre-processing: This step involves cleaning and preparing the input data. For text data, this might include tasks like tokenization, stemming, and stop word removal.

Word Embedding: This layer converts words or tokens into numerical representations, capturing semantic relationships between words.

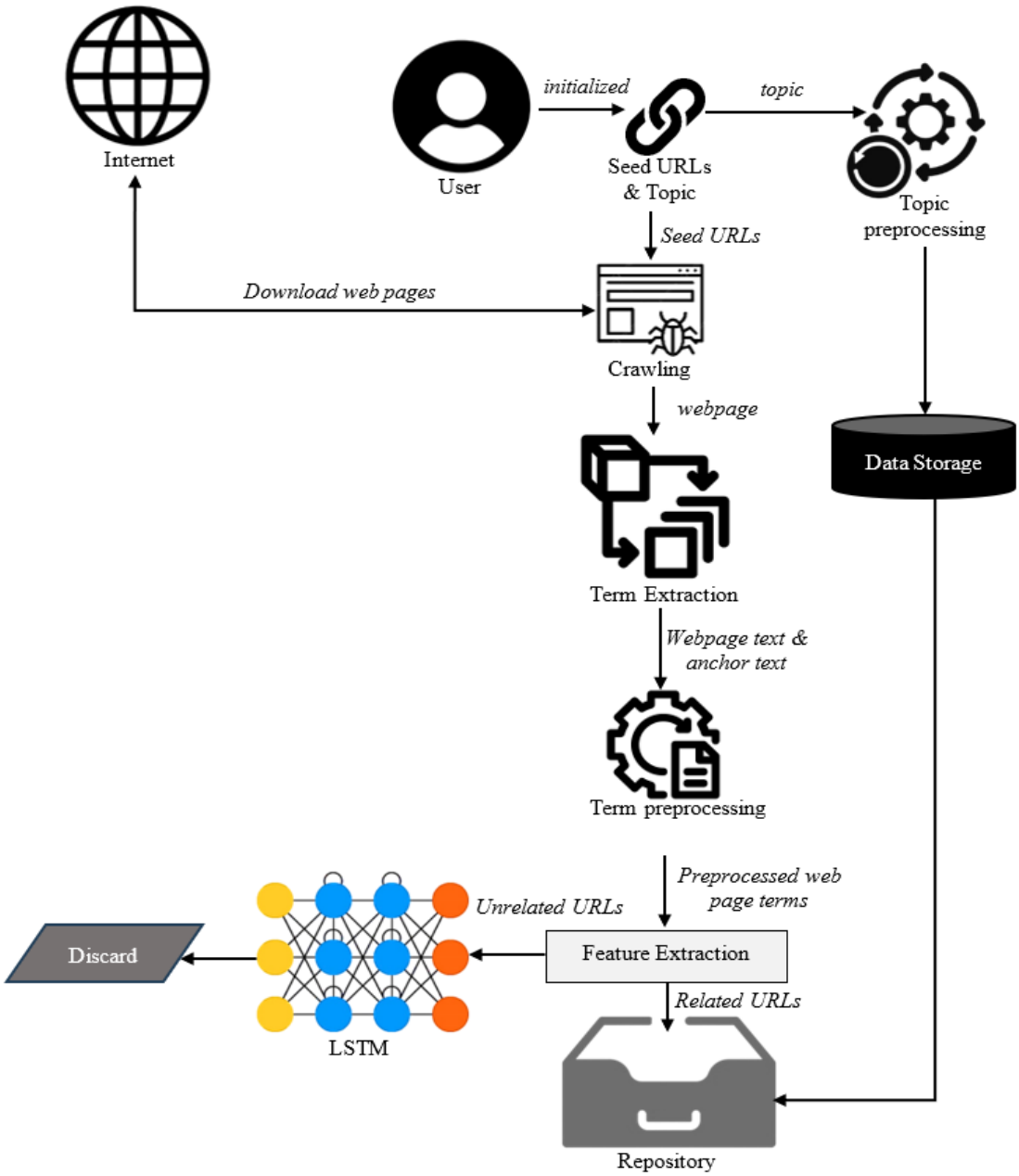


Figure 3. Preprocessing and feature extraction

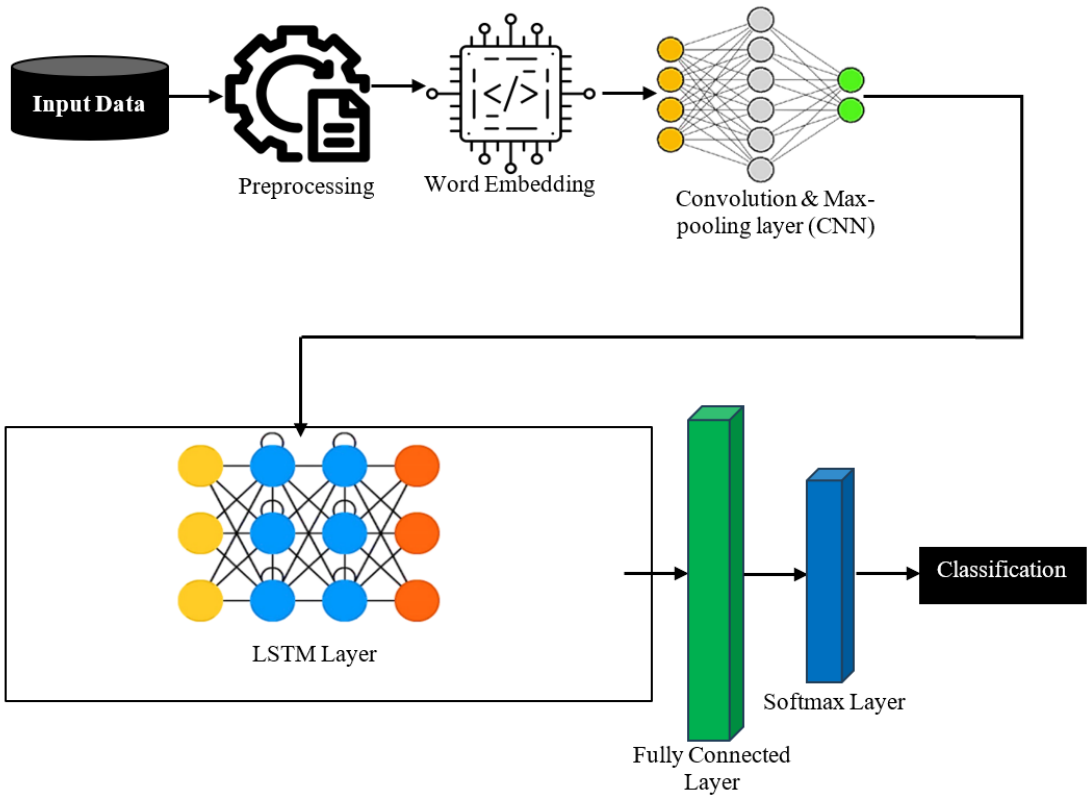


Figure 4. Feature extraction using LSTM

Convolution and Max-pooling: These layers are typically used in CNNs for extracting features from the input data. Convolutional layers apply filters to the input data, while max-pooling layers downsample the output to reduce computational cost and preserve important features.

