Data Dynamic Prediction Algorithm in the Process of Entity Information Search for the Internet of Things

Tianqing Liu
College of Artificial Intelligence and Big Data, Zibo Vocational Institute, Zibo, 255000, China

***Abstract*—To address the issue of insufficient real-time capability in existing Internet search engines within the Internet of Things environment, this research investigates the architecture of Internet of Things search systems. It proposes a data dynamic prediction algorithm tailored for the process of entity information search in the Internet of Things. The study is based on the design of a data compression algorithm for the Internet of Things entity information search process using the Rotating Gate Compression Algorithm. The algorithm employs the Least Squares Support Vector Machine to dynamically predict changes in entity node states in the Internet of Things, aiming to reduce sensor node resource consumption and achieve real-time search. Finally, the research introduces an Internet of Things entity information search system based on the data dynamic algorithm. Performance test results indicate that the segmented compression algorithm designed in the study can enhance compression accuracy and compression rate. As compression accuracy increases, errors also correspondingly increase. The prediction algorithm designed in the study shows a decrease in node energy consumption as reporting cycles increase, reaching 0.2 at 5 cycles. At the 5-cycle point, the prediction errors on two research datasets are 0.5 and 7.8, respectively. The optimized data dynamic prediction algorithm in the study effectively reduces node data transmission, lowers node energy consumption, and accurately predicts node state changes to meet user search demands.**

Keywords—Internet of Things; sensors; swinging door trending (SDT); support vector machine (SVM); data dynamic prediction

I. Introduction

In the era of the Internet of Things (IoT), massive devices continuously generate vast streams of data containing crucial insights for entity information search. With the rapid development of IoT technology, effectively processing and predicting this data to support real-time, accurate information search has become a significant research topic. The data dynamic prediction algorithm for entity information search in the IoT aims to predict dynamic changes in data in the IoT environment using advanced data analysis techniques to enhance search efficiency and accuracy [1-2]. The complexity of the IoT environment is mainly reflected in the high dynamism, large-scale, and heterogeneity of data. These characteristics make traditional data processing and prediction methods challenging to adapt. For example, IoT devices generate diverse data types with a fast update frequency, influenced by various factors such as environmental changes, device status, and user behavior. Therefore, developing an algorithm that can accurately predict these dynamic data changes is crucial for efficient IoT entity information search. Currently, IoT data processing and analysis rely mainly on traditional data processing techniques and algorithms. However, these methods face numerous challenges when handling large-scale, highly dynamic, and heterogeneous IoT data. On the one hand, unlike the traditional Internet, the object of IoT search is structured and highly dynamic, and traditional search engines cannot adapt to IoT information search. On the other hand, the Internet of Things has a large number of nodes, diverse data types, and fast update frequencies. Traditional search strategies have poor timeliness and waste a large amount of computing resources [3]. Based on this, research has been conducted on the architecture of IoT search systems, proposing a data dynamic prediction algorithm for IoT entity information search processes. The innovation of the research is mainly in two aspects, one of which introduces a lightweight data compression method to compress the original data and reduce the network communication pressure. The second introduces a minimized quadratic loss function to improve the support vector machine for real-time search. Additionally, the research considers the real-time and dynamic nature of data to ensure that the algorithm can adapt to the rapid changes in IoT data. This research is expected to provide new theoretical and practical references in the field of IoT data processing and analysis on the one hand; on the other hand, it promotes the development and innovation of IoT applications, and has great potential to provide richer and smarter services for people.

The research is divided in six sections. Section II delves into related. Section III presents an IoT entity information search system based on the data dynamic prediction algorithm. Section IV tests and analyzes the performance of the model and algorithm. Results and discussion is given in Section V. Section VI summarizes and concludes the above content.

II. Related Works

In the field of sensor networks, numerous studies focus on data transmission and application. Sreedharan et al. proposed a cluster head selection and hybrid routing protocol algorithm based on fuzzy multi-attribute decision-making to enhance the performance of wireless sensor networks. Subsequent experiments demonstrated the effectiveness of their algorithm [4]. Pattnaik et al. introduced an algorithm that combines fuzzy clustering and elephant herd optimization to achieve efficient routing assembly in wireless sensor networks, with simulation experiments validating its advantages in energy utilization and system lifespan [5]. Javaheri et al. conducted research on data fusion techniques in wearable sensors, comparing Kalman fusion and alpha-fine-tuned mean filtering. They designed an algorithm that switches between the two fusion methods based on Signal Quality Index, with experimental results showing its superiority over baseline dual-channel RR interval averaging by approximately 54% and 21%, respectively [6]. The IoT, as a network connecting everything, involves processes where sensor-acquired data is uploaded to the network and then processed by artificial intelligence or corresponding algorithms to issue instructions for desired actions by executive devices. As an emerging concept, IoT has been widely studied in various fields. Turner et al. focused on reducing the environmental impact of the manufacturing industry by aligning with practices in digital computation and the automotive industry. They constructed an IoT network framework for the circular economy model in manufacturing based on Industry 4.0 computing and IoT interaction networks [7].

Zhang et al. addressed the issue of battery replacement in large-scale IoT deployments by employing wireless power transfer using radio frequencies. Based on DEIN technology, they developed a time allocation model to manage radio frequency energy and the transmission of uplink data in different time slots. Additionally, they proposed a joint time and power allocation algorithm to reduce the energy consumption per bit for uplink data transmission in the system [8]. Mabodi et al. focused on the ability of IoT nodes to provide general intelligent predictions and used a method to reduce gray hole attacks by inspecting node information. The approach involved validating trust in IoT nodes, testing routes, detecting gray hole attacks, and eliminating malicious attacks in the MTISS-IoT process. Experimental results demonstrated a high detection rate [9]. Dynamic data prediction algorithms have been studied in various fields, and Huang et al. conducted research on the numerical simulation process of Steam-Assisted Gravity Drainage (SAGD). They developed and tested a workflow based on a data-driven model for predicting SAGD production performance. A comparison of different machine learning algorithms indicated that a model based on Gated Recurrent Units (GRU) exhibited the best predictive capability [10]. Xiao et al. considered the spatiotemporal dynamic characteristics for air quality prediction and proposed a Dual-Path Dynamic Directed Graph Convolutional Network (DP-DDGCN) for air quality prediction. The model aimed to comprehensively capture dynamic spatial dependencies, and experimental results showed its superior performance in predicting PM2.5 concentration [11].

