# Incremental Hierarchical Clustering driven Automatic Annotations for Unifying IoT Streaming Data

Sivadi Balakrishna[1]*, M.Thirumaran[1], Vijender Kumar Solanki[2], Edward Rolando Núñez-Valdez[3]

[1] Department of Computer Science and Engineering, Pondicherry Engineering College, Pondicherry (India)
[2] Department of Computer Science and Engineering, CMR Institute of Technology, Hyderabad, TS (India)
[3] Department of Computer Science, University of Oviedo (Spain)

Received 27 September 2019 | Accepted 23 January 2020| Published 21 March 2020

## Abstract

In the Internet of Things (IoT), Cyber-Physical Systems (CPS), and sensor technologies huge and variety of streaming sensor data is generated. The unification of streaming sensor data is a challenging problem. Moreover, the huge amount of raw data has implied the insufficiency of manual and semi-automatic annotation and leads to an increase of the research of automatic semantic annotation. However, many of the existing semantic annotation mechanisms require many joint conditions that could generate redundant processing of transitional results for annotating the sensor data using SPARQL queries. In this paper, we present an Incremental Clustering Driven Automatic Annotation for IoT Streaming Data (IHC-AA-IoTSD) using SPARQL to improve the annotation efficiency. The processes and corresponding algorithms of the incremental hierarchical clustering driven automatic annotation mechanism are presented in detail, including data classification, incremental hierarchical clustering, querying the extracted data, semantic data annotation, and semantic data integration. The IHC-AA-IoTSD has been implemented and experimented on three healthcare datasets and compared with leading approaches namely- Agent-based Text Labelling and Automatic Selection (ATLAS), Fuzzy-based Automatic Semantic Annotation Method (FBASAM), and an Ontology-based Semantic Annotation Approach (OBSAA), yielding encouraging results with Accuracy of 86.67%, Precision of 87.36%, Recall of 85.48%, and F-score of 85.92% at 100k triple data.

## Keywords

IoT Sensor Data, Semantics, Automatic Annotation, Incremental Hierarchical Clustering, Healthcare, Agent, SPARQL.

DOI: 10.9781/ijimai.2020.03.001

## I. Introduction

THE semantic technologies address the problem of various heterogeneous devices, communication protocols, and data formats of the generated data in the Internet of Things. Annotation of IoT sensor data is the substance of IoT semantics [1]. The future generation of IoT not only deals with the physical sensor devices but also the meanings they carry with virtual representation of smart data. On an average, every day around 3.2 quintillion bytes of data are generated on the Internet. The CISCO predictions state that more than 60 billion devices will be connected to the internet by 2025, as a result zetta bytes of sensor data will be generated continuously and exponentially. The IoT sensors generated raw data is stored in the data repositories and it supports to heterogeneous smart city applications. Therefore, applying the raw data into applications may result in structural data with pre-notified format, date, source, affiliation, unit, and encryption. The next level of data is perception data that contains the multi abstraction from low-level to high-level applications to perform actionable and predictive data for the final evaluation. For understanding the perception data more concisely, the structural information is needed. Without structural information, the data may mislead to false results and may fail to integrate the real-time application data [2]. The perception data is extracted from the structured data that is more compressive and occupies less space than the raw data. Machine Learning (ML) clustering techniques are used for performing analysis on the perception data and automatic generation of semantic annotations. Moreover, in IoT, the real-time streaming data plays a major role to perform cluster analysis. The streaming data is flowing continuously as data stream from the IoT device to the peer network. The stream processing has been effectively analyzes the cluster data, improve the cluster efficiency, and able to make quicker decisions on clustered data [3]-[5].

The hierarchical clustering techniques are used for representing logical, temporal, and spatial relations on the IoT streaming data. The most important aspect of clustering IoT streaming data is its dynamic and heterogeneous nature. Therefore, a novel clustering mechanism is needed to represent the hierarchical relationships-based annotations for the IoT streaming data [6]. In this paper, incremental hierarchical clustering is deployed for unifying the streaming data in a hierarchical manner. SPARQL queries are used for extracting semantic annotations between the hierarchical clustered data. The agents will receive the raw data streams as input data from the IoT sensor devices and then perform the classification between the data streams for generating the RDF data patterns for the hierarchical clustering. The RDF data patterns are combined with the pre-notified metadata of the IoT sensors for the incremental hierarchical clustering process. At last, the hierarchical streaming data is annotated with the automatic semantic annotations using SPARQL queries.

* Corresponding author.
E-mail address: balakrishna.sivadi@pec.edu

Semantic annotation has mainly taken from the field of text annotation. It provides machine-readable descriptions along with labels for URIs. Dealing with IoT semantic data is a difficult and challenging task for researchers and developers with technical issues. To solve this problem on providing the manual annotation and semi-automatic annotation, one approach for providing a semantic annotation to IoT semantic data is proposed [7]-[8]. Using manual annotation and semi-automatic annotation cannot be applicable if the IoT sensor data is huge in volume. It consumes more time to annotate the huge data and unable to capture the IoT devices generated data [9]. Therefore, a new and innovative automatic semantic annotation with more efficient mechanisms are needed.

The main contributions of this work are listed as follows: Firstly, build an architectural model using hierarchical clustering driven automatic annotation for unifying IoT streaming data. Thereafter, add semantic annotations using SPARQL queries. Then extract and visualize the streaming data using the proposed IHC-AA-IoTSD mechanism and SPARQL queries. Afterwards, find the performance evaluation of the proposed model. Finally, comparison has been made of the proposed architectural model with existing approaches.

The remainder of this paper is described as follows: section II discuss background of the related work and the state of the art schemes. In section III, the authors discuss the proposed mechanism Incremental Hierarchical Clustering based Automatic Annotation for IoT Streaming data (IHC-AA-IoTSD). Experimental Methodology and Evaluation are described in section IV and section V respectively. Finally, section VI concludes this work along with the future scope.

## II. Background and Related Work

In this section, the related work of semantic annotations in IoT platforms for unifying streaming data in efficient way is discussed. Majority of the researchers has put their efforts on how to deal with big volume and variety of data generated by IoT devices. As a result, ontologies and standards, mapping technologies and exchange systems, semantic annotations, data integration, interoperability, scalability, cluster efficiency and energy-efficient issues are identified. In semantic annotations, manual and semi-automatic annotations are time consuming and perform the annotation process with labels and manually. In addition, these all are dealing with web documents, text documents, and sensor networks. While thinking of Cyber-Physical Systems (CPS) and IoT dynamic data, it generates the big volume of data, therefore, it requires automatic annotation for handling large dynamic data.

Annotation is the process of adding additional information to the existing data, which is enriched with labels, keywords, things, etc. Semantic annotation is the term of enriching data with meanings and descriptions. Annotation plays a major role by providing semantics between humans and machines. These are categorized as three ways-Manual annotation, Semi-automatic annotation, and Automatic annotation. In manual annotation, the data is annotated manually. Here keywords are used for annotating the additional information with existing data. Humans with their self-imagination annotate the keywords. Therefore, it yields the highest accuracy, but it consumes more time to complete the entire triple data. In semi-automatic type of annotation, some part is carried out with keywords and the rest of the part is finished with trained pre-defined set automatically [21]. Two steps complete this process. In the first step, the annotator can annotate the data with keywords. In the second step, the semantic annotation tools are used to toggle the data. Both accuracy and efficiency are improved in this type of annotation system. Automatic annotation is the advanced and recently used annotation system by developers and researchers. In this, the whole process is measured by the annotation system. Annotation tools like Gruff (https://franz.com/agraph/gruff/), Jena (http://jena.apache.org/), and Protégé (https://protege.stanford.edu/) are used for this approach. Based on the instructions given by a user, the annotation tool will place corresponding predicates among the subject and object. At last, a meaningful label and property are assigned to it.

The existing research work on semantic annotation, majority of researcher's intention have been attentive on the semantic based Web documents, and a few researches pay attention to the IoT streaming data based automatic semantic annotation. As shown in Table I, the authors has been associated the former semantic annotation methods in seven aspects, such as "Automatic Annotation", "Semi-Automatic Annotation", "Manual Annotation", "Training Data Set", "Application Specific Domain", "Data Type based on" and "Model/Framework/Technology used". In Table I, the authors has been deliberate based seven aspect and it indicates the following:

- Supreme of the annotation methods focus on the Internet field and are applied for Web documents.
- The existing research of semantics for Web documents primarily pay attention towards Ontology based annotation methods.
- Majority of the existing works on semantic annotation methods in the IoT data are manual. Furthermore, they primarily focus on architectural models and deployable frameworks.

Nowadays, the methods compared in Table I are the most powerful and popular mechanisms to achieve semantic data integration in IoT platforms. The existing data models are updates with semantic annotations on providing semantic labels to become model elements. Kolozali et al. [23] proposed SensorSAX and SAX (Symbolic Aggregate Approximation) methods for adaptive and non-adaptive window size segmentation of data streams real-time processing. Their algorithms are efficient in improving data aggregation in streaming data. However, these are unfair while annotating the IoT dynamic data. Mazayev et al. [24] proposed a CoRE framework for data integration and profiling of objects, as a result, it facilitates semantic data annotation, validation of results, and reasoning of annotated data. This framework adopted the RESTful resources for validating the user profiling of objects with the COAP server. However, the proposed framework is limited for validating and annotating IoT dynamic data efficiently. Mayer et al. [25] developed an Open Semantic Framework (OSF) for industrial IoT applications to make the web of things into semantic web of things. This framework is widely designed to enable the industrial things with semantics to the IoT domains. However, the OSF is not implemented under consideration of various industrial applications.

Shi et al. [26] concentrated on data semantization in IoT applications. They reviewed and overviewed all architectural elements and applications supported for IoT domain. In addition, they surveyed on how to add semantics to the IoT dynamic data, discussed on current research issues and challenges faced by semantic scholars. However, they limited to perform analysis on IoT data integration techniques. Zamil et al. [27] have proposed automatic data annotation techniques for smart home environments by adopting temporal relations. In addition, they incorporated HMM and Random Field models for integrating temporal and spatial relations enhanced by detection accuracy rate. The produced results are moderate and there is a space for enhancement with other incremental clustering techniques. Moutinho et al. [28] have extended the semantic annotations for integrating XML-messages using generating translators under the domain of arrowhead framework. These annotations are not automatic and only domain specific. Therefore, it consumes more time and space for annotating the IoT dynamic data.