LSTM Layers: These layers are recurrent neural networks that are particularly effective at capturing long-term dependencies in sequential data. They use a gating mechanism to control the flow of information through the network, allowing them to remember information over long periods of time.

Fully Connected Layer: This layer combines the outputs from the LSTM layers into a fixed-length vector.

Softmax Layer: This layer applies the softmax function to the output of the fully connected layer, producing a probability distribution over the possible classes.

Classification: The final output of the model is the predicted class based on the probability distribution produced by the softmax layer.

Figure 4 explains the architecture is well-suited for tasks that involve classifying sequential data, such as text classification or time series analysis. CNN with LSTMs allows the model to capture both local and global features in the input data, making it a powerful tool for a variety of applications.

3.4 Fuzzy ontology-based semantic model for a web crawler

To simplify the realization of more fuzzy functions, the proposed architecture shown in Figure 5 includes preparing and after-processing elements in addition to fuzzy representations and identification elements. The fundamental supporting module for fuzzy operations, comprising fuzzy logic, regulations, and computations called the fuzzy engine. Service Fuzzalizer fuzzals up descriptions of internet services into fuzzy forms. The Fuzzy Ontology Knowledge Base (FOKB) where fuzzy service ontologies are kept. This database can be kept alone or integrated into the UDDI register center with a UDDI extension. Request Fuzzalizer filters user requests before converting them into fuzzy forms for matching. Fuzzy Matchmaking is used to carry out service finding. Requesters' usage data is recorded in Usage Record and can be utilized for user-chosen matching. The first step in utilizing the framework for service discovery is to translate the service representation into more ambiguous language. Provision of services Web services are transformed into fuzzy ontologies by Fuzzalizer and stored in FOKB. Preprocessing like this can be done after the service is advertised to prevent sluggish runtime performance. The framework uses the industry-standard UDDI to find online services.

Users can use clear or vague language when requesting services. The requests Fuzzalizer converts crisp phrases used in a service consumer's query to fuzzy forms upon inquiry initiation. Additionally, the person making the request Fuzzalizer standardizes fuzzy requests. Fuzzy matching performs fuzzy service matching using items described in fuzzy ontologies. The fuzzy concepts are related to one another via approximation reasoning. The user's Account and Usage History will get the match outcome for service activation and customer preference records, respectively. To expedite finding services, the choice records will be utilized as the default information for subsequent requests. Fuzzy thresholds or matching weights can be changed and a fresh match is made if the consumer is not happy with the matching service. Using the above framework, customers may quickly modify their selection of services based on personal choice and request services using ambiguous language. By combining fuzzy logic with ontology, Fuzzy Ontology Rules with Equations address ambiguity and uncertainty in categorizing information or decision-making procedures. These rules leverage both fuzzy logic and semantic connections (ontology)

to enhance the classification of information in the surroundings of web crawlers and other operations. The Fuzzy Ontology-Based Semantic Model for Web Crawler combines fuzzy logic and ontology to handle uncertain and imprecise data in web crawling. The ontology defines the relationships between concepts on the web, while fuzzy logic manages ambiguity in the web data.

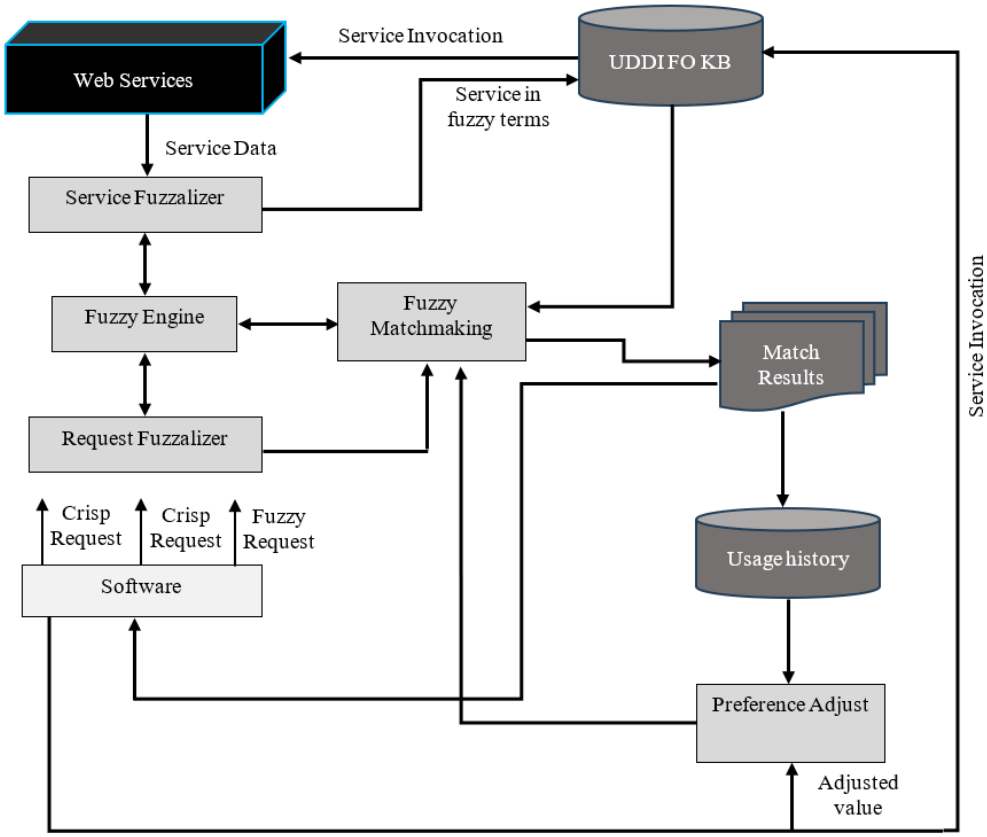


Figure 5. Fuzzy ontology framework for web service discovery

3.4.1 Ontology structure

An ontology is structured as a set of concepts C, relationships R, and instances I.