In summary, although scholars have conducted extensive research on algorithms and functional optimizations to enhance IoT performance, more research has been done on application improvement, node prediction, and performance enhancement for IoT only. There has been limited optimization of the entity information search process. However, the search optimization for dynamic data content of sensor nodes has important potential application value in IoT performance transmission. Therefore, the research started from three aspects: compression of data resources, dynamic prediction, and real-time search, and designed an IoT entity information search system based on data dynamic algorithms, which to a certain extent fills the research gap of entity information search.

III. Design of Data Dynamic Prediction Algorithms for Entity Information Search in IoT

Traditional IoT search methods involve broadcasting search requests to each sensor node, enabling real-time acquisition of dynamically changing node status information. However, this approach leads to high communication overhead, bandwidth consumption, and consequently, a reduction in the lifespan of nodes. To address this challenge, research has been conducted on data dynamic prediction algorithms, proposing a novel architecture for an IoT search system to overcome the inadequacy of existing internet search engines in providing real-time results within the IoT environment.

A. *Design of IoT Entity Information Search System Based on Data Dynamic Prediction Algorithms*

The study focused on researching data dynamic prediction algorithms for IoT entity information search processes, integrating web technologies into the IoT domain. They proposed a Web of Things (WoT) search system based on data dynamic prediction algorithms. The architecture of this search system is illustrated in Fig. 1.

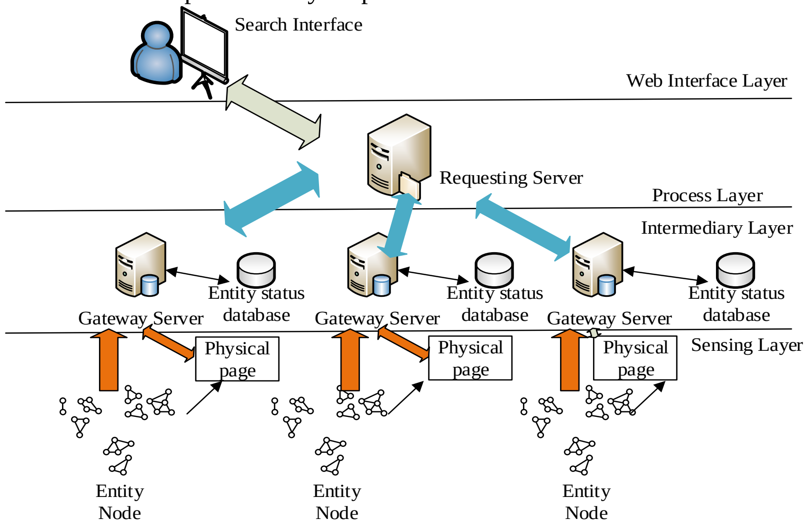


Fig. 1. WoT search system.

As shown in Fig. 1, the WoT Search System is primarily divided into four layers, each serving different functions to achieve efficient, flexible data processing, and user interaction. The Sensing Layer constitutes the core of the system's data source, consisting of numerous sensor nodes responsible for collecting environmental data. These data undergo initial compression processing within the nodes themselves to reduce transmission burdens. The Mediation Layer (Gateway Service Layer) acts as the hub for data processing and forwarding, collecting data from the Sensing Layer for further analysis and prediction, ensuring real-time and accurate data. Simultaneously, this layer responds to external query requests, retrieving and filtering relevant data. The Processing Layer (Request Service Layer) serves as the direct processing center for user requests, analyzing user queries, interacting with the Mediation Layer, integrating acquired data, and optimizing the presentation order of results. The User Interface Layer provides users with an intuitive, user-friendly interface, presenting complex backend data processing results in a clear and concise manner to meet specific user query requirements. This search system is based on existing search technology and optimized for the characteristics of IoT network architecture.

B. Data Compression Algorithm based on SDT Compression Algorithm

In order to enhance the efficiency and accuracy of network searches, data compression technology is employed to reduce data transmission volume and enable accurate querying of IoT node statuses through efficient prediction of sensor data. This approach effectively improves the utilization of network and server resources, ensuring users obtain satisfactory search results. Among various compression algorithms, the study compared lossy and lossless compression algorithms, and the Swinging Door Trending (SDT) algorithm, characterized by lower compression rates, lower time complexity, simplicity, and minimal system resource usage, emerged as a suitable choice for compressing sensor data in IoT sensor networks [12]. Refer to Fig. 2 for an illustrative diagram of the SDT algorithm.

The SDT algorithm is a compression algorithm designed for time series data. It introduces the concept of a "swinging door" that allows data to fluctuate within a preset error range. When data points deviate from the set threshold, the algorithm creates a new data segment. The core of SDT lies in balancing data fidelity and compression rate, maintaining sufficient data information while eliminating redundant data. This method is particularly effective in industrial process control and real-time data monitoring systems, significantly reducing data storage and transmission requirements while ensuring critical information is not lost. The SDT algorithm's implementation process selects the first sampled data point of the sensor as the starting point, and the sampled value sequence is as shown in Eq. (1).

$$P = \{t_i, p(t_i)\} \quad i = 1, 2, \cdots, n \tag{1}$$

In Eq. (1), where i represents the sampling time point and $p(t_i)$ is the corresponding sensor value at time t_i, as shown in Fig. 2, A is the starting point of a data segment, and the sampled value is denoted as $p(t_A)$. B is a sampling point after A, with a sensor data compression precision Δe, and the trend line calculation Equations are given by Eq. (2) and Eq. (3).

$$K_{AB}^1 = \frac{p(t_B) + \Delta e - p(t_A)}{t_B - t_A} \tag{2}$$

Eq. (2) represents the upper limit of the trend line, where $p(t_B)$ represents the sampled value of B, and the calculation Equation for the lower limit of the trend line is given by Eq. (3).

$$K_{AB}^2 = \frac{p(t_B) - \Delta e - p(t_A)}{t_B - t_A} \tag{3}$$

Eq. (3) is the calculation Equation for the lower limit of the trend line. After adding C to the samples, the slope K_{AC} of the line segment AC is calculated using Eq. (4).

