A Semantic Question Answering in the Domain of Smart Factories

Orcl̀§un Orucl̀§ and Uwe Aßmann

March 19, 2020

A Semantic Question Answering in the Domain of Smart Factories

Orçun Oruç*, Uwe Aßmann†
*Software Technology Group, Technische Universität Dresden, orcun.oruc@tu-dresden.de, Dresden, Germany
†Software Technology Group, Technische Universität Dresden, uwe.assmann@tu-dresden.de, Dresden, Germany

***Abstract*—Industrial manufacturing has become more interconnected between smart devices such as the industry of things edge devices, tablets, manufacturing equipment, and smart phones. Smart factories have emerged and evolved with digital technologies and data analytics in manufacturing systems over the past few years. Basically, smart factories make complex data enables digital manufacturing and smart supply chain management and enhanced assembly line control. However, the more data created by smart factories, the harder it is for human operators and experts to understand the meaning of data because of the readability of machine-readable data. Nowadays, smart factories produce a large amount of data that needs to be apprehensible by human operators and experts in decision making. However, linked data is still hard to understand and interpret for human operators, thus we need a translating system from linked data to natural language or summarize the volume of linked data by eliminating undesirable results in the linked data repository. In this work, we propose a semantic question answering that can understand and interpret linked data repository in the domain of a smart factory. We have used heterogeneous RDF Turtle datasets from one of the OPC UA Server which is connected to the Fraunhofer IWU [1] edge devices, an annotated time-series data SPARQL endpoint and statically generated data from eniLINK [2] linked data repository. The semantic question answering might interpret the data from open or closed domain questions, but rather we will examine the question answering system in a restricted smart factory domain with the above-mentioned data source. Lastly, we will perform qualitative and quantitative evaluation of the semantic question answering, as well as discuss findings and conclude the main points regarding our research questions.**

***Index Terms*—Semantic Web, Web 3.0, Information Retrieval, Natural Language Processing, Industry 4.0**

I. Introduction

Currently, a vast amount of unlabeled data can not be used by applications, World Wide Web Consortium (W3C) decided to create standardization of the Web 3.0 called Semantic Web in order to apply Linked Open Data [3] concept. In this concept, hypertext documents of the web sites or ad-hoc have been connected to each other through links such as Uniform Resource Identifiers (URIs)[4]. As part of this development, Fraunhofer IWU started to organize its smart factories that are capable of generating structured linked data. Smart factories can use real-time data or linked data in order to diminish bottlenecks in assembly lines, provide predictive maintenance, enhance human-machine interaction with digitalization.

The present study introduces a human-machine-interaction concept for smart factories in terms of linked data processing integrated into a question answering. The Semantic Web is a state-of-the-art research area that orchestrates the use of understanding in linked data between humans to machines and machines to machines. You can link data and documents to external data through linked data. In the present day, smart factories equipped with intelligent manufacturing devices, sensors, and actuators create a massive amount of data.

A semantic question answering is used for information retrieval to provide answers to questions through linked data. The proposed semantic question answering can understand complex natural language expressions, and it can respond to the user by answers. Mainly, the semantic question answering system employs unstructured data or structured data. We obtain linked data generated by an OPC-UA Server named *Dynamic Server* and the *eniLINK* streaming data. The empirical analysis indicates the answer return rate and precision; therefore, it evaluates the usability for a human operator, experts or an end-user web application. The goal of this research is to show a model of semantic question answering for a smart factory that utilizes the natural language expressions as sentences, questions or keywords to give a precise and rapid answer to human operators or experts.

The remainder of this paper is structured as follows: Section II will provide a brief overview of smart factories and Industry 4.0 and serialization process of linked data from another type of data such as the *Information Model* and from streaming data to the linked data. Section III introduces the theoretical background of natural language understanding and practical implementation of the question answering. Section IV summarizes the research approach of the semantic question answering aspect of the smart factory constructed by Fraunhofer IWU. In Section V, we implement an application and we give the implementation details of the present study. As for Section VI, we will explain the test environment; accordingly, we give the results of the semantic question answering. Section VII explains the state-of-the-art status the *Industry 4.0* and *Smart Factories*. Then, we answer specified research questions in order to clarify key points with discussion in Section VIII. Finally, we conclude in Section IX.