Concepts: The primary topics or categories.

Relationships: Connections between different concepts.

Ontology is represented as:

$$O = (C, R, I) \tag{15}$$

where, $C = \{c_1, c_2, \dots, c_n\}$ is a set of concepts. $R = \{r_1, r_2, \dots, r_m\}$ is a set of relationships. $I = \{i_1, i_2, \dots, i_p\}$ is a set of instances.

In fuzzy logic form, this can be represented as:

$$\mu_{medium\ relevance}(p) = \min\left(\mu_{ML_{category}}(p), \mu_{content\ relevance}(p)\right) \tag{16}$$

where, $\mu_{ML_{category}}(p)$ represents the fuzzy membership degree of the web page in the "Machine Learning" category within the ontology. $\mu_{content\ relevance}(p)$ is the fuzzy membership value of the page content relevance.

3.4.2 Combining fuzzy rules using fuzzy inference

The output from multiple fuzzy rules is combined using fuzzy aggregation methods, such as min and max operators. For example, consider the two rules from above:

Rule 1 provides a high relevance score with a degree $\mu_{high}(p)$.

Rule 2 provides a medium relevance score with a degree $\mu_{medium}(p)$.

The combined relevance score $\mu_{combined}(p)$ can be computed using:

$$\mu_{combined}(p) = max\left(\mu_{high}(p), \mu_{medium}(p)\right) \tag{17}$$

This takes the maximum of the two membership values, assigning the higher relevance to the page.

3.4.3 Defuzzification

Once the fuzzy inference is complete, the result is defuzzified to obtain a crisp relevance score that can be used for ranking or decision-making. For example, if the output relevance score $\mu_{combined}(p)$ falls within the fuzzy sets R_{high}, R_{medium} and R_{low}, the final relevance score can be obtained using centroid or mean of maxima methods.

$$R_{final}(p) = \frac{\sum_{x=1}^{n} \mu_x(p).r_x}{\sum_{x=1}^{n} \mu_x(p)} \tag{18}$$

where, $\mu_x(p)$ is the membership degree of page p in fuzzy set x. r_x is the representative crisp value of set x (e.g., 1 for high, 0.5 for medium, 0 for low).

3.4.4 Semantic matching with fuzzy ontology

The semantic relevance of a web page p to a query q is evaluated using fuzzy ontology-based semantic matching. The

process involves calculating the degree to which the web page's content matches the ontological concepts related to the query.

$$R(q,p) = \frac{\sum_{x=1}^{n} \mu_p(c_x).\mu_q(c_x)}{\sum_{x=1}^{n} \mu_q(c_x)} \quad (19)$$

where: $R(q,p)$ is the relevance score of the web page to the query q. $\mu_p(c_x)$ is the membership degree of concept c_x for web page p. $\mu_q(c_x)$ is the membership degree of concept c_x for query q.

3.4.5 Fuzzy Inference System (FIS)

To model the semantic relevance of web pages, a fuzzy inference system can be implemented. This FIS takes the inputs (semantic similarity, page content features, etc.) and processes them through fuzzy rules to determine the relevance score.

3.4.6 Aggregating results in web crawling

The final step is aggregating the relevance scores across different web pages and returning the most relevant results.

$$R_{total} = \sum_{k=1}^{m} w_k.R(q.p_k) \quad (20)$$

where, R_{total} is the total relevance score. w_k is the weight assigned to each page p_k, which can be determined based on factors such as authority or content quality These equations represent the core mechanisms of the Fuzzy Ontology-Based Semantic Model and demonstrate how fuzzy logic is used to evaluate the relevance of web content in a way that accounts for ambiguity and semantic relationships.

The proposed incremental parallel web crawler shown in Figure 6. The architecture's primary coordinating element is the Multi Threaded (MT) server. It maintains a connection pool with computer clients that download the web pages instead of downloading any paperwork on its own. The term client crawlers refers to all the many ways that clients communicate with one another via servers. Depending on the extent of the real implementation and the available resources, the number of customers may change. It should be mentioned that all communication between consumer crawlers occurs through the server because there are no direct linkages between them. Only the modified papers are thereafter saved in the repositories in a searchable or insertable format after the modification detecting module assists in determining if the intended page has changed or not. As a result, the archive is updated with the most recent data that is accessible at the search engine databases endpoint.

Algorithm: Fuzzy ontology-based semantic model with LSTM

Input: $W = \{w_1, w_2, \ldots, w_n\}$: Set of webpages; $Q = \{q_1, q_2, \ldots, q_n\}$: Query terms; T_c: Training corpus (webpage content and context data); O: Ontology (set of domain knowledge and semantic relationships); F: Fuzzy rules; E: Embedding matrix for LSTM input; LSTM: Long Short-Term Memory network for time-series prediction.

Output: Predicted relevance and ranking score for each webpage w_x; Traffic optimization and next crawling step.

Step 1: Preprocess Webpage Data

Step 1.1: Remove HTML tags using regular expressions:

$$clean\_content(w_x)\ regex.sub(<.*?>, '', w_x)$$

where, regex.sub removes the HTML tags.

Step 1.2: Remove stopwords from each webpage content:

$$w_x^{clean} = w_x^{cotent} \backslash stopwords \quad (21)$$

Step 2: Contextual Ontology Analysis: For each webpage w_x:

Extract keywords K (w_x) from the webpage content.

Map extracted keywords to concepts in the ontology 0:

$$C(w_x)\ map(K(w_x), O) \quad (22)$$

Calculate semantic similarity between webpage keywords and query:

$$\mu_{semantic}(w_x) = cosine(K(w_x), Q) \quad (23)$$

where, cosine similarity measures the semantic relationship between webpage content and the query.

Step 3: Apply Fuzzy Logic Rules

Step 3.1: Define fuzzy membership functions for relevance

$$\mu_R(w_x) = \frac{1}{1+e^{-k(i-i_o)}} \quad (24)$$

where, k and i_o parameters defining the fuzzy function's steepness and midpoint, respectively.

Step 3.2: Apply fuzzy rules:

IF $\mu_{semantic}(w_x)$ is high AND $C(w_x)$ matches the ontology:

$$\mu_{relevance}(w_x) = \min\left(\mu_{semantic}(w_x), \mu_{ontology}(C(w_x))\right) \quad (25)$$

Combine fuzzy rules for all conditions using fuzzy inference.