$$K_{AC} = \frac{p(t_C) - p(t_A)}{t_C - t_A} \tag{4}$$

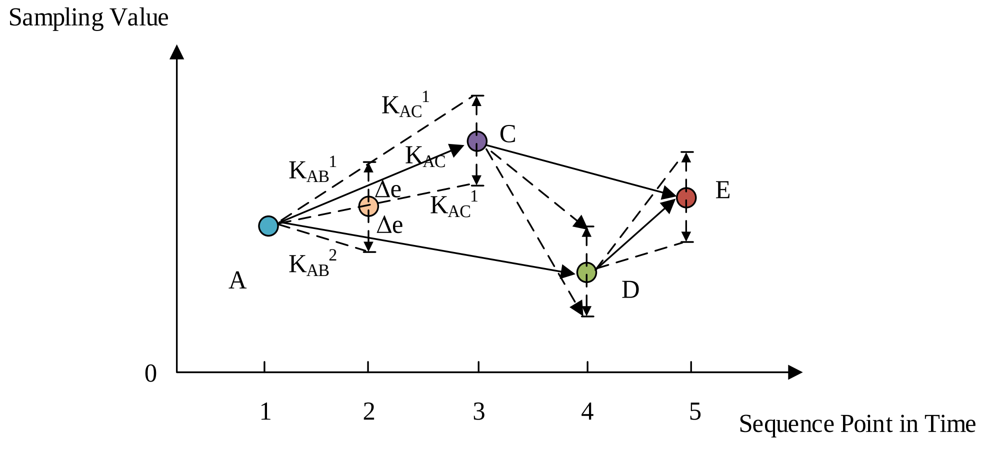


Fig. 2. SDT algorithm.

In Eq. (4), if $K_{AB}^1 \leq K_{AC} \leq K_{AB}^2$, it indicates that the segmentation compression condition is satisfied. Compression is performed on point C, and trend lines K_{AC}^2 and K_{AC}^1 are calculated following Eq. (2) and Eq. (3) for further assessment. Otherwise, if K is not equal to A, it indicates that the condition is not satisfied. The data at point B is saved, and the slope of the trend line BC is recalculated for further assessment. This operation is repeated for different data points until all data points are compressed, ensuring that the compression precision loss for each sampling point is less than Δe. For decompression, the first step is to extract the sensor data set UT that needs to be decompressed. All the data in $U\{t_i, p(t_i)\}$ are restored, assuming adjacent data points $\{t_i, p(t_i)\}$ and $\{t_j, p(t_j)\}$ have a difference expression as given in Eq. (5).

$$diff = (t_j - t_i) / t_s \tag{5}$$

In Eq. (5), $t_s = t_i - t_{i-1}$, representing the sampling time interval. When $diff = 1$, there are no sampling points between the two points, and the process moves on. If $diff > 1$, the restoration is performed, and the calculation is given by Eq. (6).

$$p(t_x) = \frac{p(t_j) - p(t_i)}{t_j - t_i}(t_x - t_i) + p(t_i), \quad x = i, i+1, \cdots, j \tag{6}$$

This operation is repeated until all data points are restored. To address the slowly varying characteristics of sensor data, an improved rotating gate algorithm is proposed. When the sensor state is stable for a long time, only the initial state is stored to reduce data redundancy. When a significant change in state occurs, the rotating gate algorithm is activated for compression. The algorithm takes the first data point as a reference, sets a threshold, and judge's subsequent points: if the change is less than the threshold, it is ignored; if it exceeds the threshold, it is recorded, and the new point becomes the reference for continued processing. This method effectively reduces data volume while retaining key changes. The flowchart of the segmentation compression algorithm is shown in Fig. 3.

As illustrated in Fig. 3, the decompression process begins by assessing whether the difference between two consecutive sampled points is less than a predefined threshold. If the difference is small, the value of the first point is directly used as the intermediate point value. If the difference is large, the standard decompression method of the rotation gate algorithm is employed. The key advantage of this algorithm lies in its handling of stable time periods. In time ranges where node states change minimally, only one state value needs to be stored, significantly improving compression efficiency. This strategy outperforms traditional rotation gate algorithms in time periods with minimal state changes. While the improved algorithm and the original algorithm perform similarly in data segments with frequent changes, considering that IoT nodes often experience stable states, the improved algorithm effectively reduces compression ratios and errors overall, despite a slight increase in compression time. Experimental results demonstrate that this enhanced algorithm is particularly suitable for scenarios with minimal state changes in IoT search systems, effectively reducing data storage requirements within an acceptable time increase range.

C. IoT Entity Information Prediction Algorithm Design Based on LS-SVM

The study employs the Least Squares Support Vector Machine (LS-SVM) to dynamically predict changes in entity node states in IoT, aiming to reduce sensor node resource consumption and achieve real-time search [13-15]. The schematic diagram of the LS-SVM model is shown in Fig. 4.

LS-SVM is a supervised learning method based on statistical learning theory and is a variant of the traditional Support Vector Machine (SVM). It addresses classification and regression problems by minimizing a quadratic loss function, as opposed to the hinge loss function in traditional SVM, simplifying the problem-solving process. The parameters of the predictive model established in the research are presented in the figure and outlined in Table I.