[1]https://www.iwu.fraunhofer.de/en.html
[2]http://platform.enilink.net/
[3]https://lod-cloud.net/
[4]https://www.w3.org/DesignIssues/LinkedData.html

II. Background

A. Smart Factories and Industry 4.0

The definition of the smart factory has been explored over the past few years. Basically, a smart factory consists of new integrable technological terms such as Machine Learning and Artificial Intelligence through intelligent devices such as tablets, smartphones, and sensors in order to make apprehensible models from unknown data areas in the manufacturing. *Industry 4.0* is a defined term that relates to the notion of smart factory bringing researchers to find state-of-the-art applications such as question answering systems, manufacturing augmented reality and semantic sensor networks.

A smart factory is a highly digitized and connected production facility that relies on smart manufacturing [1]. This concept is one of the key outcomes of Industry 4.0, which intelligently changes manufacturing technologies. Smart manufacturing is a term coined by a set of departments of the United States [2]. The central power of the smart factory is that it makes data collection possible. Additionally, sensors enable the monitoring of specific processes throughout the factory that increases awareness of what is happening on multiple levels [3].

The development of *Industry 4.0* has a significant influence on the manufacturing industry. In the era of smart manufacturing systems, *Industry 4.0* needs to standardize all connection pipelines in smart factories. The primary objectives of *Industry 4.0* are making the manufacturing technologies of factories more capable of handling semantic triples, optimizing the chain of processes, and enhancing the capabilities of communication with each other. Moreover, *Industry 4.0* enforces end-to-end digital integration of engineering throughout the value chain to facilitate highly customized products, thus reducing internal operating costs [4].

B. Linked Data Serialization

Our data sources are structured as semantic triples such as Turtle, RDF, or OWL, which it is also publicly available in the Github repository [5]

1) The Semantic Data for OPC Unified Architecture Information Model: OPC Unified Architecture was developed for devices of industrial internet of things to remedy problems about *service orientation*, *loose coupling*, and *object-orientation paradigm*. The OPC UA has been evolved from OPC to OPC UA over the past few decades and the architectural designed was entirely changed. The fundamental disadvantage of OPC was that it was restricting devices to connect just to Windows-based operating systems. After developing the *Distributed OPC* and *OPC UA* ideas, the foundation of *Open Platform Communications* has constructed a viable concept that consists of object-oriented, loose-coupling and service orientation in manufacturing systems.

Aside from the OPC UA is a complex protocol; the OPC UA is one of the ubiquitous communication protocols that can be used in the various stages of the manufacturing. Thanks to the OPC client-server architecture, any devices can connect to the protocol in a manufacturing system. A programmable logic controller, a sensor or an actuator can connect to the same server, and they can assign their values into different folder organizations to represent data in an address space. The address space is a data plane for an OPC UA server; hence it should coordinate *variables*, *methods*, *objects*, and *nodes* respectively. An end-user can identify primitive and user-defined types so that the complex structure of devices can be represented as a whole in a big data plane. However, this data plane only provides definitions and types.

The *Information Model* supports object-oriented paradigms such as abstraction and inheritance between *References* and *Objects*. It is well known that an object can live as a *Node Class* in the address space. The objects may have relationships with other objects in the *Information Model*. Utilizing *References*, a user can traverse in the address space of OPC UA to reach all levels of nodes and variables. Nevertheless, neither the *Address Space* of OPC UA nor the *Information Model* is unable to understand the meaning of data. The semantic understanding of the *OPC UA Information Model* has a vital role in performing an answering question system. The *Information Model* holds all device-specific information such as *device type*, *data changes of the device*, *vendor type* and *relationship between devices*. These information sources could be helpful to human operators or experts aspect of system information in regard to manufacturing systems.

2) Mapping the OPC UA Data into Semantic Data: The primary data sources are semantically parsed data from *eniLINK* [5] and the OPC-UA server in Fraunhofer IWU named *Dynamic Server*. In the phase of the OPC UA server generated data, we used an SDK, which is published by *FreeOPCUA*[6] and TU-Dresden[7]. We contributed to the aforementioned projects with extra conversion steps such as XSLT and triple store processing.

OPC UA standard utilizes an information model and the information model can be used to simulate OPC UA Servers with Extensible Markup Language (XML). Due to the nature of the XML language, it is a language that depends on strong hierarchical elements and has its own data model with elements and attributes that are hardly parseable. However, semantic data such as Resource Description Framework (RDF) can employ triples with the SPARQL query language.