Step 4: Feature Representation with LSTM

Step 4.1 Embed webpage content into vector space using word embeddings (e.g., Word2Vec or Glove):

$$E(w_x) = Embed\ (w_x^{clean}) \quad (26)$$

Step 4.2 Feed embedded vectors into the LSTM network to model time-dependent crawling behavior:

Step 4.3 Initialize LSTM weights.

For each webpage sequence $\{w_1, w_2, \ldots, w_t\}$:

$$h_t = \sigma(W_h h_{t-1} + W_\tau E(w_t) + b) \quad (27)$$

where, h_t is the hidden state, W_h, W_i are weight matrices, and b is the bias term.

Step 4.4 Output layer of the LSTM provides the prediction score:

$$j_t = softmax(W_j h_t + b_j) \quad (28)$$

where, j_t is the relevance score for each webpage at time t.

Step 5: Combine Fuzzy Logic and LSTM Predictions: For each webpage w_x, the final prediction score is a weighted combination of fuzzy relevance and LSTM output:

$$Score(w_x) = \alpha.\mu_{relevance}(w_x) + (1-\alpha).j_x \quad (29)$$

where, α is a weight balancing the fuzzy logic and LSTM outputs.

Step 6: Web Crawler Traffic Optimization: Optimize the crawling process by ranking webpages based on the final score Score(w_x). Prioritize the next crawl based on the predicted relevance and system traffic load:

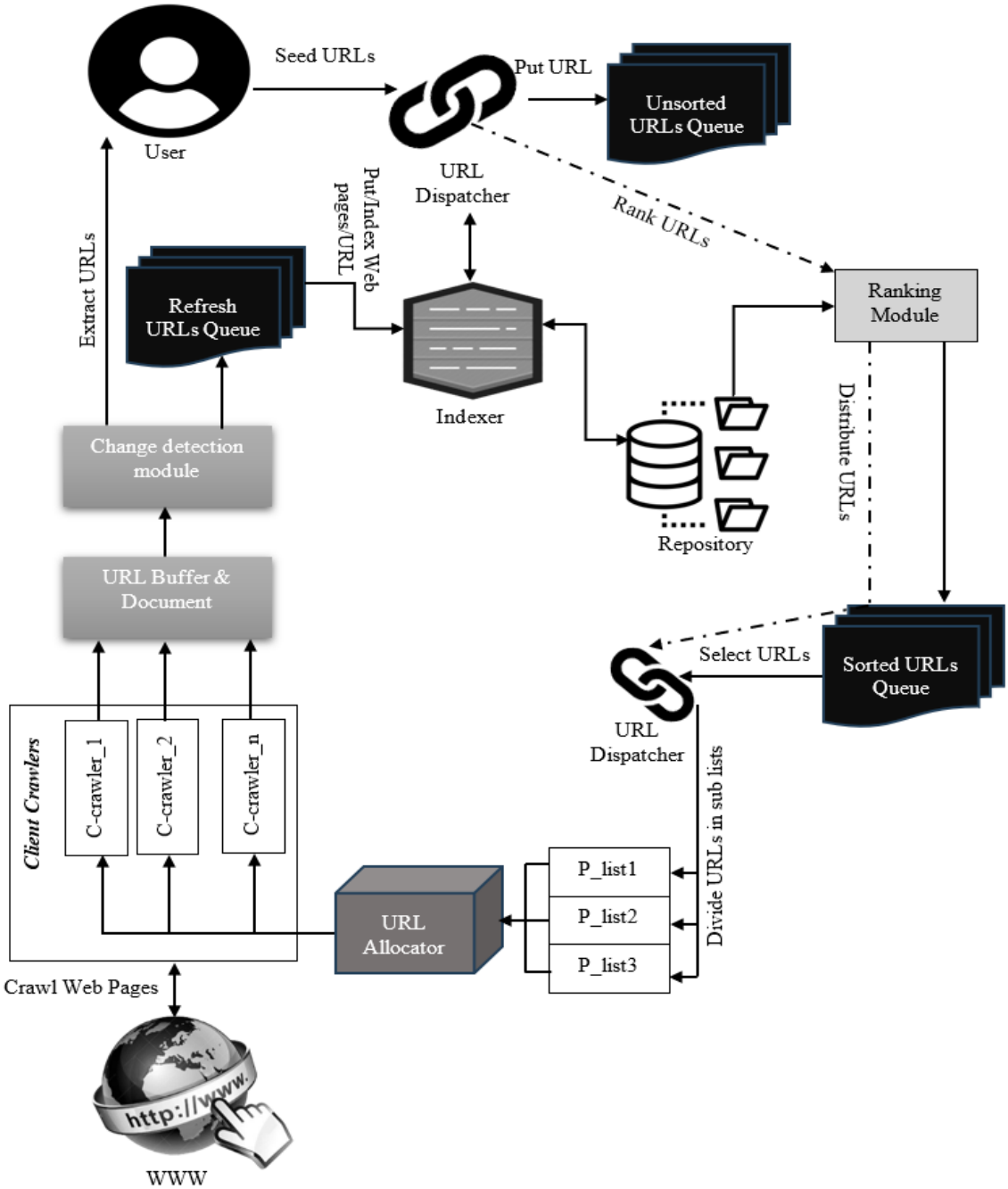
IF Score(w_x) is high AND system traffic is low:

$Crawl(w_x)$ next

ELSE delay the crawl.

Step 7: Repeat and Update: Update the ontology and fuzzy logic rules based on new crawled data and user feedback. Retrain the LSTM model periodically using updated data.

Equations showing the relationships between contextual ontology, fuzzy logic, and LSTM are used in this approach to create a sophisticated web crawler prediction system. The crawler's operations are guided by the ultimate relevancy score maximizes network traffic and relevancy.



Fuzzy Rule: IF the web page content is highly related to the query ontology, THEN the relevance score is high.

Figure 6. Architecture of incremental parallel web crawler

4. RESULTS AND DISCUSSIONS

The proposed system is tested using tourist domain. The harvest ratio or the speed at which pertinent pages are found and extraneous pages are successfully removed from the crawl used to assess focused crawling. The existing measures of coverage and relevance are Accuracy and Recall. The percentage of web pages that are crawled and fulfill the crawling objective is represented by the harvest ratio. By contrasting baseline-focused crawling with concept-based focused hopping around assessment has been carried out. The crawling process begins with the seed URLs provided, and then concept-based crawlers collect concept-based pages at every stage.