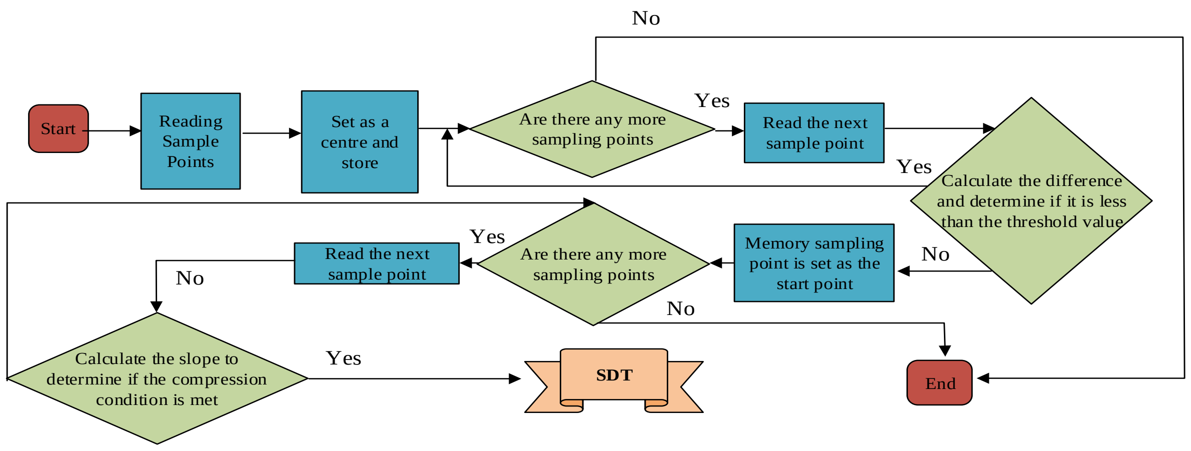

Start
Reading Sample Points
Set as a centre and store
Are there any more sampling points
Yes
No
Read the next sample point
Calculate the difference and determine if it is less than the threshold value
Yes
No
Memory sampling point is set as the start point
Are there any more sampling points
Yes
No
Read the next sample point
No
Calculate the slope to determine if the compression condition is met
Yes
SDT
End

Fig. 3. Segmented compression algorithm flowchart.

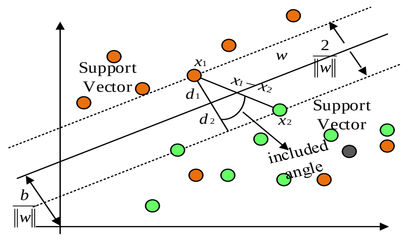


Fig. 4. LS-SVM Model schematic diagram.

TABLE I. MODEL PARAMETERS AND PROCESS

Input Parameter	✧Sensor History Sampling Value List X_{tr} ✧parametric $sigma$ ✧parametric $gamma$ ✧Projected point in time $time$
Output Parameter	✧Predicted sensor values $result$
Algorithm steps	
Step 1	Construct the historical data input matrix X and output matrix Y from X_{tr}
Step 2	Compute the $omega$ value of the kernel function matrix from X
Step 3	Compute the value of the symmetric semipositive definite matrix A from X $omega$, $gamma$
Step 4	Compute the value of the intermediate variable b from the semipositive definite matrix A and the output matrix Y
Step 5	The value of the Lagrange factor a is obtained from the values of A , Y and b
Step 6	According to the intermediate variables and related parameters obtained from the calculation to obtain the final prediction function $f(x)$, input the predicted time point $time$, so as to obtain the predicted value of the moment $result$

Firstly, the input-output matrices X and Y are constructed based on sampled data. Following the principle of minimizing structural risk, the optimal nonlinear regression function is transformed into an optimization problem, as shown in Eq. (7).

$$\min J(w,e)=\frac{1}{2}w^T w+\frac{1}{2}\gamma\sum_{i=1}^{n}e_i^2 \tag{7}$$

In Eq. (7), w^T represents the transpose of the weight vector w , γ is a tunable parameter controlling the penalty for samples outside the error range of the function, n denotes the matrix dimension, and e_i represents the deviation between predicted and actual values. By applying the Lagrangian method to solve this function, it is transformed into a constrained problem, as illustrated in Eq. (8).

$$L(w,b,e,a)=J(w,e)-\sum_{i=1}^{n}a_i(w^T\varphi(x_i)+b+e_i-y_i) \tag{8}$$

In Eq. (8), a_i represents the Lagrange multiplier, $\varphi(x_i)$ denotes the non-linear mapping of x_i , b signifies the bias term, and x_i and y_i belong to the matrix X and the matrix Y . Employing the partial derivative method to eliminate w and e yields a set of Equations as shown in Eq. (9).

$$\begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & \varphi(x_1)^T\varphi(x_1)+\frac{1}{\gamma} & \cdots & \varphi(x_1)^T\varphi(x_n) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi(x_n)^T\varphi(x_1) & \cdots & \varphi(x_1)^T\varphi(x_n)+\frac{1}{\gamma} \end{bmatrix}\begin{bmatrix} b \\ a_1 \\ \vdots \\ a_n \end{bmatrix}=\begin{bmatrix} 0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} \tag{9}$$

Solving Eq. (9) allows us to determine the values of a and b , leading to the construction of Eq. (10).

$$f(x)=\sum_{i=1}^{n}a_i\varphi(x_i)^T\varphi(x)+b \tag{10}$$

Utilizing Mercer's transformation on Eq. (10), we derive the kernel function as depicted in Eq. (11).

$$K(x_i,x_j)=\varphi(x_i)^T\varphi(x_j) \tag{11}$$

Within Eq. (11), $K(x_i,x_j)$ represents the kernel function. Based on this kernel function, the calculation Equation for the non-linear predictive model is shown in Eq. (12).

$$f(x)=\sum_{i=1}^{n}a_iK(x,x_i)+b \tag{12}$$

The selected kernel function for study is the Gaussian radial basis kernel function, with the final predictive function depicted in Eq. (13).

$$f(x)\approx\sum_{i=1}^{n}a_i e^{\frac{\|x-x_i\|^2}{2\sigma^2}}+b \tag{13}$$

In Eq. (13), σ denotes the kernel width, and $\|x-x_i\|^2$ represents the norm between vectors. LS-SVM commonly utilizes a grid-search method in conjunction with cross-validation to determine parameters and find the combination that minimizes prediction error. This method involves constructing a grid within the parameter space to conduct a global search. Throughout the computation, a sparse grid reduces computational load but might miss the optimal parameters, while a dense grid enhances precision but increases computational complexity. This research adopts a multi-grid approach, initially defining a parameter range, selecting a set of parameter points within this range for training, adjusting the grid based on error results, gradually narrowing the search scope until finding the optimal parameter pair. This method enhances parameter optimization efficiency while controlling computational load. The system's functional module design based on the above algorithm is illustrated in Fig. 5.