[5]https://github.com/zointblackbriar/QuestionAnswering/tree/master/AlgorithmQuestionAnswering/SemanticSource

[6]http://freeopcua.github.io/

[7]https://github.com/plt-tud/opc_ua_xml_export_client

Algorithm 1 Node Extraction

```
function MainFunction()                                  ▷ Starting point
    export = ServerExport(serverurl, filename)
    export.IMPORT NODES(serverurl)
    export.EXPORT FILE(outputFile, namespaces)
end function
function Build Node Tree($nodes$)                        ▷ Node Formatting
    $client \leftarrow$ GetEndpoint()
    $client \leftarrow$ Client(serverurl)
    $nodecumulated$ ←None
    $nodeID$ ←0
    for node < nodes do
        nodecumulated = node.nodeid.Namespaceindex
        for ref < node.getreferences() do
            nodecumulated.extend( ref.nodeid.Namespaceindex
        end for
        nodecumulated = list(set(nodecumulated)       ▷ Clear duplicates
    end for
    return $nodeID$                                      ▷ Return node id list
end function
function Import Nodes($serverurl$)                       ▷ Traverse Node
    client = Client(serverurl)
    client.connect()
    for ns < client.getNamespaces() do
        namespaces[client.getNamespaceIndex(ns)] = ns
    end for
    root = client.getRootNode()
    child = client.iterateChildNodes()
end function
function Export File($outputFile, namespaces = None$)     ▷ Export into XML
    if namespaces != None then
        for node != nodes do
            if node.nodeid.namespaceindex is namespaces then
                nodes = [node]
            else
                nodes = list(nodes)
            end if
        end for
    end if
    export = XmlExport(client)
    export.BUILD NODE(nodes)
    export.appendXML(outputFile)
end function
```

The algorithm, as shown in Algorithm 1, identifies tree elements of a node by taking namespace indexes. The *namespace index* contains *node ids*. Once a user browses from a node to another, the user needs to know the node identification number. If the user did not scan the total number of references, the application should get all nodes that have references until the algorithm reaches all of the mesh networks. Accumulated nodes are inserted into a list to export an XML format. After obtaining XML structures, the system can convert the elements into linked data such as *Turtle RDF* through *Extensible Stylesheet Language Transformations* (XSLT). XSLT can transform from the XML format to the RDF format by minimizing the nodes without resources called blank nodes. Once the application is converted to RDF/XML format, graph libraries can deal with the conversion process into triple formats. The application only takes care of the uniform locator identifier to do a conversion, and then the application ought to arrange uniform locators by considering different from '*example.org*'.

3) Mapping an OPC UA Data into Semantic Data: Streaming data sources have intricacies in the use of linked data taken from sensors, actuators or software logs. The aspect of smart factories, sensors, and actuators that are the underlying structure of manufacturing machines mostly create streaming data. Fraunhofer IWU collects the real data source by saving it into a time-series database. The major drawback is that when the time series data are taken, the endpoint of a semantic query cannot use the unstructured data without annotating it. Such annotations can be the formulation of triples, insertion of predicates, or serialization, which means that convert one type of formal language to another. The proposed architecture utilizes a streaming semantic data annotator that extracts triples from time-series data in a database provided by Fraunhofer IWU. To this end, it serves as a federated SPARQL endpoint.

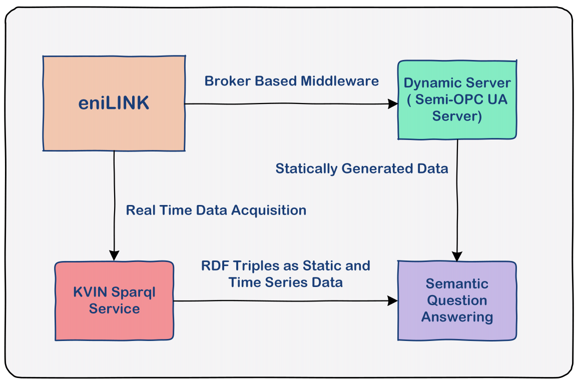
eniLINK
Broker Based Middleware
Dynamic Server (Semi-OPC UA Server)
Statically Generated Data
Real Time Data Acquisition
KVIN Sparql Service
RDF Triples as Static and Time Series Data
Semantic Question Answering

Fig. 1: KVIN Continuous SPARQL Mapping

The current study proposes a service named *KVIN*[8] to perform a SPARQL request against a specific endpoint. This

[8]https://github.com/enilink/enilink

service is based on a combination of the triple store through the *Level DB*, which is a key-value storage library written by Google Inc. It has been used as an RDF4J's extension to create a SPARQL service. After obtaining time-series data, the data are mapping the SPARQL triples. These graphs contain mapped triples with their time-stamped value so that we can employ values with some complex processes with SPARQL language. Moreover, a federated service replies to the queries determined by users, which reduces the answer return time of a question answering system. The KVIN does not create instantly hard-coded triples nor a new language such as C-SPARQL [6]. It only arranges the size of the time window and puts the graphs into the service to present them to the end-user.