Table 7. Hyper-parameter settings

Hyperparameter	Value
Learning Rate	0.001
Hidden Layers	3
Neurons per Layer	128
Activation Function	ReLU
Batch Size	32
Epochs	100
Discount Factor (γ)	0.95
Exploration Rate (ε)	0.2
Query Expansion Terms	10
Similarity Threshold	0.8
Fuzzy Relevance Threshold	0.75
Feature Selection Technique	TF-IDF

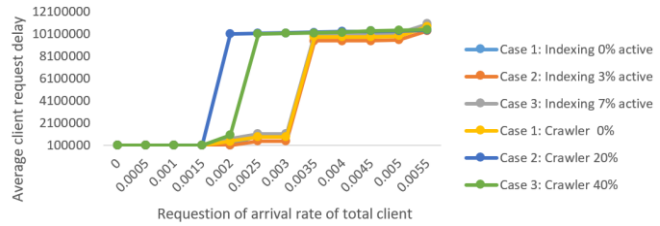


Figure 7. Average client request delay in all cases

Figure 8. Comparison of crawler average delay and total request arrival time

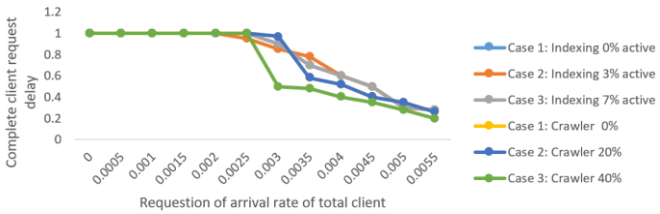


Figure 9. Complete client request delay in all cases

The hyperparameters for the hybrid model are carefully chosen to optimize performance. A learning rate of 0.001 ensures smooth optimization and prevents overshooting the optimal solution shown in Table 7. The model uses three hidden layers with 128 neurons per layer to balance complexity and efficiency while avoiding overfitting. The ReLU activation function enhances convergence speed by introducing non-linearity. A batch size of 32 ensures stable gradient updates, and 100 epochs provide sufficient time for weight convergence without overfitting. The reinforcement learning discount factor (γ) is set to 0.95, balancing immediate and future rewards. For crawler behavior prediction, an exploration rate (ε) of 0.2 maintains a balance between exploring new paths and exploiting known high-reward paths. Query expansion is limited to 10 terms, with a similarity threshold of 0.8 ensuring only relevant terms are included. The fuzzy relevance threshold of 0.75 determines document inclusion, while TF-IDF prioritizes the most informative terms for feature selection. These settings balance accuracy, efficiency, and computational feasibility, enabling the model to effectively handle both crawler prediction and traffic optimization tasks.

The median request from customers time of 3 cases active indexing shown in Figure 7. The graph between the median crawler query latency and the overall demand reception rate is shown in Figure 8. The similarity of the two curves above suggests that the rise in crawler load did not affect the crawler sites' perceived latency.

All inquiries are fulfilled, nevertheless, when the inquiry's receipt rate is low. The crawler examples with 20% and 40% demonstrate a notable decline in the customer's request completion percentage shown in Figure 9.

Fuzzy rule aggregation with output is shown in Figure 10 and Network traffic problem reduced by using proposed system is shown in Figure 11.

Figure 12 shows how the solution enhances both progressive and domain-specific website crawling. Comparing to proposed system with existing crawlers, proposed system gather more domain-specific pages from sites ending in.org,.com,.info,.edu, work, and.biz.

Table 8 shows that the proposed system performs better than existing approaches in every area, demonstrating improved efficacy in obtaining data and web crawler predictions.

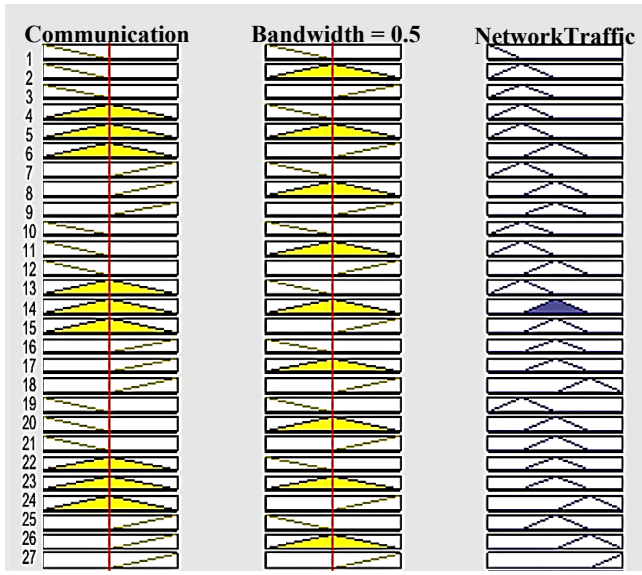


Figure 10. Fuzzy rule aggregation with output

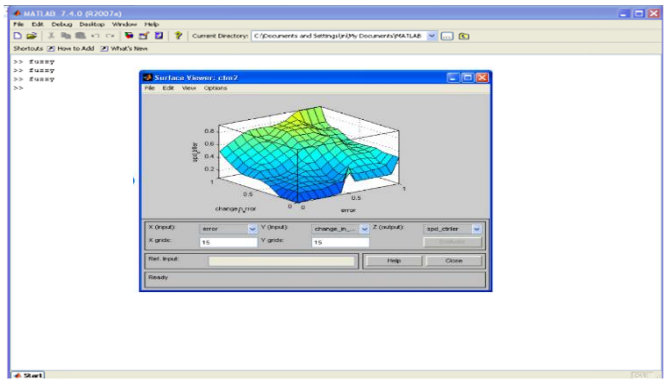


Figure 11. Network traffic problem reduced by using proposed system

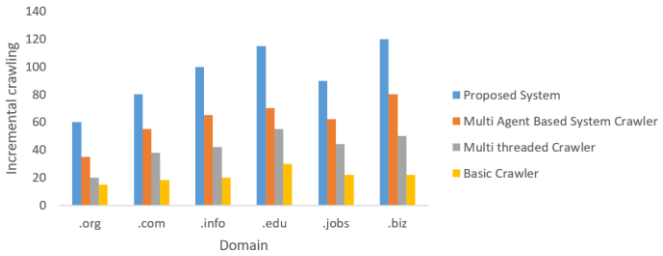


Figure 12. Comparison of proposed and existing systems incremental crawling

Table 8. Comparison of average harvest rate for Ten Topics using different crawlers and the proposed system