An exhaustive and optimistic survey has been conducted under the literature survey. Nevertheless, these all do not light the prerequisites

TABLE I. Comparison of Semantic Annotation Methods

| Approaches/ Methods | Automatic Annotation (Yes/No) | Manual Annotation (Yes/No) | Semi-Automatic Annotation (Yes/No) | Training data set | Application Specific Domain | Data Type based on | Model/Framework Technology used |
|---|---|---|---|---|---|---|---|
| **SRSM and MTCRF [9]** | No | Yes | Yes | No | Internet | Web documents | Rule, CRFs |
| **Chen et al., SSMIMCR [10]** | No | Yes | Yes | No | Internet | Web documents | Conceptual relationships |
| **De Maio et al., FBASAM [11]** | Yes | Yes | Yes | Yes | Internet | Web documents | Relational concept analysis |
| **Barnaghi et al., SM2SS [12]** | No | Yes | Yes | Yes | IoT | Sensor networks | Sensor streams model |
| **Kolozali et al., KBA4IoTDS [13]** | No | Yes | Yes | Yes | IoT | IoT data streams | IoT data model |
| **Wei and Barnaghi et al., SAM4SD [14]** | No | Yes | Yes | No | IoT sensor network | Sensor networks | Sensors streams model |
| **Chenyi et al., SOESAF [15]** | No | Yes | Yes | No | IoT | IoT entity information | Entity semantic annotation framework |
| **Bing, et al., SAM4IoTD [16]** | No | Yes | Yes | No | IoT | Documents | Rule |
| **Ming et al., SAM4WSDL [17]** | No | Yes | Yes | Yes | IoT | WSDL documents | Rule, Machine learning |
| **Charton et al., ASAM4NE [18]** | Yes | Yes | Yes | No | Internet | Web documents | Semantic similarity, linked data |
| **Diallo et al., OBSAA [19]** | Yes | Yes | Yes | Yes | Biomedicine | Biomedical Texts | NLP, TF-IDF |
| **Ahmed E. Khaled and Sumi Helal, ATLAS[22]** | Yes | Yes | Yes | No | IoT | Text Labelling | Topic, REST |
| **IHC-AA-IoTSD (Proposed)** | Yes | Yes | Yes | Yes | IoT | Streaming sensor data | Hierarchical clustering, automatic annotations, SPARQL Queries |

of the semantic scholars and users for adding automatic semantic annotations in IoT streaming data. Therefore, in this paper, we present IHC-AA-IoTSD mechanism using SPARQL queries to improve the clustering based annotating process in IoT sensor streaming data. Through a unification of machine learning and semantic technologies, the proposed approach gives better results in terms of efficiency, reliability, scalability and security compared to the state of the art schemes.

## III. Proposed Mechanism

In this proposed research work, to achieve semantic annotations among data samples, a Resource Description Framework (RDF) is used to annotate the data objects in meaningful way. The authors have analysed an incremental hierarchical clustering driven automatic annotation architectural model based on IoT for unifying the streaming data. For this reason, in this paper, a new and novel IHC-AA-IoTSD mechanism is proposed for annotating the streaming data semantically. The Fig. 1 shows an overview of the simplified architectural model of the proposed work. At first, the data generated from IoT sensors are collected from IoT sensor data world. On identification of the sensor data, then the agents will classify and analyze the data. The IoT streaming data generated from the data repository section; firstly, to interpret the objects in the streaming data, the RDF framework is used. Secondly, to abstract the data from the triple store, SPARQL queries is required. The SPARQL Query Engine mainly consists of three subcomponents. Those are Query Parser (QP), Query Optimizer (QO), and Query Processor (QP). The Query Parser (QP) is used for generating the triple patterns in a sequential manner. With the use of the Query Optimizer (QO), the SPARQL queries are optimized and processed. This task is accomplished before it goes to the next component called Query Processor (QP). The SPARQL Query Engine depicts the overall picture and model of the proposed approach. Each component workflow descriptions discussed as follows:

### *A. Query Parser*

This is the first subcomponent of the Query Execution Engine. This subcomponent finds the input healthcare related SPARQL queries from users, abstracts subsequent resources for the consequent subcomponent named as Query Optimizer (QO) and produces a node list for the Query Processor (QP). In this work, we used only basic SPARQL queries with simple SELECT and WHERE clauses. The proposed approach also supports other clauses, such as ORDERBY, GROUP BY, and FILTER.

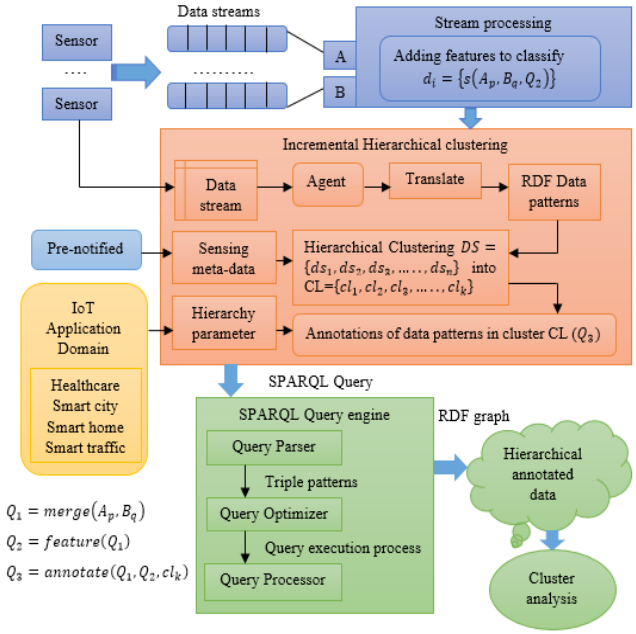


Fig. 1 Architectural model of the proposed mechanism.

## B. Query Optimizer

This is the second subcomponent and generates a Query Execution Plan (QEP) for the SPARQL query. The processing of queries is optimized by assessing the input query patterns in a meaningful way. The query triple patterns are arranged in a hierarchical manner for finding matching value result of a triple pattern function for the subsequent triple pattern in the query execution plan.

## C. Query Processor

In query processor subcomponent, the matching value results are found, verified with triple patterns and finally combined for answering the full query result. The validity of the triple patterns and input queries are arranged in a hierarchical and topological order. Then the intermediate mismatched patterns are reduced. Table II shows all the symbols or notations used in this paper.

TABLE II. SYMBOLS AND ITS DESCRIPTIONS

| Symbol | Description | Symbol | Description |
|---|---|---|---|
| sN | starting Node | cML | common Matching List |
| Tp | Triple pattern | MV | Matching Value |
| nextN | next Node | nextNML | next Node Matching List |
| QEP | Query Execution Plan | nextcN | next common Node |
| TL | Triple List | tmpP | temporary Plan |
| Ts | Triple store | subP | sub Plan |
| TpL | Triple pattern List | nextT | next Triple |
| GenTriple() | Generate Triple | cN | common Node |

The Algorithm 1 is used to translate the given label of information into a RDF label using *TranslateLabel()* function. The input is taken as *n* number of triples and correspondingly specify each type of label and the processing triple time *t* is measured. The annotated <*label*> of RDF data is the output. Firstly, it collects the various types of data from IoT devices, to store the triples data starting from 1 to n as decision iterator. If the condition $p\,(t, x_c) \leq 1$ is satisfied then extract every row and label whichever is matched. At last, the list L has to be returned.

**Algorithm 1**: *TranslateLabel ()* translates the given label into RDF label

Input: Number of triples denoted as n; type of each data item (type); t is the processing time.

Output: List out the annotated *(<label>)* from RDF.

1: First collect the various types of data generated from IoT sensors
2: **for** i: =1 to n // i is the decision iterator
3: **if** $p\,(t, x_c) \leq 1$ as per the Eq. 3
4: then extract every row and label
5: add the matched label
6: close the each completed label
7: List L: = add element
8: **end if**
9: **end for**
10. Finally, return the list L.

The Algorithm 2 is used to generate the given RDF label of information into triples using *GenTriple()* function. The input is taken as n number of triples value and correspondingly specify each type of label and subsequent time t is measured. The annotated <label> of RDF data is the output. Firstly, it collects the various types of data from IoT devices, to store the triples data starting from 1 to n as decision iterator. If the condition *label[i].isElement()* = 1 is satisfied then it extracts each label and annotates it as a triple, then it allocates the unique id for the namely added resource whichever is matched. Finally, the list L with RDF triples is returned.

**Algorithm 2**: For the generation of triples *GenTriple()* from a given RDF label

Input: Number of triples named as n; value measured; type of each data item (type); *t* is the measured time.

Output: List out the annotated *(<label>)* from RDF.

1. First, expand each label
2. **for** i: = 1 to n do // i is the decision iterator
3. **if** *label[i].isElement() = 1* // return 1 when the current label is matched element
4. extract every label and annotate it as triple
5. then allocate the unique id for namely added resource
6. List T: = add triples
7. close the completed triple list
8. **end if**
9. **end for**
10. return list L

**Algorithm 3**: IoT sensor data into annotated RDF data transformation

Input: Dataset to annotate; type of each data item (type)

Output: The annotated *(<label>)* is transformed into a reduced triple format from source data.

1: First, collect the various types of data generated from IoT sensor S
2: Repeat this collection process up to the end of the last triple
3: Then annotate the *List L: GenTriple(TranslateLabel())*
4: There after extract every label and annotate it as triple
5: Allocate the unique id for newly added resource
6: End process loop
7: Return list L

In Algorithm 3, the IoT sensor data into annotated RDF data transformation is shown. The input is taken as a dataset to annotate and specifies each data item type. The annotated *(<label>)* is transformed into a reduced triple format from source data. Firstly, it collects various type of sensor data. It repeats this process until the last triple item is matched. Then it annotates the *List L: GenTriple (TranslateLabel()).* Thereafter it extracts every label and annotates it as a triple. It allocates the unique id for the newly added resource.

### *D. Incremental Hierarchical Clustering driven Automatic Annotation Process*

The agents will play a key role to place the classification of data in time basis by using the matching mechanism for grouping each instance resources occurrence.

The matching objects are denoted as m of the RDF data and current capture objects as C. the matching m $\epsilon$ RDF instances as shown in Eq. 3.1 and current capture C as shown in Eq. 3.2, at $t_n \epsilon$ time interval. $X_m$ and $X_{cc}$ form the instances ($X_{cc}^{t_1}$, $X_{cc}^{t_2}$, $X_{cc}^{t_3}$, …… , $X_{cc}^{t_n}$) with the corresponding time interval range $t_1 \leq t_2 \leq t_3$ .....$\leq t_n$, for each individual i.

$$X_m[i]\epsilon[min_i^m, max_i^m] \tag{3.1}$$

$$X_c^j[i]\epsilon[min_i^c, max_i^c] \tag{3.2}$$

Here j starts from 1 to n.

For pattern recognition of data, let us take a resource r that should be any category of data d. The scoring function $S^d$ is used to calculate the matching pattern data d at a particular time T. The individual match value is x at time period t < T and is defined using Eq. (3.3).

$$Pr(T, x_p) = \begin{cases} Add\ data & whenever, T > t_r \text{ or } S(x^T) < 1 \\ Reject\ data & whenever, S(x_c^T) \approx S_r(x_m) \\ x & otherwise \end{cases} \tag{3.3}$$

In order to generate the hierarchical clustering driven tree of the IoT streaming data, the problem is formulated as follows: the input of the sensor raw data is classified with agents and represented as data streams $DS = \{ds_1, ds_2, ds_3, ....., ds_n\}$ in the D dimensional space, the pre-notified meta data as the k dimensions $\{x_1, …., x_k\}$, the pre-clustered streaming data values as $\{x_{k+1}, …., x_l\}$, and to measure the cluster distance among the data patterns as $dist(cl_1, cl_2)$. At starting, each classified data is assigned to its own cluster. Each data pattern in DS and the cluster $cl_i = \{ds_i\}$, $CL = \{cl_1, ....., cl_n\}$ are selected for measuring the minimum distance between data object. Then the merge operation is performed until the none of the cluster can be left blank or empty.

while $CL.size > 1$ do
  if $(cl_{min1}, cl_{min2}) = min\ dist\ (cl_i, cl_j)$ then
    for all $(cl_i, cl_j)$ in cluster CL
    Remove $cl_{min1}, cl_{min2}$ from cluster CL
    Add $\{cl_{min1}, cl_{min2}\}$ to cluster CL
  end if
end while

The $dist\ (cl_1, cl_2)$ is measured as, for example $cl_1 = \{ds_{11}, ds_{12}, ds_{13}, ....., ds_{1n}\}$ and $cl_2 = \{ds_{21}, ds_{22}, ds_{23}, ....., ds_{2n}\}$, then $dist(cl_1, cl_2) = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} dist(ds_{1i}, ds_{2j})$. Whereas $dist\ (ds_{1i}, ds_{2j})$ may be calculated using any of the mahalanobis distance, Euclidean distance, or Minkowski distance function in the D dimensional space. The same procedure is performed until the semantic annotations are extracted from the hierarchical tree by cutting into horizontally or vertically and adding the data streams in incremental manner.