SPARQL was designed for the use of semantically structured triples, not for relational datasets. *PREFIX, SELECT* and *WHERE* are there basic operators of SPARQL Protocols. *PREFIX* makes the serialization easier in referencing to the Uniform Locator, *UNION* statement can help at federating multiple triples into a single query - *OPTIONAL* statement is used to allocate a particular portion of SPARQL into triples. As can be seen, the primary goal of the query language is to federate information among various data sources.

III. Theory of the Natural Language Understanding

In natural language processing, we need to identify the structure of a natural expression to reach the step of *Query Formulation*. The following methods that we used in the practical application are concisely given.

Preprocessing and Tokenization: Chiefly, all of the natural language processing tasks starting with preprocessing, which means cleaning the data for specific tasks, could be the reduction of non-optimized data and discrepancies between the values or removing non-related morphological properties. A question answering system should parse natural language expressions as tokens. Tokenization is the initial step for *part-of-speech tagging* to parse from a natural language to lexical grammatical structure such as verbs, nouns, cardinal numbers, or adjectives.

Lemmatization and Stemming: *Lemmatization* and *Stemming* are similar to each other with one difference. While a stemming algorithm is used to find syntactical structures, a lemmatization algorithm looks for a semantic structure. Stemming clears out the morphological structure of suffix and prefixes. In our proposed system, we are supposed to use a lemmatizer and stemmer in order to reduce lexical complexities. A lemmatizer is used to examine the morphological analysis of verbs, e.g. from *"contains"* and *"contained"* to *"contain"*. Then we need to take this verb to map it into a predicate to construct a SPARQL query. The lemmatization and stemming are part of the normalization process in terms of morphological properties.

Part-of-Speech Tagging: It is a preprocessing step for parse trees to identify item taggers such as verbs, adjectives or nouns. A sentence consists of a couple of structures including expressions like nouns, verbs, pronouns, prepositions, adverbs, conjunctions, participles, and articles that are the main categories of part-of-speech processing [7]. The part-of-speech (POS) tagger mostly applies the *Markov Model* [7] that is a part of statistical natural language understanding. The *Markov Model* stands for a state that can depend on the current state, but there is no dependency between previous states. For instance, a noun or a verb defines its neighbors, e.g. nouns are preceded by determiners, adjectives, verbs [7]. As an example, a chess player moves any chess piece according to the last movement of a rival rather than guessing from the first movement of the competitor. In this way, pre-saved corpora that have a massive amount of words have to be tagged by the POS Tagger.

Parsing: The approach of parsing is two fold, which are the rule-based approach and the probabilistic approach [8]. The rule-based approach is a top-down approach to solve problems via predefined rules such as regex-parsing and character-based parsing. Therefore, a question answering system should define rules precisely to get the correct answer. Open-domain question answering systems use this approach because of the high complexity of the bottom-up approach and broad question types. Nevertheless, the rule-based approach could give undesirable results to question answering systems in restricted-domain so that this could be a time-wasting and an error-prone approach.

A dependency parser analyzes the grammatical structure of a sentence, and it gives information about the relationship among them. The dependency parser also defines the relationship between dependent words and root words. Thus, we can identify the center verbs or dependent nouns of complex sentences. This parser utilizes a dependency treebank file and word embedding files. Chiefly, a dependency parser applies the supervised machine learning method to reach a syntactical and semantical result. A constituency (phrase) parser is likely to be known as a phrase parser that has an objective is to check the grammatical structure of sentences by parsing the chunks of morphological structure. The constituency parser may not handle the relationship among language items. The dependency parser examines the grammatical structure of given natural expressions to identify the relationship between a root word and dependent words that relate to the root word.

Named-Entity Recognition: It is a subtask of information extraction to locate and classify named entities with pre-classified labels, such as names of people, organizations, locations, etc. Named-entity recognition is a method that identifies the item of a sentence as a domain-specific one.

It solves the problem of recognition in the same way that the chunking method does. However, the named-entity recognition may be trained with labeled data and it is a more advanced technique than the chunking technique, which has deep and shallow parsing methods.