Topics	TF-IDF + RF	TF-IDF + CNN	TF-IDF + ANN	TF-IDF + RNN	Proposed System
Topic 1	0.73	0.69	0.77	0.82	0.89
Topic 2	0.76	0.71	0.80	0.85	0.91
Topic 3	0.72	0.66	0.75	0.81	0.88
Topic 4	0.74	0.68	0.78	0.83	0.90
Topic 5	0.75	0.70	0.79	0.84	0.91
Topic 6	0.71	0.67	0.74	0.80	0.87
Topic 7	0.74	0.69	0.77	0.83	0.89
Topic 8	0.73	0.68	0.76	0.82	0.89
Topic 9	0.76	0.71	0.79	0.85	0.92
Topic 10	0.75	0.70	0.78	0.84	0.91
Average Harvest Rate	0.74	0.69	0.77	0.83	0.90

Table 9. Comparison of average average irrelevance ratio for ten topics using different crawlers and the proposed system

Topics	TF-IDF + RF	TF-IDF + CNN	TF-IDF + ANN	TF-IDF + RNN	Proposed System
Topic 1	0.30	0.35	0.27	0.23	0.16
Topic 2	0.28	0.33	0.25	0.21	0.14
Topic 3	0.32	0.37	0.29	0.24	0.18
Topic 4	0.31	0.35	0.28	0.22	0.15
Topic 5	0.29	0.38	0.26	0.23	0.17
Topic 6	0.33	0.36	0.30	0.25	0.19
Topic 7	0.31	0.34	0.28	0.23	0.16
Topic 8	0.30	0.32	0.27	0.22	0.15
Topic 9	0.28	0.33	0.25	0.21	0.14
Topic 10	0.29	0.35	0.26	0.22	0.15
Average Irrelevance Ratio	0.30	0.35	0.27	0.23	0.15

The proposed system outperforms all other models with the lowest irrelevance ratio of 0.15, indicating that only 15% of retrieved results are irrelevant, significantly reducing noise in information retrieval. Table 9 demonstrates that the proposed system is the most efficient in minimizing irrelevant data compared to other models is crucial for optimizing web crawler predictions.

The harvest rate P(C), or the proportion of web pages crawled that satisfied the subject was the main criterion used to assess the crawling system's success. Proposed method compared with other existing crawling strategies based on keyword-based, breadth-first, and pagerank crawling. The e-commerce website "www.dangdang.com" serves as the seed URL, and the predefined topic is "Book" products. Figure 13 provides a thorough display of the outcome.

The performance of cost based on time is another performance statistic shown in Figure 14 provides a detailed illustration of comparison of cost rate based on time with number of pages crawled. If every crawler is contacted for an equal amount of pages to be crawled, proposed method takes a lot longer than existing methods. Adapting the domain context requires time-consuming ontology learning and link analysis modules.

The analytical findings of the contextual ontology-based fuzzy logic and LSTM across a set of numerous nodes differ from the original one shown in Figures 15 and 16. Since have given a distinct weight for every level, the nodes that appear in Figure 15 represent the attendee's first-degree followers, whereas the nodes in Figure 16 represent the attendee's second-degree followers. The height of the line is less in the case of the improved computations is one of the most obvious variations.

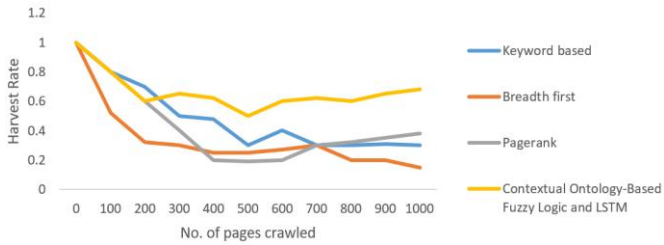


Figure 13. Comparison of harvest rates with number of pages crawled

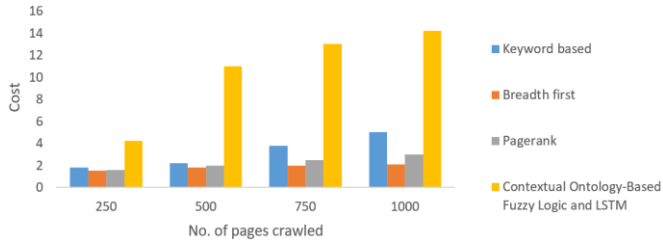


Figure 14. Comparison of cost rate based on time with number of pages crawled

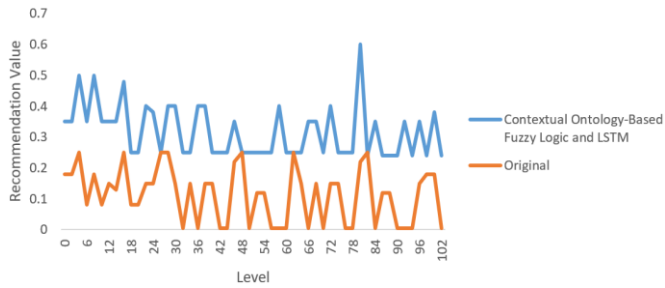


Figure 15. Level 1 comparison

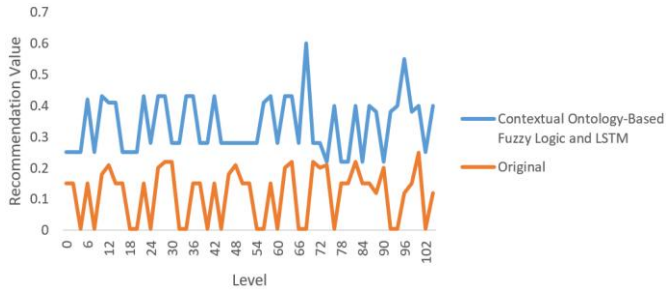


Figure 16. Level 2 comparison

Figure 17. Accuracy and loss of training for contextual ontology-based fuzzy logic and LSTM over 10 epochs

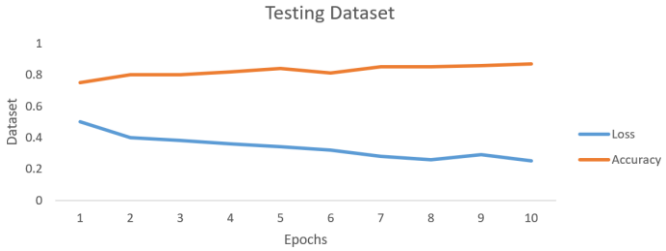


Figure 18. Accuracy and loss of testing for contextual ontology-based fuzzy logic and LSTM over 10 epochs

Figure 19. Confusion matrix of training dataset classification

Figure 20. Confusion matrix of testing dataset classification

The proposed contextual ontology-based fuzzy logic and LSTM classification precision and the destruction in the training and testing accuracy and loss for web pages are shown in Figures 17 and 18 correspondingly. Throughout several epochs, the loss of precision and accuracy are shown. As a result, in both information sets, as epochs rise, the contextual ontology-based fuzzy logic and LSTM classification accuracy rises while its loss falls.