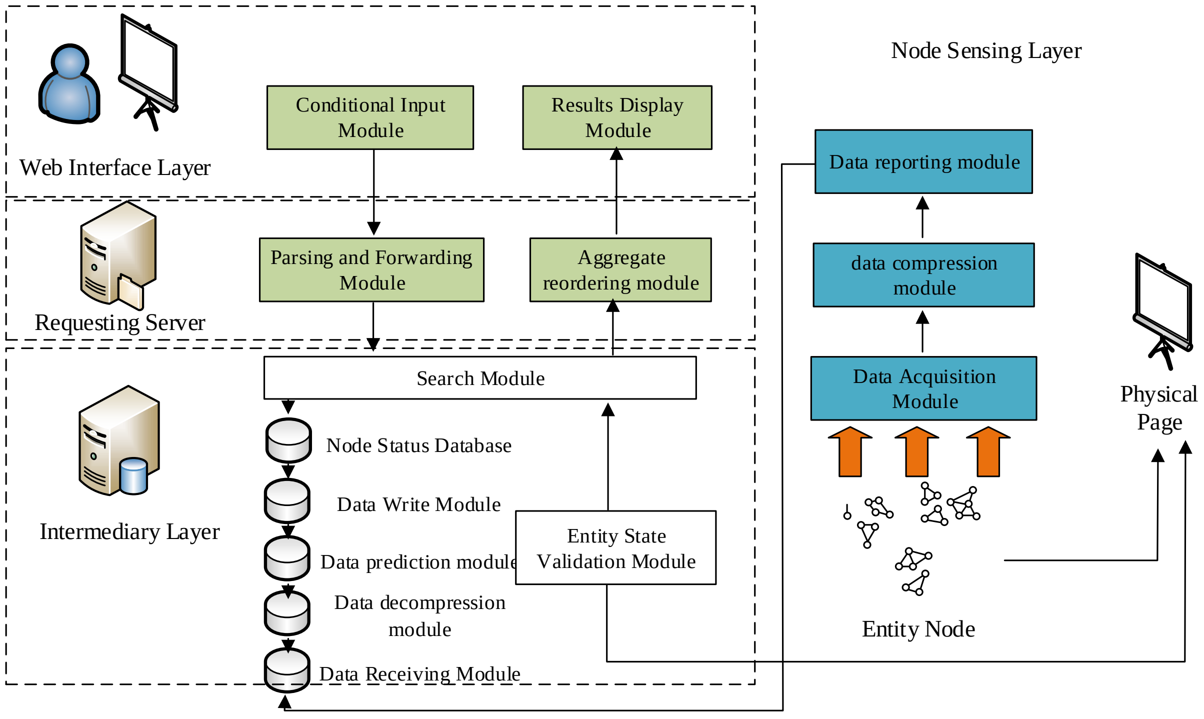


Fig. 5. WOT system functional module design.

The constructed WOT search system comprises three core layers: the node perception layer, the gateway service layer, and the frontend interface layer, developed using Python and Java. In the node perception layer, multiple simulated sensor nodes run as independent threads, periodically collecting and compressing data, then transmitting it to the gateway service layer through a data reporting module. Additionally, each node maintains a web page, recording real-time status. The server in the gateway service layer queues the received data, sequentially decompresses and restores it, utilizes the prediction module to forecast future states, and updates them in the status database. The frontend web interface layer serves as the user interaction point, receiving search requests via a browser. These requests, once parsed, are forwarded to the gateway server. The search module within the gateway server retrieves nodes meeting the criteria from the database, conducts relevance scoring and status validation, and ultimately feeds back the sorted results to the frontend for display. Despite sharing the same hardware platform in actual deployment, the request server and gateway server are logically independent, ensuring seamless system functionality. The entire search system, customized based on an open-source search engine, effectively implements the required search flow.

IV. Performance Testing of WoT System Based on Data Dynamic Prediction Algorithm

The simulated testing in this study focuses on an IoT search platform utilizing a data dynamic prediction algorithm. The testing process involves collecting multi-dimensional time series data from various physical entities, such as car engines and study room environmental data. The collected data is classified, cleaned, and ensured of its validity. After processing, the data is stored as text, with each text representing a node for testing purposes. The platform employs compression techniques to reduce data transmission, saving energy. Additionally, a prediction mechanism forecasts the future state of nodes, providing accurate search results. The main focus of the testing lies in evaluating the platform's capabilities in scenarios like fault detection and resource retrieval. The dataset used in the research is presented in Table II.

Due to limited sources for simulated testing data, the research concentrates on scenarios such as searching for a suitable study room and finding an appropriate exercise location. The first simulation scenario involves students looking for study rooms, considering factors like temperature, humidity, and light intensity. The system calculates relevance scores for study rooms, ranks them, and helps users find the most suitable place to study. The second scenario involves searching for exercise locations based on user-specified conditions like temperature, PM2.5 levels, and wind speed. The system provides a list of parks sorted according to the specified conditions, assisting users in choosing the best exercise location. The original data used in the testing follows a specific format. The research selects compression accuracy as a measurement parameter for compression algorithms. By setting different levels of accuracy, the compression effects are compared, as shown in Fig. 6.

TABLE II. RESEARCH-CONSTRUCTED DATASET

Causality	Corridor	Data format	Unit (of measure)	Example
Dates	/	Year-Month-Day	/	2004/2/28
Times	/	Hours:Minutes:Seconds. Milliseconds	/	39:46.2
Serial number	/	Integers	/	84
Node number	/	Integers	/	18.9594
Temperature value	Variation interval	Floating point type	Degrees centigrade	38.8039
Humidity value	[0,100]	Floating point type	Per cent	43.24
Light intensity	Positive number	Floating point type	Lux	2.68742
Voltage level	[0,4]	Floating point type	Vodka	/
PM2.5 data set				
Causality	Corridor	Data format	Unit (of measure)	Example
Point in time	/	"Year-Month-Day Hour:Minute:Second"	/	"2010/1/2 0:00:00"
PM2.5 concentration	Positive number	Floating point type	Micrograms per cubic metre	129
Temperatures	Float range	Floating point type	Degrees	-4
Barometric Pressure Indicator	Float range	Floating point type	Megapascal	1020
Wind speed	Positive number	Floating point type	metres/second	1.79

Fig. 6(a), Fig. 6(b), Fig. 6(c), and Fig. 6(d) present the compression test results for the segmented compression algorithm designed in the study on temperature, humidity, light intensity, and voltage data from the IntelLab dataset. Overall, it is observed that increasing compression accuracy leads to higher compression rates, as higher accuracy allows for more data loss. However, different sensor types exhibit varying sensitivities to changes in compression accuracy; humidity sensors show lower sensitivity, while temperature sensors demonstrate a more significant response. This suggests that different sensor types have unique compression behaviors. With the growth of compression accuracy, errors also increase accordingly, aligning with compression logic. Therefore, selecting an appropriate compression accuracy based on sensor data characteristics is crucial for balancing errors and reducing data volume. After testing, the research selects compression accuracies as follows: 0.5 for temperature, 1.0 for humidity, 1.0 for light intensity, 0.2 for voltage, 3.0 for PM2.5, 0.7 for humidity, 3.0 for air pressure, and 0.5 for wind speed. The study compares the proposed segmented compression algorithm with the rotation gate algorithm using the IntelLab dataset, and the results are shown in Fig. 7.