The following list of steps are required to design an incremental hierarchical clustering driven automatic annotations for unifying IoT streaming data.

The input data streams $DS = \{ds_1, ds_2, ds_3, ....., ds_n\}$ are obtained from the IoT sensor data repositories in the D dimensional space.

The incremental hierarchical clustering based nearest neighbor chain is used for clustering streaming data.

It starts with any node S in the hierarchical tree, elaborates it until a RNN (Reciprocal Nearest Neighbor) pair of data samples, and then agglomerates these data samples.

Continue the same process with the hierarchical tree of the previously annotated objects using RNN.

The RNN of object p and q, where object q must satisfy the condition

$$if dist(p, q) \leq min\{dist(p, r), dist(q, r)(r \neq p, q)\}$$

Thereafter, the clustering distance $dist(p, q)$ using the Euclidean similarity distance measuring function is measured.

The establishment of the linkage or distance between clusters of the hierarchical tree is done using wards method

$\Delta(a, b) = \frac{w_a w_b}{w + w_b} \|\overrightarrow{n_a} + \overrightarrow{n_b}\|$ where $\overrightarrow{n_a}$ is the center of the cluster e and $n_e$ is the number of data samples involved in it.

Finally, the semantic representations between the clustered hierarchical trees are annotated with SPARQL queries.

In the Resource Description Framework (RDF), the data is generally warehoused as a combination of statements in triples format as {Subject Sb, Predicate Pr, Object Obj}, which is similar to an entity representation in DBMS as {entity e, property p, value v}. Subjects and predicates stored in triples are URIs when objects can be either Uniform Resource Identifiers (URIs) or literal values. SPARQL is a Simple Protocol and RDF Query Language is used for retrieving data stored in RDF repositories. Its syntax is similar to SQL; thus it contains two main clauses, e.g., SELECT and WHERE. The SELECT clause identifies the statements as triples that will appear in the query results. The WHERE clause provides the basic graph pattern to match against the data graph. We consider four disjoint sets Var (variables), Uri (URIs), Blnk (blank nodes) and Ltr (literals).

Almost every SPARQL query contains a set of triple patterns called a basic graph pattern. A basic graph pattern, BGP, is a finite set of triple patterns {tp1, tp2, tp3….,tpn}, in which each tp is a triple as shown in Eq. (3.4).

$$(Sb, Pr, Obj) \in (Var \cup Uri \cup Blnk)\ X(Var \cup Uri)\ X\ (Var \cup Uri \cup Blnk \cup Ltr) \tag{3.4}$$

The sequence of the patterns is framed with different combinations of the triples as shown in Eq. (3.5)

$$SId: [node] = Value: [resource] \cap Sub\_Pre\_Obj: \{S_1 : [p_1, p_2, ; o_1, o_2], S_2 : [p_1, p_2, ; o_1, o_2]\} \tag{3.5}$$

The Query Execution Plan (QEP) is measured based on the sequence of patterns generated by triples ($tp_1$, $tp_2$, $tp_3$, … $tp_n$) as long as the long sequence patterns are generated. Such that there is at least one common medium of sequence patterns among $tp_1$ and $tp_{i+1}$ from (Subject as S, Object as O, and Predicate as P) being selected, and it follows any one of the patterns as shown in Eq. (3.6-3.8).

$$S(tp(i)) = S(tp(i+1))\ and\ O(tp(i+1)) \tag{3.6}$$

$$P(tp(i)) = S(tp(i+1))\ and\ P(tp(i+1)) \tag{3.7}$$

$$O(tp(i)) = S(tp(i+1))\ and\ O(tp(i+1)) \tag{3.8}$$

Here S(tp), P(tp), and O(tp) are the Subject, Predicate, and Object respectively. If anyone of the triple patterns is satisfied with the required query then the query execution plan is assigned to the Query

Optimizer (QO) subcomponent.

In order to develop the query execution plan, the query in Algorithm 4 is processed and stored in the triple store (ts). The loaded query is mapped with each triple pattern to subsequent nodes. Sometimes, it refers to other triple patterns that are matched with the stored triple values i.e. (node, [adjacent_TL]). The subject, predicate, and object manner are matched by applying the SPARQL query; finally, the corresponding RDF graph is generated. In Algorithm 4, the query execution plan is shown. The motivation behind this query execution plan is engaging the ordered triple patterns to indexed RDF data and improved version of ordered triple patterns to process the queries in an efficient manner. After generating an ordered triple pattern list for the execution plan, the residual sequential triple patterns that are in the triple list are not attached to the current execution plan. The Tp is considered as the triple pattern in QEP. The nextN <= get_nextN (Tp, sN) is placed subject as first node and object as the second node and vice versa generated. The subP is the sub plan used for storing the remaining triple pattern part for QEP. The appended data triple pattern is a subset of adjacent triple list and TpL is the not visited list then create an intermediator plan for annotating current pattern objects. Such that, consider this one as the current query evaluation plan for executing queries. Finally, the next triple and triple pattern are merged with adjacent triple list, evaluated with QEP.

**Algorithm 4:** GET_ Query_Execution_PLAN (sN, Tp)

**Input:** sN - starting Node; tp- triple pattern; ts- triple store (for storing every query generated triple).

**Output: QEP**, - generates longest triple patterns in RDF format

**1:** QEP **<=** Tp // triple pattern is considered in QEP

**2**: *nextN***<=** get_nextN (Tp, sN)

*// if N is the Node and it is in subject place, nextN is its object.*

*// if N is the Node and it is in object place, nextN is its subject*

**3**: *adjacentTL***<=** getTL (Ts, nextN) *// TL- Triple List; Ts- Triple store*

**4:** *subP***<=** Ø *// load the remaining triple pattern part for QEP; subP- sub Plan*

**5: for each** data triple pattern TpL ϵ *adjacentTL* **do** *// TpL- Triple pattern List*

**6:** **if** (TpL ≠ visited node) **then**

**7:** tmpP **<=** GET_Query_PLAN (nextN, TpL)

**8**: **if** len *(subP)*< len *(tmpP)***then**

**9:** *subP* **<=***tmpP*

**10:** *nextT***<=** *TpL*

**11:** **end**

**12:** **end**

**13: end**

**14: QEP**<= *adjacentTL* \ {*nextT*, Tp} *// nextT is included in subP*

**15: QEP<=***subP*

**16: return QEP**

The following are the list of steps required to execute the query plan.

1. Firstly, go to the file menu, in that select new triple store- appropriate path name that has been given for storing work in the triple store.
2. Once the path is identified by triple store, a maximum number of estimated triples are selected. (E.g. 100000).
3. Then load the triples of any format (E.g. N-triples, RDF/XML, N-Quads).
4. Go to the Query view in View menu bar. The required query is applied for annotating the data in RDF format.
5. Then run the SPARQL query, it shows the result as ?s ?p ?o in tabular form.
6. Finally, click on the create visual graph icon, then it generates the annotated RDF graph. Make changes on the graph as per the neediness of the user.

In the Query Execution Plan (QEP), the subject, predicate, and object are placed in triple patterns format. Therefore, at any point, only the vertices and edges can be placed. The time complexity for generating the query execution plan is $O(|S|).|P|)$. Here, $|S|$ is the number of Subjects placed in the healthcare dataset, and $|P|$ is the number of Predicates placed in the healthcare dataset. Therefore, the proposed algorithm 4 and algorithm 5 take total computational time $O(|S|) + |P|)$ as the time complexity, because this work uses every Subject and Predicate or node only once.

## IV. Experimental Methodology

In this section, the proposed mechanism is described with automatic annotations for unifying the IoT streaming data. In addition, the implementation results of the proposed mechanism with SPARQL queries processing is discussed.

### A. Adding Semantic Annotations to the IoT Streaming Data

In this proposed research work, to achieve semantic annotations, the query processing mechanism and triple patterns are used. The authors have analyzed and tested on three different healthcare datasets using incremental hierarchical clustering driven automatic annotations based on IoT streaming data.

### B. Query Processing

In this section, the queries are processed and executed based on the query execution plan so Algorithm 4 is used. To perform that operation we need to observe the correct triple patterns from the triple store or find the invalid annotation results. The following common steps used in Algorithm 5 using a triple store are followed:

1. Firstly, the input cN is considered as the common node or node pattern to retrieve the subsequent triple pattern (tp) from query execution plan and generate the annotation results from triple store to the common node cN.
2. The matching common node cN is attached to the common matching list cML.
3. For each common annotated data is resulted to merge the annotator list of final matching list.
4. Then, each matching value is a subset of the final matching list and contains the annotated attributes for matching value identification.
5. The next value is placed on the basis to get the next node and add the mapping value to the triple store.
6. If any node is mapped with the next node then the mapping annotations consisting of next node, matching value, and next node matching list are added. If any node is not matched with the next node then all corresponding matching values and associated annotations are removed.
7. This entire process is repeated until the all-existing triples are reached and mapping of the common node cN exists that was taken from triple pattern tp.

The SPARQL queries are processed for annotating the matching healthcare data and its associated values. However, the queries are different triple patterns ($tp_1$, $tp_2$, ...., $tp_n$) and the matching subject of any common node is retrieved as well as its corresponding predicates.

**Algorithm 5:** GET_ PROCESS_TRIPLE_PATTERN (cN)

**Input:** cN is the common Node (node pattern).

**Data: QEP-** Query Execution Plan for processing triple patterns using SPARQL queries. Tp- Triple pattern;

Ts- Triple store (for storing every intermediate generated data). Ant-annotation

**Output:** SPARQL query and RDF triple data mapping

**1:** Tp<= QEP.getNext () // *next triple pattern is considered in QEP and is placed*

**2:** cML <=M (*cN*) // *common node match list placed*

**3: for each** ant Є getAnnotatorList (*cML*) **do**

**4:** **for each** MV Є cML.getMV (*ant*) **do**

**5:** nextN <=get_nextN (*Tp, cN*)

**6:** nextNML <=findMatches (*nextN, MV*)

**7:** **if** any node is mapping with *nextN* **then**

**8:** M. addMap (*nextN, MV, nextNML*) // *now MV is the annotator of nextN*

**9:** **else**

**10:** remove (*MV*) // *remove the matching values and associated annotators*

**11:** **end**

**12:** **end**

**13: end**

**14: if** any node is matched with result of *Tp* **then**

**15**: *nextcN*<=findNextcN (*Tp*)

**16:** PROCESS_TRIPLE_PATTERN (*ncN*)

**17: end**

```
select ? s ?  ? p ? o  where
{<https://Resource url> ? s ? p ? o
limit 100
}
```

Fig.2. SPARQL Query.

The Fig. 2 is a sample SPARQL query for annotation of triples such as subject, predicate, and object manner. The SPARQL queries are widely used for annotating the RDF data for machine-readable and semantically describable data. There is another option in SPARQL queries to extract the full dataset attribute information with a limit basis like 10k, 20k, 30k, 100k, and so on triples.