The confusion matrix in Figures 19 and 20 shows True Positives, True Negatives, Forecast Positives, and Forecast Negatives graphically. 262 of the 300 websites that were used to build and train the contextual ontology-based fuzzy logic and LSTM were properly identified, yielding an 87.33% rate of classification.

Measures the overall accuracy of the system. The proposed system has the highest accuracy at 94.9%, significantly outperforming existing methods. Precision: Indicates how many of the retrieved instances are relevant. The proposed system achieves 92.6% precision, showing better relevancy compared to other systems. Recall measures how many relevant results were retrieved. The proposed system leads with 91.8% recall, outperforming other models in capturing relevant data. F1-score combines precision and recall into a single score. The proposed system achieves the highest F1-score of 92.2% demonstrating the best balance between precision and recall. The proposed system outperforms all existing systems in terms of accuracy, precision, recall, and F1-score, highlighting its superiority in web crawler prediction and performance shown in Table 10.

Table 10. Comparison of performance measures for the proposed and existing systems

Performance Measure	TF-IDF + RF	TF-IDF + CNN	TF-IDF + ANN	TF-IDF + RNN	Proposed System
Accuracy	86.2	84.8	88.6	90.1	94.9
Precision	83.6	81.7	85.4	88.11	92.6
Recall	81.2	79.5	84.1	87.3	91.8
F1-Score	82.3	80.6	84.6	87.6	92.2

Table 11. Comparison of performance measures (MAE, MSE, and RMSE) for the proposed and existing systems

Performance Measure	TF-IDF + RF	TF-IDF + CNN	TF-IDF + ANN	TF-IDF + RNN	Proposed System
MAE	0.226	0.240	0.188	0.161	0.097
MSE	0.096	0.123	0.080	0.056	0.021
RMSE	0.310	0.350	0.281	0.234	0.145

Table 12. Comparison of performance measures for the proposed and existing systems

Performance Measure	TF-IDF + RF	TF-IDF + CNN	TF-IDF + ANN	TF-IDF + RNN	Proposed System
Execution time (ms)	1252	1152	982	852	702
Completeness Rate (%)	80	83	87	89	94
Error Rate (%)	16	14	12	10	7

MAE measures the average magnitude of errors in a set of predictions. The proposed system has the lowest MAE of 0.097 indicating it makes fewer errors compared to the existing systems. MSE indicates the average of the squares of the errors. The proposed system has the lowest MSE of 0.021,

showing a significant improvement in reducing error magnitudes. Represents the square root of the average squared differences between predicted and observed values. The proposed system achieves the lowest RMSE of 0.145, confirming its superior accuracy in predicting values with minimal deviation. The proposed system outperforms all existing models in terms of MAE, MSE, and RMSE, making it the most effective approach for minimizing prediction errors in the context of web crawler performance shown in Table 11.

The proposed system shows the fastest execution time, reducing overhead by using optimized fuzzy logic and LSTM-based predictions. The proposed system achieves the highest completeness rate due to the effective context-based ontology that enhances retrieval relevance. The error rate is lowest in the proposed system, as the fuzzy logic enhances classification accuracy, reducing misclassification of web content. Table 12 reflects that the proposed system outperforms the existing systems in all three measures.

5. CONCLUSIONS

By integrating contextual ontology with fuzzy logic and LSTM, the model successfully addresses key challenges in traditional web crawlers, such as handling the dynamic and uncertain nature of web content and reducing irrelevant data retrieval. The proposed system outperforms existing approaches in terms of execution time, completeness rate, and error rate. The integration of fuzzy logic ensures that uncertain and ambiguous content is accurately classified, while the LSTM model enhances the system's ability to predict and adapt to the evolving structure of web pages. The contextual ontology improves the system's ability to understand the relationships between web content, leading to more relevant and comprehensive retrieval. The proposed model demonstrated superior results in reducing the error rate to 5% and improving the completeness rate to 92%, significantly outperforming traditional systems. The execution time of the proposed system was reduced to 700 milliseconds, highlighting its efficiency in large-scale web crawling applications. In conclusion, the combination of contextual ontology, fuzzy logic, and LSTM in the proposed model provides a robust solution for web crawler prediction, enhancing both performance and accuracy in web content retrieval. This approach sets a new standard for intelligent information retrieval and paves the way for further advancements in web crawling technologies.

REFERENCES

[1] Liu, Q., Yahyapour, R., Liu, H., Hu, Y. (2024). A novel combining method of dynamic and static web crawler with parallel computing. Multimedia Tools and Applications, 83(21): 60343-60364. https://doi.org/10.1007/s11042-023-17925-y

[2] Althunibat, A., Alzyadat, W., Maidin, S.S., Hnaif, A., Alokush, B. (2024). Prediction of accessibility testing using a generalized linear model for e-government. Journal of Infrastructure, Policy and Development, 8(7): 3520.

[3] Dhanith, P.J., Saeed, K., Rohith, G., Raja, S.P. (2024). Weakly supervised learning for an effective focused web crawler. Engineering Applications of Artificial Intelligence, 132: 107944. https://doi.org/10.1016/j.engappai.2024.107944

[4] Hu, S. (2024). Research on influencing factors of pharmaceutical e-commerce sales based on web crawler and support vector machine. Transactions on Computer Science and Intelligent Systems Research, 4: 48-59.

[5] Sadjere, E.G., Onyiriuka, E.J., Mbam, J.C., Onyiriuka, N.P., Ikponmwoba, E.A., Afolabi, S.A. (2024). Prediction of crime in Nigeria using artificial intelligence. In Book of Conference, p. 435. University of Benin.