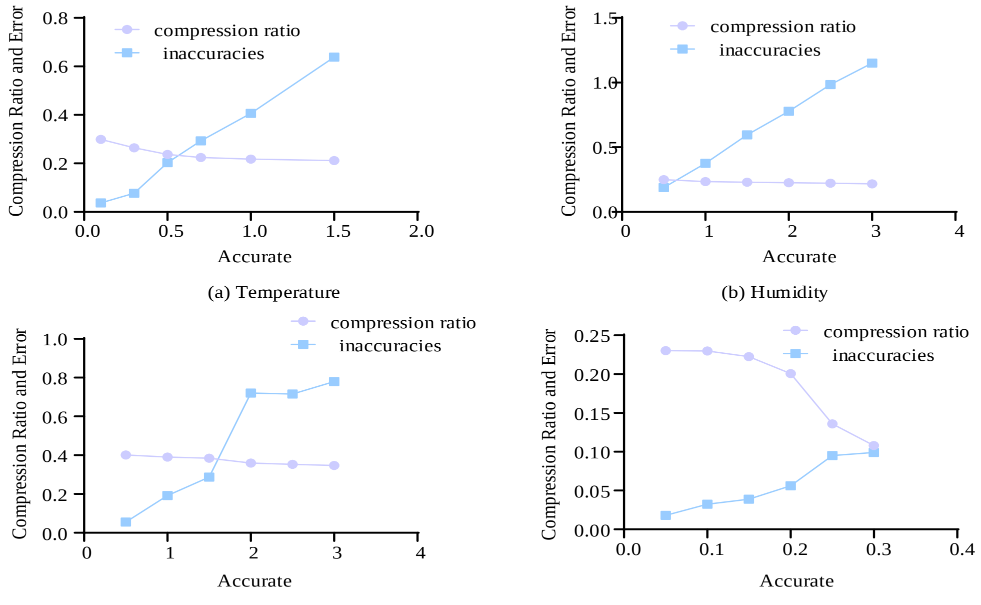
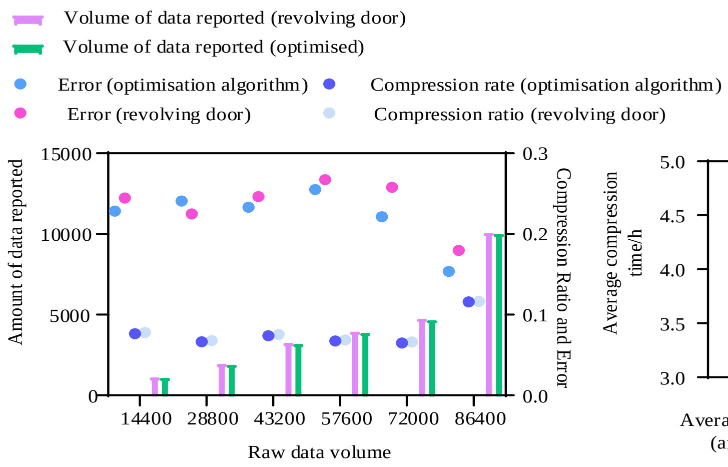
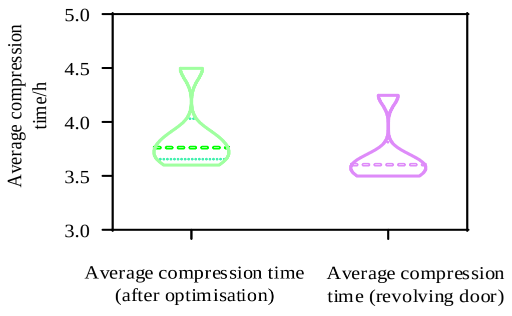


Fig. 6. Compression algorithm test results.



(a) Comparison of data volume and compression rate



(b) Comparison of average compression times

Fig. 7. Comparison of segmental compression algorithm and rotation gate algorithm test results.

The results of the comparison test between the proposed Segmental Compression Algorithm and the Rotation Gate Algorithm are illustrated in Fig. 7. Due to the regularity of the dataset, both the original Rotation Gate Algorithm and the Segmental Compression Algorithm exhibit stable performance in terms of compression rate and error. As seen in Fig. 7(a), the improved algorithm shows a 6.14% reduction in compression rate, a 5.95% decrease in error. However, as seen in Fig. 7(b), an increase in compression time by 13.12%. This performance improvement is attributed to the improved algorithm using initial points instead of the stable interval, optimizing the compression effect despite increased complexity and processing time. In order to achieve the goal of reducing node energy consumption, the study proposes the Segmental Compression Algorithm based on the Rotation Gate Algorithm to reduce the amount of data reported by sensor nodes. To verify its energy-saving performance, the study tests its energy savings and time relationship on two datasets, as shown in Fig. 8.

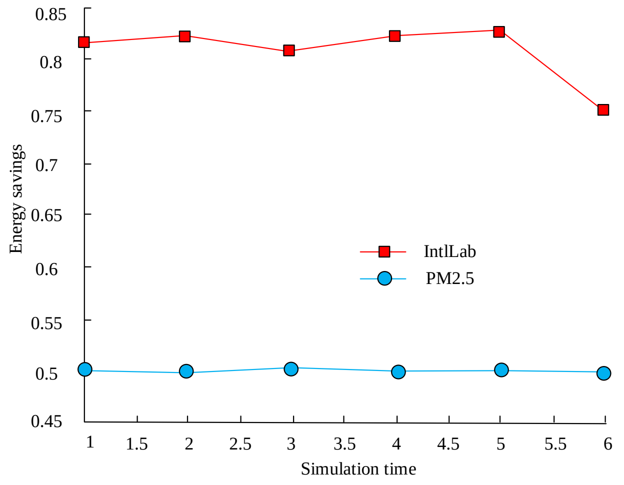


Fig. 8. Energy savings and time relationship of the segmental compression algorithm.