## V. Performance Evaluation

This section employs the experimental datasets used for the proposed IHC-AA-IoTSD mechanism. In addition, the performance evaluation metrics are discussed for evaluating the performance of the IHC-AA-IoTSD in detail. In the final analysis, the time complexity of the proposed algorithms are measured.

### A. Data Setup

For evaluation of the proposed mechanism IHC-AA-IoTSD, three different kinds of healthcare datasets, namely Heart diseases, Heart attack, and Diabetes are taken. These are openly available datasets from the UCI Machine learning repository**.** Table III shows the dataset details including names of datasets, the number of triples in the datasets, and downloadable resources information.

TABLE III. Dataset Details

| S.No | Dataset Name | No. of Triples | Source |
|---|---|---|---|
| **1** | Heart diseases | 212154 | https://archive.ics.uci.edu/ml/datasets/heart+Disease |
| **2** | Heart Attack | 112896 | https://www.kaggle.com/imnikhilanand/heart-attack-prediction |
| **3** | Diabetes | 142547 | https://archive.ics.uci.edu/ml/datasets/diabetes |

### B. Experimental Environment

To evaluate the performance proposed mechanism, a conventional and regular laptop was used with the configuration of Windows 10 Home 64-bit, 8 GB RAM, 1 TB HDD, 2 cores, 2.2 GHz CPU clock speed, and Intel® Core™ i7-8th Gen-8750H CPU type. The Gruff tool with Java 1.8.0 platform was used to experiment the healthcare data. The Tableau and Allegro Graph tools support to visualize the data in a good manner for users. The SPARQL query language was used for annotating the healthcare data to communicate patient and doctors in a meaningful way.

### C. Performance Metrics

To evaluate the performance of the proposed framework, the following metrics are considered for measuring the framework. These metrics are generated from the confusion matrix as shown in Table. IV.

TABLE. IV. Confusion Matrix

| | **Predicted as "YES"** | **Predicted as "NO"** |
|---|---|---|
| **Actually as "YES"** | True Positive $[Cl_{ant} \rightarrow Cl_{ant}]$ | False Negative $[Cl_{ant} \rightarrow NCl_{ant}]$ |
| **Actually as "NO"** | False Positive $[NCl_{ant} \rightarrow Cl_{ant}]$ | True Negative $[NCl_{ant} \rightarrow NCl_{ant}]$ |

- *True Positive* $Cl_{ant} \rightarrow Cl_{ant}$: This is an assessment of correctly clustered annotations considered correctly as clustered annotations.
- *True Negative* $NCl_{ant} \rightarrow NCl_{ant}$: This is an assessment of non-clustered annotations considered correctly as non-clustered annotations.

$$TPR = \frac{Cl_{ant} \rightarrow Cl_{ant}}{[Cl_{ant} \rightarrow Cl_{ant} + Cl_{ant} \rightarrow NCl_{ant}]} \quad (4.1)$$

- *False Positive* $NCl_{ant} \rightarrow Cl_{ant}$: This is an assessment of non-clustered annotations considered incorrectly as clustered annotations.
- *False Negative* $Cl_{ant} \rightarrow NCl_{ant}$: This is an assessment of clustered annotations considered incorrectly as non-clustered annotations.

### 1. True Positive Rate (TPR)

TPR states the sensitivity value and measures correctly clustered annotations from the dataset as shown Eq. (4.1). Eq. (4.2) corresponds to the true negative rate (TNR).

$$TNR = \frac{NCl_{ant} \rightarrow NCl_{ant}}{[NCl_{ant} \rightarrow NCl_{ant} + NCl_{ant} \rightarrow Cl_{ant}]} \quad (4.2)$$

### 2. False Positive Rate (FPR)

FPR measures the significance level, which scales the proportion of non-clustered annotations that are interpreted as clustered annotations in the automatic annotation process, and generated as input dataset sequence as shown Eq. (4.3).

$$FPR = \frac{NCl_{ant} \rightarrow Cl_{ant}}{[NCl_{ant} \rightarrow Cl_{ant} + NCl_{ant} \rightarrow NCl_{ant}]} \quad (4.3)$$

### 3. False Negative Rate (FNR)

FNR scales the proportion of clustered annotations that are interpreted as non-clustered annotations in the clustered data annotation process as shown Eq. (4.4).

$$FNR = \frac{Cl_{ant} \rightarrow NCl_{ant}}{[Cl_{ant} \rightarrow NCl_{ant} + Cl_{ant} \rightarrow Cl_{ant}]} \quad (4.4)$$

### 4. Accuracy

Accuracy is the first step towards performance measure where it defines the ratio between the total counts of correct clustered annotations made to a total count of clustered annotations made as shown Eq. (4.5).

$$Accuracy \frac{(Cl_{ant} \rightarrow Cl_{ant} + NCl_{ant} \rightarrow NCl_{ant})}{[Cl_{ant} \rightarrow Cl_{ant} + NCl_{ant} \rightarrow NCl_{ant} + NCl_{ant} \rightarrow Cl_{ant} + Cl_{ant} \rightarrow NCl_{ant}]} \quad (4.5)$$

### 5. Precision, Recall & F-measure

Precision discourses about the exactness of the clustered data, and the Recall voices about completeness of the data. The Precision and Recall discuss more about the detected accuracy of the data, and the accuracy should not deal much about false results. The F-measure is the mean of precision and recall. The equations depicted from (4.6) to (4.8) is Precision, Recall, and F-measure respectively.

$$Precision = \frac{Cl_{ant} \rightarrow Cl_{ant}}{[Cl_{ant} \rightarrow Cl_{ant} + NCl_{ant} \rightarrow Cl_{ant}]} \quad (4.6)$$

$$Recall = \frac{Cl_{ant} \rightarrow NCl_{ant}}{[Cl_{ant} \rightarrow Cl_{ant} + Cl_{ant} \rightarrow NCl_{ant}]} \quad (4.7)$$

$$F - measure = 2 * \left(\frac{Precision * Recall}{Precision + Recall}\right) \quad (4.8)$$

These ML metrics are used on the proposed mechanism for improving cluster efficiency and unifying the IoT streaming data.

### D. Experimental Results and Discussions

This experiment is conducted under the stimulus of three healthcare datasets namely- Heart Diseases, Heart Attack, and Diabetes by applying various triple sizes with 10k, 20k, 30k, 40k, 50k, and 100k respectively. Annotating the objects of streaming data, six SPARQL queries are used to evaluate the proposed IHC-AA-IoTSD mechanism as represented in Fig.3 to Fig.10.

```
select ? drug ? type ? value where
{<https://www.drugs.tms/drugs/resource/MeDRAconceptType> ? drug ? type ? value
limit 100
}
```

Fig. 3. SPARQL Query 1.

The SPARQL query 1 shown in Fig. 3, queries for the drug types and values annotated with hierarchical clustered data. The SPARQL query 2 shown in Fig. 4 is used to extract unique heart attack attributes and their values from heart attack dataset.

```
select ? heart attack ? value where
{<https://data.medicare.gov/resource/ygty-mm5a3> ? heart attack ? value
limit 100
}
```

Fig. 4. SPARQL Query 2.

The role of SPARQL queries is highly enrich to all attributes for annotation. In addition, the queries are effectively annotated various attributes in lower execution time.

```
select ? heart attack ? type where
{<https://data.medicare.gov/resource/ygty-mm5a3> ? heart attack ? type
limit 100
}
```

Fig. 5. SPARQL Query 3.

The SPARQL query 3 is as shown in Fig. 5 and its resultant RDF graph as shown in Fig. 7. The hierarchical tree based predicates are annotated over the various triple data objects.

```
select ? Row ID ? Member ? Year ? Number of diabetes deaths where
{<https://sensormeasurement.appspot.com/m3#diabetesdeaths>
? Row ID ? Member ? Year ? Number of diabetes deaths
limit 100
}
```

Fig. 6. SPARQL Query 4.

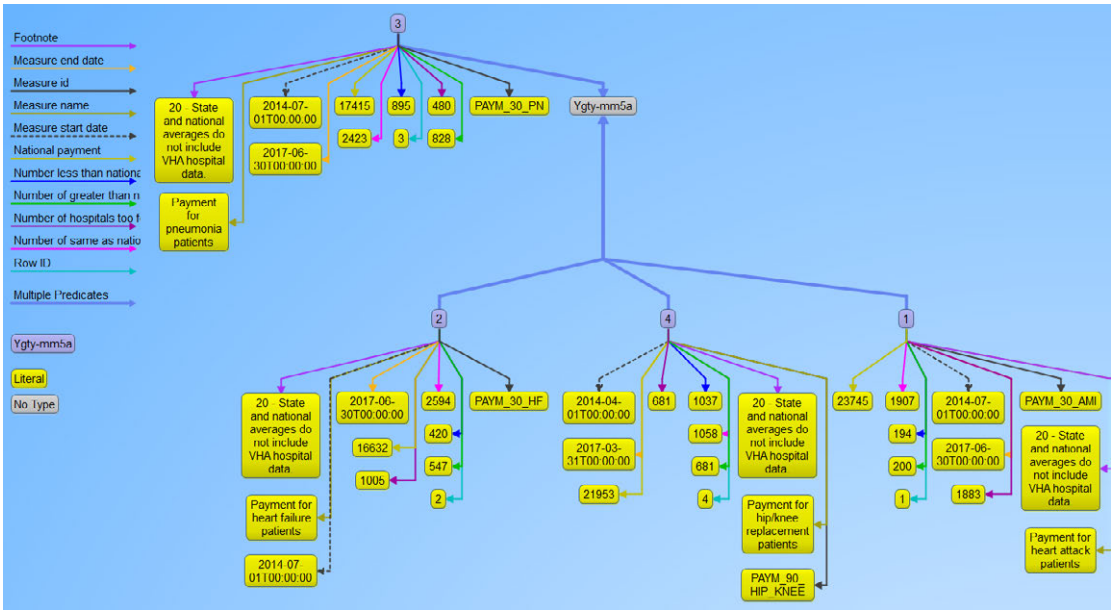


Fig. 7. Annotated heart attack diagnostic measurement values in a hierarchical tree.