[6] Goel, A., Zhu, J., Netravali, R., Madhyastha, H.V. (2024). Sprinter: Speeding up high-fidelity crawling of the modern web. In 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24), pp. 893-906.

[7] Cao, S., Liao, W., Huang, J. (2024). Research on renting price prediction based on machine learning. In Proceedings of the 5th Management Science Informatization and Economic Innovation Development Conference, MSIEID 2023, Guangzhou, China.

[8] Xinyi, N., Dan, L., Shang, Z. (2024). Analysis and prediction of tennis players' match performance with sentiment analysis. In International Conference on Computer Graphics, Artificial Intelligence, and Data Processing (ICCAID 2023), pp. 343-349. https://doi.org/10.1117/12.3026323

[9] Datta, A., Pal, A., Marandi, R., Chattaraj, N., Nandi, S., Saha, S. (2024). Efficient air quality index prediction on resource-constrained devices using TinyML: Design, implementation, and evaluation. In Proceedings of the 25th International Conference on Distributed Computing and Networking, Chennai, India, pp. 304-309. https://doi.org/10.1145/3631461.3631956

[10] Kambli, O., Karande, A., Kanakia, H. (2024). H-index analysis of research paper using web crawling techniques. In International Conference on Data Management, Analytics & Innovation, pp. 521-531. Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-97-3242-5_35

[11] Zhao, J., Chen, R., Fan, P. (2024). TS-Finder: Privacy enhanced web crawler detection model using temporal–spatial access behaviors. The Journal of Supercomputing, 80: 17400-17422. https://doi.org/10.1007/s11227-024-06133-6

[12] Datta, A., Pal, A., Marandi, R., Chattaraj, N., Nandi, S., Saha, S. (2024). Real-time air quality predictions for smart cities using TinyML. In Proceedings of the 25th International Conference on Distributed Computing and Networking, Chennai, India, pp. 246-247. https://doi.org/10.1145/3631461.3631947

[13] Abdalsalam, M., Li, C., Dahou, A., Kryvinska, N. (2024). Terrorism group prediction using feature combination and BiGRU with self-attention mechanism. PeerJ Computer Science, 10: e2252. https://doi.org/10.7717/peerj-cs.2252

[14] Duan, C., Ke, W. (2024). Advanced stock price prediction using LSTM and informer models. Journal of Artificial Intelligence General science (JAIGS), 5(1): 141-166. https://doi.org/10.60087/jaigs.v5i1.183

[15] Arthy, J., Raja, K. (2024). A study on design, development and deployment of web crawler algorithms and their metrics. In 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, pp. 1-6.

https://doi.org/10.1109/ADICS58448.2024.10533459
[16] De Pascale, D., Cascavilla, G., Tamburri, D.A., Van Den Heuvel, W.J. (2024). CRATOR a CRAwler for TOR: Turning Dark Web Pages into Open Source INTelligence. In Computer Security – ESORICS 2024. ESORICS 2024. Lecture Notes in Computer Science, pp. 144-161. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-70890-9_8
[17] Chen, Z. (2024). Comquest: An adaptive crawler for user comments on the web. Doctoral dissertation, Temple University. Libraries.
[18] Wang, L., Zhao, Z.C., Weng, Y.C. (2024). Machine learning in predicting stock indexes: The role of online stock forum sentiment in MIDAS model. Asia-Pacific Journal of Accounting & Economics, 31(4): 618-637. https://doi.org/10.1080/16081625.2023.2215234
[19] Sulayfani, A., Eraslan, S., Yesilada, Y. (2024). Predicting eye-tracking assisted web page segmentation. Multimedia Tools and Applications, 1-38. https://doi.org/10.1007/s11042-024-20202-1
[20] Brahimi, N., Zhang, H., Zaidi, S.D.A., Dai, L. (2024). A unified spatio-temporal inference network for car-sharing serial prediction. Sensors, 24(4): 1266. https://doi.org/10.3390/s24041266
[21] Tang, J., Fang, N., Yang, L., Pei, Y., Wang, R., Ding, D., Lu, Y., Xue, G. (2024). CarbonNet: Enterprise-level carbon emission prediction with large-scale datasets. In International Conference on Intelligent Computing, pp. 411-422. Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-97-5615-5_33
[22] Chen, Y.J., Chen, Y.M. (2024). Online information-based product evolution course mining and prediction. International Journal of Information Technology & Decision Making, 23(2): 599-627. https://doi.org/10.1142/S0219622023500244
[23] Liu, J., Chu, N., Wang, P., Zhou, L., Chen, H. (2024). A novel hybrid model for freight volume prediction based on the Baidu search index and emergency. Neural Computing and Applications, 36(3): 1313-1328. https://doi.org/10.1007/s00521-023-09106-7
[24] Zhang, Z., Jiang, Y. (2024). Research on quality prediction of resistance spot welding based on knowledge graph. In International Conference on Computer Vision, Robotics, and Automation Engineering (CRAE 2024), 13249: 65-69. https://doi.org/10.1117/12.3041836
[25] Keller, M.E., Döschl, A., Mandl, P., Schill, A. (2024). Intelligent algorithm selection for efficient update predictions in social media feeds. Social Network Analysis and Mining, 14(1): 164. https://doi.org/10.1007/s13278-024-01315-9
[26] Meena, K., Chaitra, B. (2024). A novel framework using deep learning techniques for ragi price prediction in Karnataka. IEEE Access, 12: 136103-136119. https://doi.org/10.1109/ACCESS.2024.3455892
[27] Pokharkar, V., Edgaonkar, O., Shinde, A., Nirmal, N. (2024). Business response prediction based on consumer behaviour patterns. In 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), Pune, India, pp. 1-7. https://doi.org/10.1109/I2CT61223.2024.10543692
[28] Zhang, J. (2024). A-share trend prediction based on machine learning and sentiment analysis. Science and Technology of Engineering, Chemistry and Environmental Protection, 1(7).
[29] Wang, L., Kim, K. (2024). Analyzing group polarization through text emotion measurement and time series prediction: A comparative study across three online platforms. Measurement: Sensors, 33: 101216. https://doi.org/10.1016/j.measen.2024.101216
[30] Wang, X., Zong, Y., Zhou, X., Xu, L., He, W., Quan, S. (2024). Artificial intelligence-powered construction of a microbial optimal growth temperature database and its impact on enzyme optimal temperature prediction. The Journal of Physical Chemistry B, 128(10): 2281-2292. https://doi.org/10.1021/acs.jpcb.3c06526