Fig. 8 presents the experimental results of the energy savings of the Segmental Compression Algorithm on the IntelLab and PM2.5 datasets over time. By applying the data compression module, energy consumption decreases by approximately 82% and 50%, respectively. This significant reduction in energy consumption is attributed to the decrease in the amount of data uploaded by nodes, reducing frequent data reporting and saving network bandwidth. Moreover, this method effectively extends the running time of nodes and the overall lifespan of the network. The training results of the prediction algorithm are shown in Table III.

TABLE III. PREDICTION ALGORITHM TRAINING RESULTS

IntelLab dataset				
Causality	Training time	Best *sigma*	Best *gamma*	Prediction error
Temp	7.417	1.537	0.927	0.516
Humidity level	15.026	2.04	0.939	0.274
Sunlight	15.058	3.941	0.902	0.086
Input voltage	14.54	8.264	-0.05	0.355
PM2.5 data set				
Causality	Training time	Best *sigma*	Best *gamma*	Prediction error
PM2.5	7.594	1.202	0.996	3.809
Temp	15.48	0.641	1.021	0.342
Pneumatic	16.017	1.819	0.895	0.238
Air velocity	15.21	1.029	0.997	2.043

As shown in Table III, setting a certain reporting period and reducing the frequency of terminal node data reporting can effectively reduce energy consumption. However, this may lead to users searching for historical data, affecting accuracy. To address this issue, a prediction module is introduced to predict the next cycle state of nodes, balancing energy consumption and search accuracy. The impact of different reporting periods on the system is analyzed using the energy consumption formula, with mean square error and prediction time as evaluation indicators. Simulation results of the prediction algorithm are shown in Fig. 9.

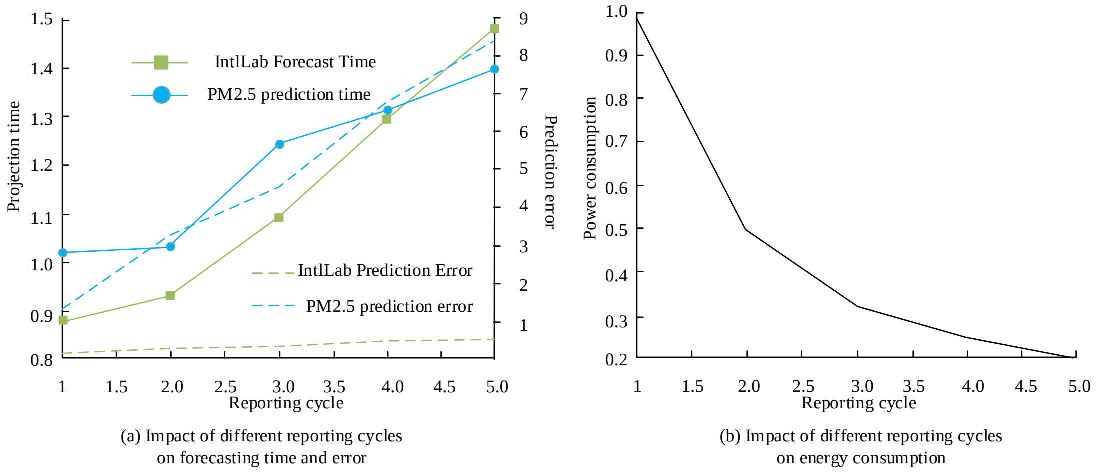


Fig. 9. Simulation results of the prediction algorithm.

Fig. 9 presents the comparative results of energy consumption, prediction error, and prediction time of the proposed prediction algorithm at different reporting intervals on the IntelLab and PM2.5 datasets. As seen in Fig. 9 (b), as the reporting interval increases, the energy consumption of nodes decreases, reaching 0.2 at a 5-cycle interval. However, the extension of the reporting interval means an increase in the prediction step, leading to a decrease in prediction accuracy and an increase in the uncertainty of search results. As seen in Fig. 9 (a), at a 5-cycle interval, the prediction error on the IntelLab dataset is around 0.5, while on the PM2.5 dataset, the error is 7.8. Additionally, a longer step length requires more computational resources and time, necessitating a balance between energy consumption, prediction accuracy, and computational costs. The prediction error on the IntelLab dataset is smaller than that on the PM2.5 dataset, as the former exhibits stronger data regularity with a smoother and more predictable variation. In contrast, the PM2.5 dataset has significant data variations, making predictions more challenging and resulting in higher errors. The model designed by the research has more predictive advantages on the data with strong regularity, is more applicable to this type of data. The setting of the reporting interval effectively influences the energy consumption, computational burden, and search performance of sensors.

V. Results and Discussion

IoT is a network system that realizes information interaction and interaction between devices by connecting various sensing devices to the Internet. As the expansion of the Internet in the physical world, IoT is widely used in smart home, intelligent transportation, health care and industrial and agricultural Internet of Things and it has become an important helper for people's production and life as well as an important technological means for the development of intelligence. Various industries at home and abroad have carried out extensive research and analysis on IoT, mainly focusing on exploring the network architecture and protocol design of IoT, data security and privacy protection, as well as big data analysis and intelligent decision-making. At the same time, a series of IoT-related industry standards and specifications have been formulated. Overall, IoT, as an important part of the future information society, has been highly valued by all walks of life, and its application and research fields are still developing and expanding.

Under the environment of IoT, the amount of data produced by numerous devices and sensors in real time is very large, and the Internet search engine is unable to efficiently process and index large-scale data, and the search capability of the existing Internet search engine does not satisfy the search demand of IoT. And the Internet search engine is usually based on batch indexing and offline processing, which is unable to obtain and process a large amount of real-time data produced by IoT devices in real time. At the same time, the data collected by IoT devices are usually structured or semi-structured data, and the existing search engines are difficult to deal with the semantic relationship between the data and the user, resulting in low accuracy and precision of the search results. In this regard, the research firstly adopts data compression technology to reduce the amount of sensor data to improve the storage and transmission efficiency, and then analyzes and predicts the future state of entity nodes using historical state information to improve the search accuracy and efficiency. The research-designed method reduces the compression rate by 6.14%, reduces the error by 5.95%, increases the compression time by 13.12%, and reduces the energy consumption by up to 82% on different public datasets. Moreover, the method can accurately predict the node state changes to satisfy the user's search needs.