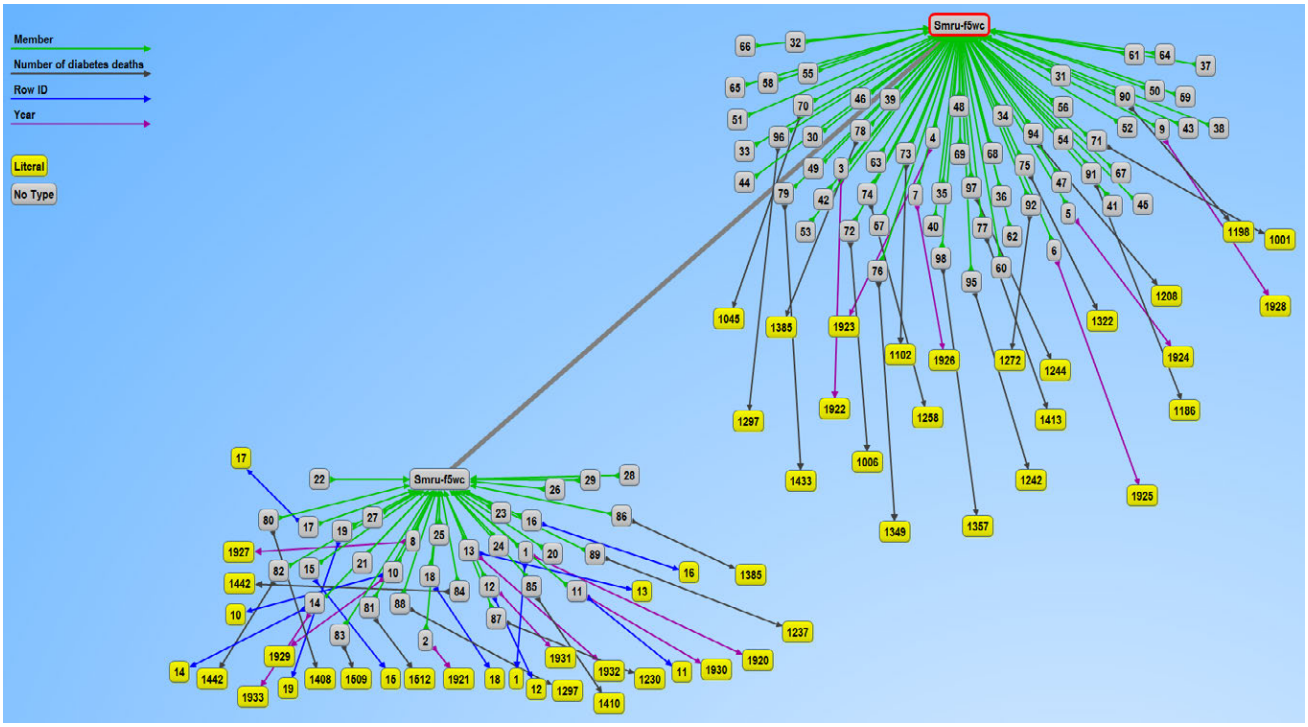


Fig. 8. Annotated diabetes diagnostic measurement values.

The SPARQL query 4 as shown in Fig. 6. The Fig. 8 shows the resulted output of query 4. Moreover, the diabetes data set contains of predicates as row id, value, and number of the deaths, etc. Using annotation process, the representation of the year wise death rates have been enriched as well as extracted.

```
select ? subject ? predicate ? object where
{<https://data.cdc.gov/resource/bi63-dtpu> ? subject ? predicate ? object
limit 100
}
```

Fig. 9. SPARQL Query 5.

The SPARQL query 5 shown in Fig. 9 has been performed on heart diseases dataset for annotating the healthcare records by means of subject, predicate, and object manner. It indicates that the annotations performed on the whole dataset with accurate annotations. The SPARQL query 6 shown in Fig. 10 is widely used for annotating the heart diseases data on value and predicate basis annotations. In this, the corresponding predicate as the number of national payments on year wise, payment for heart diseases, measure id, measure name, measure start date, measure end date, type and corresponding values are annotated. The SPARQL query 5 and query 6 are used in this paper to annotate the healthcare data by varying triple data size up to 100k triples. These results have not been presented because these annotations make the things complex and not visible to the users.

```
select ? value ? predicate where
{<https://data.maryland.gov/resource/smru-ftwc> ? value ? predicate
limit 100
}
```

Fig. 10. SPARQL query 6.

However, the results of SPARQL query 1 to query 6 clearly indicate that automatic annotations are more concisely preferable than the manual and semi-automatic annotations. Because in automatic semantic annotations, the trained and classified data are labelled using an automated annotation system. The average execution time of the various queries are measured, and it achieves the lowest compared with ATLAS [22], FBASAM [11], and OBSAA [19] approaches.

The first experimental investigation of IHC-AA-IoTSD is validated through TPR by applying various triples with respect to a stable FPR 10, 20, 30 and 40% over the benchmark mechanisms such as ATLAS, FBASAM, and OBSAA is observed in Figs. 11, 12, and 13 respectively.

Fig.11 (a-d) shows the leading TPR value on Heart Diseases dataset of proposed IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA with respect to the stable FPR of 10%, 20%, 30%, and 40% respectively. Fig. 11 (a) result proves that IHC-AA-IoTSD is capable to preserve the TPR around 0.95 at dynamically allocated triples and this TPR value infers 12% success rate than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively under 10% FPR. Fig. 11 (b) shows the dominant TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA respectively under 20% FPR and is capable to maintain its TPR value around 0.92 at dynamically allocated triples even the FPR is increased. In addition, the proposed IHC-AA-IoTSD proves a greater TPR around 13% than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively. Likewise, Fig. 11 (c) represent the TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA under 30% FPR and is capable to withstand its TPR value around 0.9 at various dynamically allocated triples and proves a greater TPR around 11% than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively. Similarly, Fig. 11 (d) represent the TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA respectively under 40% FPR. Besides, proposed IHC-AA-IoTSD is achieved a marginable TPR around 0.88 at dynamically allocated triples and proves this TPR value infers 8% higher accurate than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively.

Fig.12 (a-d) shows the dominant TPR value on Heart Attack dataset of proposed IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA with respect to the stable FPR of 10%, 20%, 30%, and 40% respectively. Fig. 12 (a) result proves that IHC-AA-IoTSD is capable to preserve the TPR around 0.92 at dynamically allocated triples and this TPR value infers 12% success rate than the benchmark

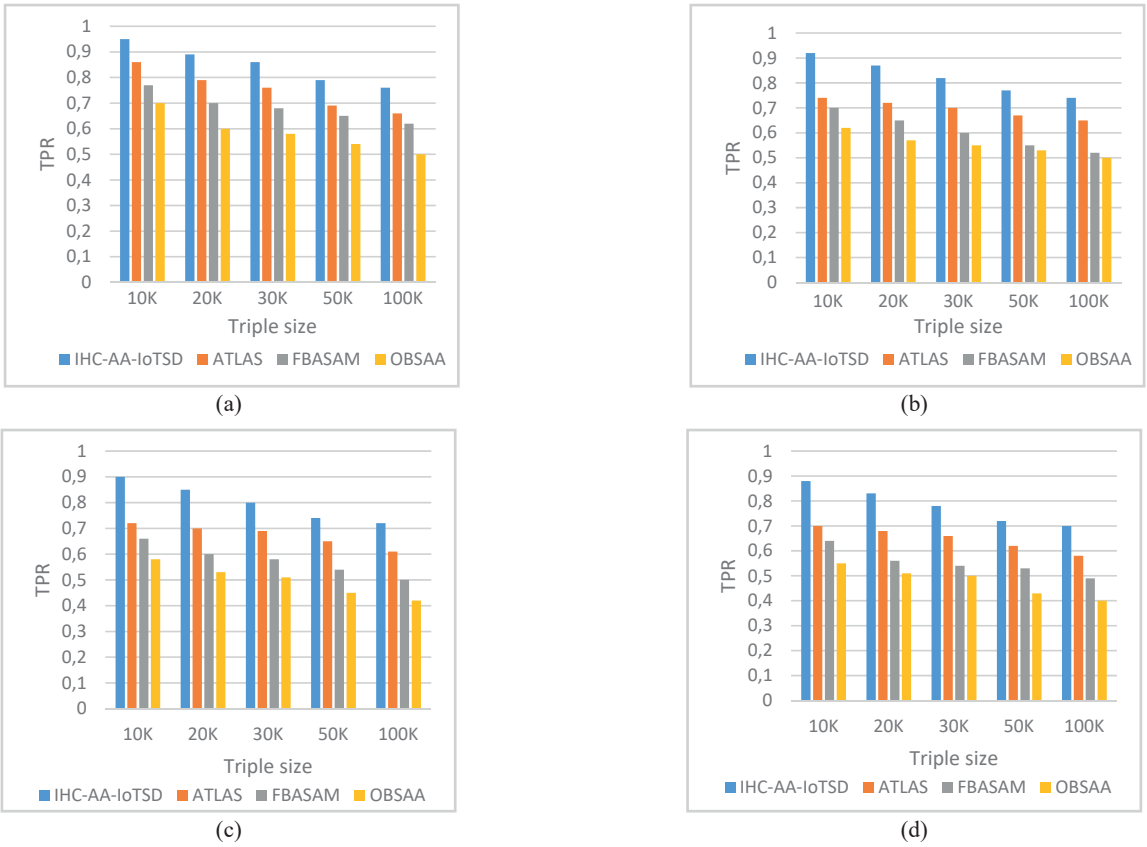


Fig. 11. True Positive Rate (TPR) on Heart Diseases dataset by varying triple size (a) false positive rate =10%, (b) false positive rate =20%, (c) false positive rate =30%, (d) false positive rate =40%.

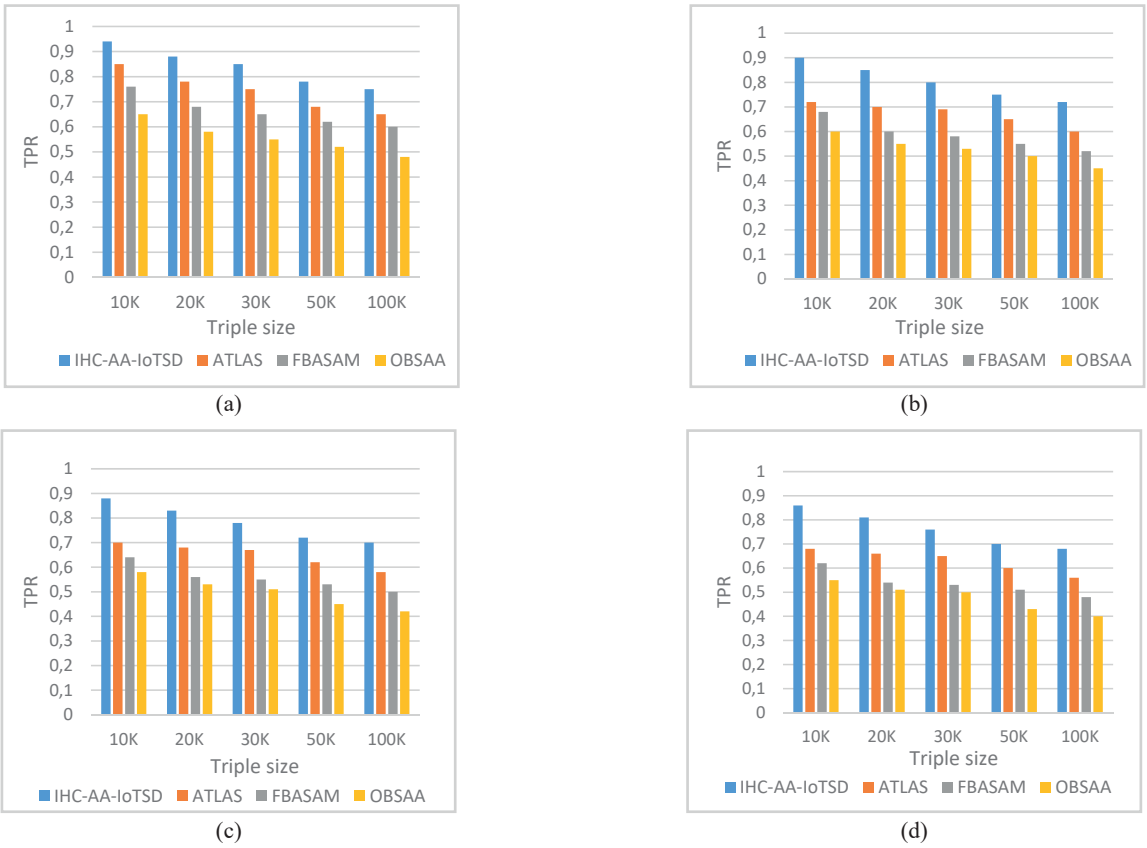


Fig. 12. True Positive Rate (TPR) on Heart Attack dataset by varying triple size (a) false positive rate =10%, (b) false positive rate =20%, (c) false positive rate =30%, (d) false positive rate =40%.