Based on the work progress achieved in the research, future research can continue to focus on data compression and indexing technology, introduce intelligent optimization search algorithms to further improve the efficiency and accuracy of search algorithms, or carry out the exploration of distributed search engine technology, combining the technology of multiple fields to further achieve efficient, accurate, secure and personalized search services.

VI. Conclusion

The study addresses the insufficient real-time search capabilities of existing Internet search engines in the IoT environment. It investigates the architecture of an IoT search system and proposes a data dynamic prediction algorithm tailored for searching IoT entity information. The research begins by employing data compression techniques to reduce sensor data volume, thereby enhancing storage and transmission efficiency. Subsequently, it utilizes historical state information analysis to predict the future states of entity nodes, aiming to improve search accuracy and efficiency. Performance test results indicate that the segmented compression algorithm designed in the study enhances compression accuracy and rates. As compression accuracy increases, errors correspondingly rise, aligning with compression logic. After testing, the research selects compression accuracies for various parameters, such as temperature (0.5), humidity (1.0), illumination (1.0), voltage (0.2), PM2.5 (3.0), humidity (0.7), atmospheric pressure (3.0), and wind speed (0.5). The improved algorithm reduces compression rates by 6.14%, decreases errors by 5.95%, but increases compression time by 13.12%. Application of the data compression module results in energy consumption reductions of approximately 82% and 50%, respectively. By setting specific reporting intervals to reduce frequent data reporting by terminal nodes, energy consumption can be effectively lowered. The designed prediction algorithm demonstrates that, with an increasing reporting interval, node energy consumption decreases, reaching 0.2 at a 5-cycle interval. At this point, the prediction error is around 0.5 for the IntelLab dataset and 7.8 for the PM2.5 dataset. Experimental results show that the optimized data dynamic prediction algorithm effectively reduces node data transmission, lowers node energy consumption, and accurately predicts node state changes, meeting user search requirements. However, the study has some limitations. The testing scale for compression and prediction algorithms is relatively small, lacking large-scale practical implementation, and the robustness is insufficient. Subsequent research could consider increasing the test data for experimentation.

References

[1] Aryavalli S N G, Kumar G H. Futuristic Vigilance: Empowering Chipko Movement with Cyber-Savvy IoT to Safeguard Forests. Archives of Advanced Engineering Science, 2023, 1(8): 1-16.

[2] Deng, Lianbing, Li, Daming, Cai, Zhiming, Hong, Lin. Retraction Note: Smart IoT information transmission and security optimization model based on chaotic neural computing (Retraction of Vol 32, Pg 16491, 2019). Neural computing & applications, 2023, 35(5):4197-4197.

[3] N. Sivasankari, S. Kamalakkannan. Detection and prevention of man-in-the-middle attack in iot network using regression modeling. Advances in engineering software, 2022, 169(10):3126-3133.

[4] Sreedharan, Panchikattil Susheelkumar, Pete, Dnyandeo Jageshwar. A fuzzy multicriteria decision-making-based CH selection and hybrid routing protocol for WSN. International journal of communication systems, 2020, 33(15):36-58.

[5] Pattnaik, PK Sahu. Assimilation of fuzzy clustering approach and EHO-Greedy algorithm for efficient routing in WSN. International journal of communication systems, 2020, 33(8):547-567.

[6] Javaheri, Danial, Lalbakhsh, Pooia, Gorgin, Saeid, Lee, Jeong-A, Masdari, Mohammad. A new energy-efficient and temperature-aware routing protocol based on fuzzy logic for multi-WBANs. Ad hoc networks, 2023, 139(Feb.):3-21.

[7] Turner C, Okorie O., Emmanouilidis C, Oyekan J. Circular production and maintenance of automotive parts: An Internet of Things (IoT) data framework and practice review. Computers in Industry, 2022, 136(10):93-97.

[8] Zhang, Yitian, Yang, Kun, Fan, Xinyu. Joint time-slot and power allocation algorithm for data and energy integrated networks supporting internet of things (IoT). International journal of communication systems, 2021, 34(8):4769-4786.

[9] Mabodi Kobra, Yusefi Mehdi, Zandiyan Shahram, Irankhah Leili, Fotohi Reza. Multi-level trust-based intelligence schema for securing of internet of things (IoT) against security threats using cryptographic authentication. Journal of supercomputing, 2020, 76(9):7081-7106.

[10] Huang Z., Li R., Chen Z. Integration of data-driven models for dynamic prediction of the SAGD production performance with field data. Fuel: A journal of fuel science, 2023, 332(Jan.15 Pt.2):171.1-186.

[11] Xiao, Xiao, Jin, Zhiling, Wang, Shuo, Xu, Jing, Peng, Ziyan, Wang, Rui, Shao, Wei, Hui, Yilong. A dual-path dynamic directed graph convolutional network for air quality prediction. Science of the Total Environment, 2022, 827(15):98-114.

[12] Khan, Md Arif, Pierre, John W., Wold, Josh, I, Trudnowski, Daniel J., Donnelly, Matthew K. Impacts of swinging door lossy compression of synchrophasor data. International journal of electrical power and energy systems, 2020, 123(Dec.):182-193.

[13] Panigrahi, Sweta, Raju, U. S. N. Pedestrian Detection Based on Hand-crafted Features and Multi-layer Feature Fused-ResNet Model. International Journal of Artificial Intelligence Tools: Architectures, Languages, Algorithms, 2021, 30(5):21-44.

[14] Li, Weiye, Wu, Zhenyu. A methodology for dam parameter identification combining machine learning, multi-objective optimization and multiple decision criteria. Applied Soft Computing, 2022, 128(10):76-94.

[15] Yu, Lang, Ma, Xin, Li, Shengjie. A fast conjugate functional gain sequential minimal optimization training algorithm for LS-SVM model. Neural computing & applications, 2023, 35(8):6095-6113.