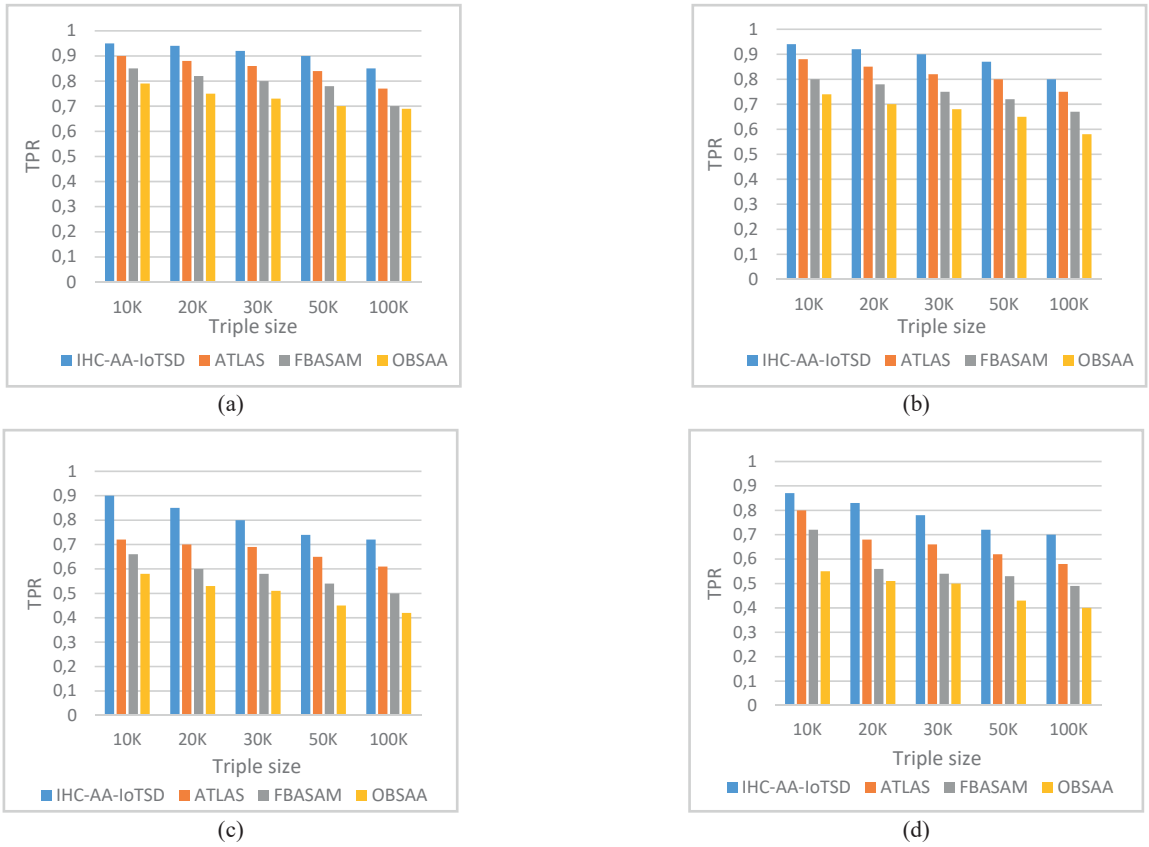


Fig. 13. True Positive Rate (TPR) on Diabetes dataset by varying triple size (a) false positive rate =10%, (b) false positive rate =20%, (c) false positive rate =30%, (d) false positive rate =40%.

mechanisms ATLAS, FBASAM, and OBSAA respectively under 10% FPR. Fig. 12 (b) shows the dominant TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA respectively under 20% FPR and is capable to maintain its TPR value around 0.9 at dynamically allocated triples even the FPR is increased. In addition, the proposed IHC-AA-IoTSD proves a greater TPR around 13% than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively. Likewise, Fig. 12 (c) represent the TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA under 30% FPR and is capable to withstand its TPR value around 0.88 at various dynamically allocated triples and proves a greater TPR around 11% than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively. Similarly, Fig. 12 (d) represent the TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA respectively under 40% FPR. Besides, proposed IHC-AA-IoTSD is achieved a marginable TPR around 0.86 at dynamically allocated triples and proves this TPR value infers 7% higher accurate than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively.

Fig.13 (a-d) shows the dominant TPR value on Heart Diseases dataset of proposed IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA with respect to the stable FPR of 10%, 20%, 30%, and 40% respectively. Fig. 13 (a) result proves that IHC-AA-IoTSD is capable to preserve the TPR around 0.94 at dynamically allocated triples and this TPR value infers 13% success rate than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively under 10% FPR. Fig. 13 (b) shows the dominant TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA respectively under 20% FPR and is capable to maintain its TPR value around 0.92 at dynamically allocated triples even the FPR is increased. In addition, the proposed IHC-AA-IoTSD proves a greater TPR around 12% than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively. Likewise, Fig. 13 (c) represent the TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA under 30% FPR and is capable to withstand its TPR value around 0.9 at various dynamically allocated triples and proves a greater TPR around 11% than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively. Similarly, Fig. 13 (d) represent the TPR value of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA respectively under 40% FPR. Besides, proposed IHC-AA-IoTSD is achieved a marginable TPR around 0.87 at dynamically allocated triples and proves this TPR value infers 9% higher accurate than the benchmark mechanisms ATLAS, FBASAM, and OBSAA respectively.

In the second experimental investigation of IHC-AA-IoTSD validated through the detection accuracy, TNR, FNR, TPR, Precision, and FPR over the benchmark mechanisms such as ATLAS, FBASAM, and OBSAA techniques respectively.

Fig. 14 (a) represents the average detection accuracy of IHC-AA-IoTSD on three healthcare datasets with various triple sizes. The results confirm that IHC-AA-IoTSD is capable to accomplish superior detection accuracy in heart dataset from the UCI data repository, and it acquired detection accuracy of 9–94% from 10k triples to 100k triples respectively. Nevertheless, ATLAS facilitates a detection accuracy of 7–90% from 10k triples to 100k triples respectively, FBASAM achieves a detection accuracy of 5–81% from 10k triples to 100k triples respectively and OBSAA ensures a detection rate of 2–75% from 10k triples to 100k triples respectively. Performing tests on heart attack dataset from the kaggle data repository, it got a detection accuracy of 11–97% from 10k triples to 100k triples respectively. Nevertheless, ATLAS facilitates a detection accuracy of 9–93% from 10k triples to 100k triples respectively, FBASAM achieves a detection accuracy of 7–89% from 10k triples to 100k triples respectively and OBSAA ensures a detection rate of 4–84% from 10k triples to 100k triples respectively. Performing tests on diabetes dataset from the UCI data repository, it

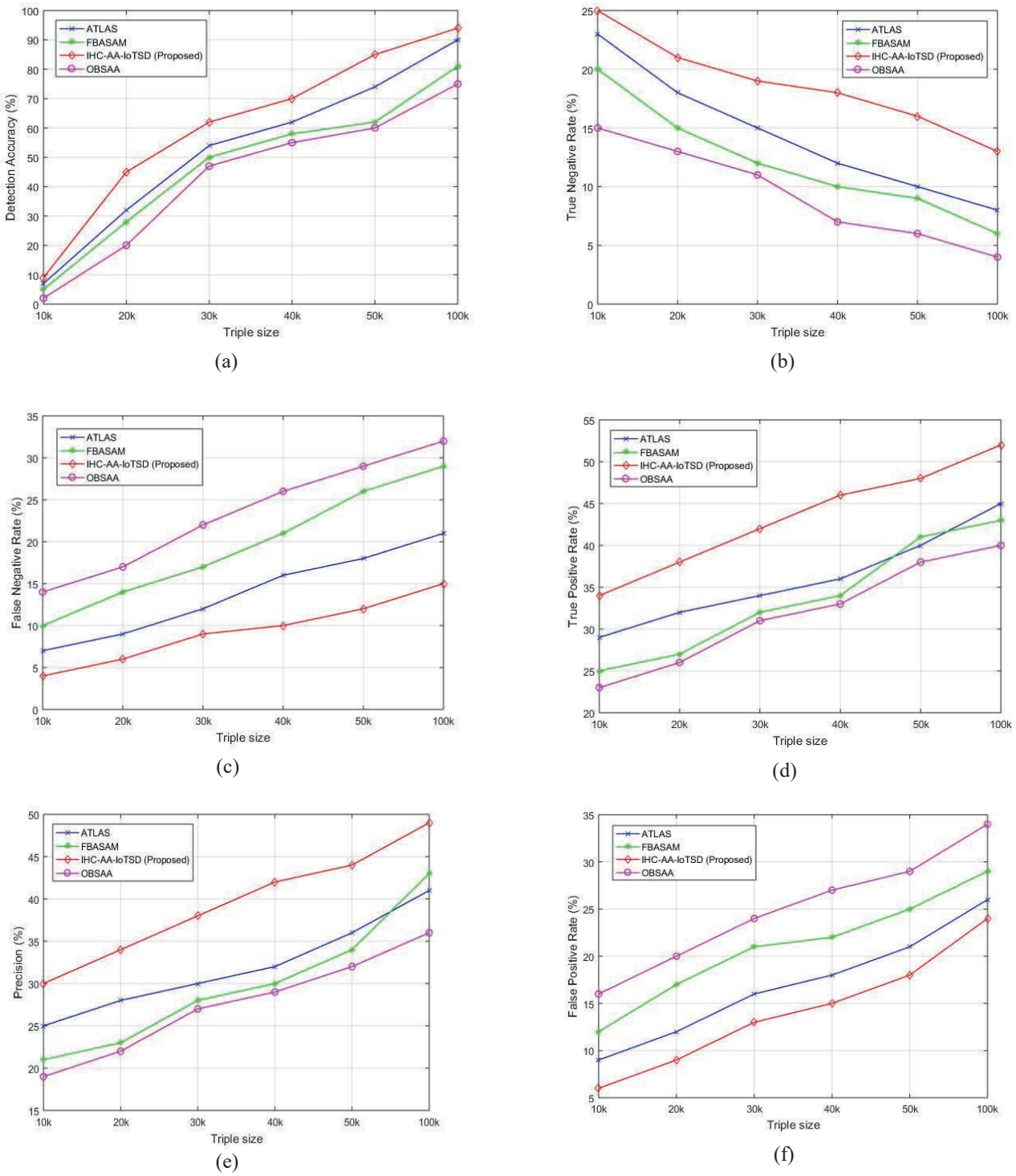


Fig. 14. (a) Detection Accuracy of IHC-AA-IoTSD under various triple sizes, (b) True Negative Rate of IHC-AA-IoTSD under various triple sizes (c) False Negative Rate of IHC-AA-IoTSD under various triple sizes (d) True Positive Rate of IHC-AA-IoTSD under various triple sizes (e) Precision Rate of IHC-AA-IoTSD under various triple sizes (f) False Positive Rate of IHC-AA-IoTSD under various triple sizes.

got a detection accuracy of 10–96% from 10k triples to 100k triples respectively. Nevertheless, ATLAS facilitates a detection accuracy of 7–90% from 10k triples to 100k triples respectively, FBASAM achieves a detection accuracy of 5–84% from 10k triples to 100k triples respectively and OBSAA ensures a detection rate of 3–80% from 10k triples to 100k triples respectively. On an average IHC-AA-IoTSD got the 4% detection accuracy increases from ATLAS mechanism at 10k triples whereas at 100k triples got the same improvement. After combining the three healthcare dataset results with increased detection accuracy, the results indicating that 2%, 4%, and 7% decrease than the ATLAS, FBASAM, and OBSAA techniques respectively. This effectiveness of IHC-AA-IoTSD by means of detection accuracy is primarily payable to the enhanced process of multi-agent based semantic annotation used for classifying and testing. This detection accuracy is also because of the agent-based automatic semantic process stimulated in the IHC-AA-IoTSD annotation mechanism.

Fig.14 (b) shows the TNR value of IHC-AA-IoTSD under varying triple data size and the result endorses that it is effective in enlightening

the TNR value by 15-23% differing to ATLAS, FBASAM, and OBSAA, which enable an improvement of 2%, 5%, and 10% from 10k triples to 100k triples. The results about the enhancement of TNR value prove that the IHC-AA-IoTSD performs better because of the patient and doctor annotating the healthcare data enabled in the detection process.

Fig.14 (c) depicts the reduced FNR of IHC-AA-IoTSD under changing triple rate and ensures that, it can minimize the FNR of about 16–30%, which is hardly 8% decrease at 10k triple size and 12% decrease at 100k triple size than ATLAS framework testing on heart diseases dataset. Testing on heart attack dataset, it can minimize the false negative rate of about 17–29%, which is nearly 8% decrease at 10k triple size and 8% decrease at 100k triple size than ATLAS framework. Similarly, by testing on diabetes dataset, it can minimize the false negative rate of about 18–29%, which is nearly 8% decrease at 10k triple size and 12% decrease at 100k triple size than ATLAS approach. The results depict that, the decrease in false positive rate at 10k triple size is nearly 8, 14, and 16% testing on Heart diseases dataset, nearly 8, 12, and 22% testing on Heart Attack dataset, and nearly 6, 14, 22% testing on Diabetes dataset than the ATLAS, FBASAM, and OBSAA techniques respectively. After combining the three healthcare dataset results with reduced False Negative Rate (FNR), the results indicate a 7%, 13%, and 18% decrease compared to the ATLAS, FBASAM, and OBSAA techniques respectively.

Fig. 14 (d) represents the TPR value of IHC-AA-IoTSD under changing triple rate and the result evidences its capacity of enhancing the TNR value by 34–23%, which is nearly 5, 9 and 11% higher than the TNR obtained by ATLAS, FBASAM, and OBSAA tested on three healthcare datasets at 10k triples. The results are considered on an average and it achieves the 5, 9, and 11% more than the TPR value achieved by ATLAS, FBASAM, and OBSAA. The importance of IHC-AA-IoTSD is based on agent preprocessing mechanism used for annotating the triple data and SPARQL queries that could be optimally applicable for healthcare data annotations.

Fig. 14 (e) represents the Precision rate of IHC-AA-IoTSD under varying data triple sizes at 10k, 20k, 30k, 40k, 50k, and 100k on 3 different datasets. The result evidences by enhancing the Precision value around 5–21%, which is nearly 5, 10 and 17% higher than the Precision rate simplified by ATLAS, FBASAM, and OBSAA testing on Heart Diseases dataset using 10k triples. Similarly, nearly 6, 10 and 15% higher than the Precision rate simplified by ATLAS, FBASAM, and OBSAA testing on Heart Attack dataset at 10k triples, and nearly 7, 12 and 21% higher than the Precision rate obtained by ATLAS, FBASAM, and OBSAA testing on Diabetes dataset using 10k triples. After combining the three healthcare dataset results with increased Precision rate, the results indicate that around 6%, 11%, and 17% increase than the ATLAS, FBASAM, and OBSAA techniques respectively.

Fig.14 (f) depicts the reduced FPR of IHC-AA-IoTSD under varying data triples and ensures that, it can minimize the FPR around 6–24%, which is nearly 3% decrease at 10k triple size and 6% decrease at 100k triple size compared to the ATLAS framework, testing on heart diseases dataset. Testing on heart attack dataset, it can reduces the FNR about 8–29%, which is nearly 5% decrease at 10k triple size and 8% decrease at 100k triple size compared to the ATLAS framework.

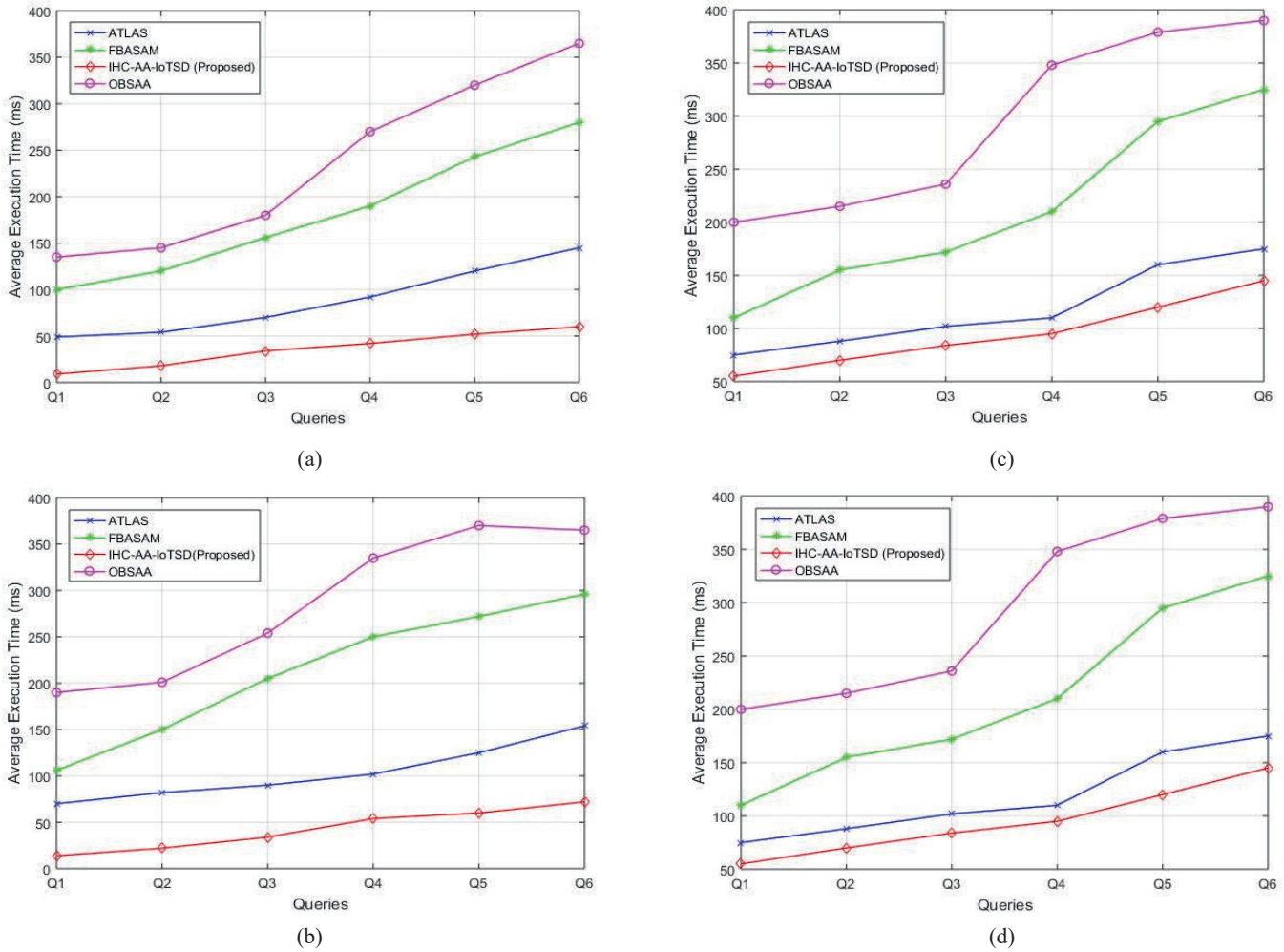


Fig.15. Average Execution Time (ms) by various queries (a) at 10k triples (b) at 20k triples (c) at 30k triples (d) at 50k triples

Similarly, by testing on diabetes dataset, it can minimize the false negative rate of about 5–25%, which is nearly 7% decrease at 10k triple size and 11% decrease at 100k triple size than ATLAS approach. The results depict that, the decrease in false positive rate at 10k triple size is nearly 3%, 6%, and 10% testing on Heart diseases dataset, nearly 5%, 8%, and 14% testing on Heart Attack dataset, and nearly 7%, 11%, and 16% testing on Diabetes dataset compared to the ATLAS, FBASAM, and OBSAA techniques respectively. After combining the three healthcare dataset results with reduced False Positive Rate (FPR), the results indicate 5%, 8%, and 13% decrease compared to the ATLAS, FBASAM, and OBSAA techniques, respectively.

In the third experimental investigation of IHC-AA-IoTSD validated through the Average Execution Time of various queries over the benchmark mechanisms such as ATLAS, FBASAM, and OBSAA techniques respectively.

Fig. 15 (a-d) shows measured average execution time by various queries from Q1 to Q6 at 10k triples, 20k triples, 30k triples, and 50k triples, respectively. The result proves that IHC-AA-IoTSD is able to maintain the Average Execution Time of 27 ms at various queries and this Average Execution Time infers 12% success rate higher than ATLAS, FBASAM, and OBSAA. Figs. 15 (a-d) highlights the predominance Average Execution Time of IHC-AA-IoTSD over ATLAS, FBASAM, and OBSAA under the 10k triples, 20k triples, 30k triples, and 50k triples respectively. The result confirms that IHC-AA-IoTSD is able to endure its Average Execution Time of 86 ms at various queries even when the triple size is increased. IHC-AA-IoTSD enables a superior Average Execution Time of 16% when compared to ATLAS, FBASAM, and OBSAA with all the queries.

### *E. Complexity Analysis*

Moreover, the time complexity of IHC-AA-IoTSD scheme, which used algorithms from 1 to 3, is determined to be $T(n)$ for algorithm perceived instances staring from j is 1 to n and i value between 1 to 9. The time complexity of algorithm 1 is calculated by $T_1(n)$, algorithm 2 is by $T_2(n)$, and algorithm 3 is by $T_3(n)$. At last, these three times complexities will be combined to get the overall time complexity $T(n)$. Let us see how to find the time complexity of $T_1(n)$, it is as follows in Eq. (5.1).

$$T_1(n) = t_1 + t_2 + \cdots + t_9,$$
$$T_1(n) = 1 + (1 + 1 + 1 + 1 + 1 + 1 + 1 + 1)n,$$
$$T_1(n) = 1 + 9n = \theta(n) \tag{5.1}$$

Similarly, the time complexity is generated for algorithm 2 as follows in Eq. (5.2).

$$T_2(n) = t_1 + t_2 + \cdots + t_9,$$
$$T_2(n) = 1 + (1 + 1 + 1 + 1 + 1 + 1 + 1 + 1)n,$$
$$T_2(n) = 1 + 9n = \theta(n) \tag{5.2}$$

Similarly, the time complexity is generated for algorithm 3 as follows in Eq. (5.3).

$$T_3(n) = t_1 + t_2 + \cdots + t_9,$$
$$T_3(n) = 1 + n + (n * n) + (n * n * n) + 1,$$
$$T_3(n) = 1 + n + n^2 + n^3 + 1 = \theta(n^3) \tag{5.3}$$

Hence the total time complexity of IHC-AA-IoTSD is $T(n)=\theta(n^3)$.

## VI. Conclusion and Future Work

In the IoT streaming data era, the sensor devices are generating dynamic data continuously, which is heterogeneous. The IoT data also consists of the real-time streaming data. To perform analysis and annotating the streaming data is a current research problem faced by researchers. Therefore, in this paper, the authors proposed IHC-AA-IoTSD mechanism for unifying the hierarchical clustered data using SPARQL queries. The experimental investigation of IHC-AA-IoTSD has been conducted on three popular healthcare datasets by varying triple data and measuring detection accuracy, precision, TPR, TNR, FPR, and FNR. In the first experimental investigation, the TPR value has been measured under the streaming of triples with stable FPR diverse with 10, 20, 30 and 40%, respectively. In the second experimental investigation, the average results have been taken for an account and proves that the IHC-AA-IoTSD outperforms compared to benchmark mechanisms such as ATLAS, FBASAM, and OBSAA. In the third experimental investigation, the query average execution time has been calculated by taking six different queries under 10k, 20k, 30k, and 50k triples. Considering that IoT streaming data is dynamic and heterogeneous, the proposed mechanism overwhelmed by efficiently annotating the hierarchical clustered data. Moreover, the proposed IHC-AA-IoTSD mechanism outperforms compared to the existing state of the art schemes. In future, the proposed mechanism can be optimized by considering the hash table (key, value pair) for storing SPARQL queries. In addition, artificial intelligent systems need quicker decisions on streaming data. In this scenario, the proposed mechanism may be useful and can achieve efficient results. Besides, it can be considered applying advanced deep learning techniques like Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), for annotating IoT sensor data with optimum results.

## References


[1] G. Xiao, J. Guo, L. D. Xu, and Z. Gong, "User interoperability with heterogeneous IoT devices through transformation," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1486– 1496, 2014.

[2] Rohit Dhall & Vijender Kumar Solanki, "An IoT Based Predictive Connected Car Maintenance Approach," International Journal of Interactive Multimedia and Artificial Intelligence, ISSN 1989-1660,Vol 4, no 3, pp 1-13, 2017.

[3] Sivadi Balakrishna, M Thirumaran, R. Padmanaban, and Vijender Kumar Solanki "An Efficient Incremental Clustering based Improved K-Medoids for IoT Multivariate Data Cluster Analysis", Peer-to-Peer Networking and Applications, Springer, Vol 13, no 3, pp 1-23, 2019.

[4] Sivadi Balakrishna, M Thirumaran, and Vijender Kumar Solanki "Machine Learning based Improved GMM Mechanism for IoT Real-Time Dynamic Data Analysis", Journal of Revista Ingeniería Solidaria, Vol 16, No 30, e-ISSN 2357-6014, pp 1-29, 2020.

[5] H. T. Lin, "Implementing Smart Homes with Open Source Solutions", International Journal of Smart Home Vol.7 Issue. 4, pp 289–295, 2013.

[6] Antunes, Mário, Diogo Gomes, and Rui L. Aguiar. "Towards IoT data classification through semantic features." Future Generation Computer Systems, Vol. 8 no 6, pp 792-798, 2018.

[7] M. Junling, J. Xueqin, and L. Hongqi, "Research on Semantic Architecture and Semantic Technology of IoT," Research and Development, vol. 8, no. 5, pp. 26–31, 2014.

[8] Q. Xu, P. Ren, H. Song, and Q. Du, "Security enhancement for IoT communications exposed to eavesdroppers with uncertain locations," IEEE Access, vol. 4, pp. 2840–2853, 2016.

[9] D. Rong, "The Research on Automatic Semantic Annotation Methods", Lanzhou University of Technology, Lanzhou, China, 2012.

[10] F. Chen, C. Lu, H.Wu. Wu, and M. Li, "A semantic similarity measure integrating multiple conceptual relationships for web service discovery," Expert Systems with Applications, vol. 6 Issue.7, pp. 19–31, 2017.

[11] C. De Maio, G. Fenza, M. Gallo, V. Loia, and S. Senatore, "Formal and relational concept analysis for fuzzy-based automatic semantic annotation," Applied Intelligence, vol. 40, no. 1, pp. 154– 177, 2014.

[12] P. Barnaghi, W. Wang, L. Dong, and C. Wang, "A linked-data model for semantic sensor streams," IEEE International Conference on and IEEE Cyber, Physical and Social Computing, Green Computing and Communications (GreenCom '13), Beijing, China, pp. 468–475August

2013.
[13] S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, "A knowledge-based approach for real-time IoT data stream annotation and processing," in Proc: International Conference on Internet of Things, IEEE, pp. 215–222, 2014.
[14] W. Wei and P. Barnaghi, "Semantic annotation and reasoning for sensor data," in Smart Sensing and Context, vol. 5741 of Lecture Notes in Computer Science, pp. 66–76, Springer, Berlin, Germany, 2009.
[15] P. Chenyi, Service-oriented entity semantic annotation in internet of things [M.S. thesis], South China University of Technology, Guangzhou, China, 2015.
[16] J. Bing, "Research on semantic-based service architecture and key algorithms for the internet of things", [Ph.D. thesis], Jilin University, Changchun, China, 2013.
[17] Z. Ming, "Research on several key issues in internet of things applications", [Ph.D. thesis], Beijing University of Posts and Telecommunications, Beijing, China, 2014.
[18] E. Charton, M. Gagnon, and B. Ozell, "Automatic semantic web annotation of named entities," in Advances in Artificial Intelligence, vol. 6657 of Lecture Notes in Comput. Sci., Springer, Berlin, Germany, pp. 74–85, 2011.
[19] G. Diallo, M. Simonet, and A. Simonet, "An approach to automatic ontology-based annotation of biomedical texts," Lecture Notes in Computer Science, vol. 40 no. 31, pp. 1024–1033, 2006.
[20] M. Jacoby, A. Antonic, K. Kreiner, R. Lapacz, J. Pielorz. "Semantic interoperability as key to IoT platform federation," in LNCS 10218: Interoperability and Open- Source for the Internet of Things, pp. 3-19, 2017.
[21] A.P. Plageras, K.E. Psannis, C. Stergiou, H. Wang, B.B. Gupta, " Efficient IoT- based sensor BIG Data collection- processing and analysis in Smart Buildings", Future Generation Computer Systems, 82, pp 349-357, 2018.
[22] A. E. Khaled, S. Helal, "Interoperable communication framework for bridging RESTful and topic-based communication in IoT", Future Generation Computer Systems, Elsevier, 92, pp 628-643, 2019.
[23] Kolozali, S. Puschmann, D.; Bermudez-Edo, M.; Barnaghi, P. "On the Effect of Adaptive and Non adaptive Analysis of Time-Series Sensory Data", IEEE Internet Things J., 3, pp 1084–1098, 2016.
[24] Mazayev, Andriy, Jaime A. Martins, and Noélia Correia. "Interoperability in IoT through the Semantic Profiling of Objects." IEEE Access 6, pp 19379-19385, 2017.
[25] Mayer, Simon, Jack Hodges, Dan Yu, Mareike Kritzler, and Florian Michahelles. "An open semantic framework for the industrial Internet of Things." IEEE Intelligent Systems 32, no. 1, pp 96-101, 2017.
[26] Shi, Feifei, Qingjuan Li, Tao Zhu, and Huansheng Ning. "A survey of data semantization in internet of things." Sensors 18, no. 1, 313, 2018.
[27] Al Zamil, Mohammed Gh, Majdi Rawashdeh, Samer Samarah, M. Shamim Hossain, Awny Alnusair, and Sk Md Mizanur Rahman. "An annotation technique for in-home smart monitoring environments." IEEE Access 6, pp 1471-1479, 2018.
[28] Moutinho, Filipe, Luís Paiva, Julius Köpke, and Pedro Maló. "Extended Semantic Annotations for Generating Translators in the Arrowhead Framework." IEEE Transactions on Industrial Informatics 14, no. 6, pp 2760-2769. 2018.

Sivadi Balakrishna

He received his Bachler of Technology (B.Tech) in the Department of Computer Science and Engineering from Jawaharlal Nehru Technological University (JNTU) in 2010 and Master of Technology (M.Tech) in the Department of Computer Science and Engineering from Jawaharlal Nehru Technological University (JNTU) in 2013, Kakinada, AP, India. He is currently a Full-time Ph.D research scholar from Pondicherry Engineering College in the Department of Computer Science and Engineering, Pondicherry University (A Central University), Pondicherry, India. He has qualified NET (National Eligibility Test) in Dec-2018, which was conducted by UGC. He has more than 4 years of teaching experience in various reputed institutions in computer science and engineering department. He has published more than 15 research articles in various reputed International Journals, International Conferences and Book Chapters. His current research interests are Internet of Things (IoT), Machine Learning, and Semantic Technologies.

M.Thirumaran

He is currently working as an Assistant Professor in Department of Computer Science and Engineering in Pondicherry Engineering College. He has completed his B.Tech in Pondicherry Engineering College in the year 2000 and completed his Post Graduation in Pondicherry University in 2002. He has qualified NET examination for three consecutive years from 2004 to 2006 which was conducted by UGC. He completed his Ph.D in Pondicherry University in the year 2014. He has interested in the domains of Service Oriented Architecture, Web Technology, Web Application Security, Principles of Compiler Design and Automata Theory and Computation. Also he has teaching experience around 15 years in the field of Computer Science Engineering. He has published more than 80 research papers in various reputed International Conferences and International Journals.

Vijender Kumar Solanki

Vijender Kumar Solanki, Ph.D., is an Associate Professor in Department of Computer Science & Engineering, CMR Institute of Technology (Autonomous), Hyderabad, TS, India. He has more than 11 years of academic experience in network security, IoT, Big Data, Smart City and IT. Prior to his current role, he was associated with Apeejay Institute of Technology, Greater Noida, UP, KSRCE (Autonomous) Institution, Tamilnadu, India & Institute of Technology & Science, Ghaziabad, UP, India. He has attended an orientation program at UGC-Academic Staff College, University of Kerala, Thiruvananthapuram, Kerala &Refresher course at Indian Institute of Information Technology, Allahabad, UP, India. He has authored or co-authored more than 50 research articles that are published in journals, books and conference proceedings. He has edited or co-edited 10 books in the area of Information Technology. He teaches graduate & post graduate level courses in IT at ITS. He received Ph.D in Computer Science and Engineering from Anna University, Chennai, India in 2017 and ME, MCA from Maharishi Dayanand University, Rohtak, Haryana, India in 2007 and 2004, respectively and a bachelor's degree in Science from JLN Government College, Faridabad Haryana, India in 2001.He is Editor in International Journal of Machine Learning and Networked Collaborative Engineering (IJMLNCE) ISSN 2581-3242, Associate Editor in International Journal of Information Retrieval Research (IJIRR), IGI-GLOBAL, USA,ISSN: 2155-6377 | E-ISSN: 2155-6385 also serving editorial board members with many reputed journals. He has guest edited many volumes, with IGI-Global, USA, InderScience & Many more reputed publishers.

Edward Rolando Núñez-Valdez

Ph.D. from the University of Oviedo in Computer Engineering. Master's in software engineering from the Pontifical University of Salamanca and B.S. in Computer Science from Autonomous University of Santo Domingo. He has participated in several research projects; He has taught computer science at various schools and universities and has worked in software development companies and IT Consulting for many years. He has published several articles in international journals and conferences. Currently working as Assistant Professor at the University of Oviedo in Spain. His research interests include Software Engineering, Object-Oriented technology, Web Engineering, Recommendation Systems, Artificial Intelligence, Distributed Systems and DSL.