Similarity Analysis: Sentence similarity is used to compare two string inputs to achieve indicative questions like *"Is the system health good?"*. Mainly, this similarity method leverages averaging word vectors such as *word2vec* and *glove* that implement *Euclidean Distance*, *Manhattan Distance* or *Cosine Similarity* [7]. In the following, three similarity methods that we analyzed are introduced:

The *Levenshtein Distance* denotes the calculation time that could be $O(|s1| \ x \ |s2|)$ using $O(min(|s1|, \ |s2|))$ space. After calculating the distance between s1 and s2, the result is divided into the maximum length of string [9]. The *Jaro Winkler* has a transposition matrix t with common characters that are calculated together to reach the similarity value [16]. *The Jaccard Similarity* algorithm takes into consideration the size of the intersection divided by the size of the union of two sets [9]. Under the same test data and methods, similarity levels of the *Jaccard*, *Jaro Winkler*, and *Levenshtein* are 0.8095, 0.7544, and 0.58 respectively. The higher score shows a better performance for similarity measurement.

In order to calculate the word-based similarity, we perform the WordNet with glove vectors. Such vectors are pre-calculated synset values are compiled and stored into a file . These synset values show the similarity value with the cosine similarity algorithm. The WordNet can calculate the similarity of acronym and hypernym except for synonym. The calculation of semantic similarity is a hard and complicated process. As we will explain in the following scenario, two phrases such as *'Internet of Things'* and *'Mesh Network'* are semantically similar. The first implies "the network of physical objects with electronics, software, sensors, and connectivity" and the latter implies "the topology of a network whose components are all connected directly to every other component". We cannot easily calculate this semantic similarity. Instead of calculating semantic similarity, we can calculate word vectors of verbs and nouns related to *similarity synset*. If a computed synset value is above the threshold value, a question answering can accept these two strings that are constructed similarly. In the practical implementation, we have used verb synonym similarity to map onto <*IRI: predicate*> triples.

Question Classification: Questions should be categorized to get the correct answer. It is a part of question processing that can parse the question input and assign it to the correct labels. Machine learning methods can define the derivation of an expected answer. This paper utilized *Logistic Regression* and *Support Vector Machine* for the question classification phase. While the support vector machine was classifying the question with *TREC Dataset*[9], the logistic regression examined the type of question at the Github repositories[10] [11]. Questions are grouped with coarse-grained labels, which are *Abbreviation*, *Entity*, *Description*, *Human*, *Location*, and *Numeric*. On the other hand, another dataset that we have been trained with Logistic Regression comprising of *'what'*, *'quantity questions-how many, how much*, 'who', 'unknown' and 'why' labels. The *Logistic Regression* and *Linear Support Vector Classification* have supervised machine-learning methods by identifying coarse-grained question indicators with pre-trained labels. Logistic Regression estimates the parameter with a logistic function. The type of regression allows classifying the aforementioned labels according to multi-labels . The *Support Vector Machine* aims to improve the quality of hyperplane that separates multi-class labels. *Linear SVC*[12] is such a method that implements a linear kernel function through the *Support Vector Machine*. The *Newton-cg* has a gradient descent function that reduces the error rate during each iteration to find out the global minimum. The *Limited BFGS* is an optimization method that can be used instead of *Newton-cg*. *Logistic Regression Cross-Validation (CV)*[13] applies cross-validation to train and test datasets by splitting at particular percentages between them. The result has been listed in Table I.

Parameters	Precision	F1	Recall
Newton-cg	95.55%	95.56%	95.57%
Linear SVC	92.75%	92.76%	92.77%
Limited BFGS	94.21%	94.22%	94.23%
Logistic Regression CV	95.63%	95.63%	95.64%
Linear SVC for Li-Roth Taxonomy	65%	45.5%	35%

TABLE I: The evaluation of the Question Classification

IV. Research Approach

Research Questions:

1) *RQ-1: Can a semantic question answering utilize heterogeneous linked data sources (e.g., OPC UA Information Model, streaming data, static data) in the domain of smart factory?*

2) *RQ-2: What are the requirements of the Semantic Question Answering for smart factories?*

[9]https://trec.nist.gov/data.html
[10]https://github.com/swapkh91/Question-Classification
[11]https://github.com/5hirish/adam_qas
[12]https://scikit-learn.org/stable/modules/svm.html
[13]https://scikit-learn.org/stable/modules/generatedsklearn.linear_model.LogisticRegressionCV.html

3) *RQ-3: Can we generalize our approach to other plants and how did we contribute to the research area?*

RQ-1: Today, a smart factory creates a massive amount of data by leveraging big data analysis technology. However, the data source suffers from comprehensible by humans. This research question relates to the implementation of a serialization process into linked data. This research question evaluates the types of data sources by implementing an application.
RQ-2: This research question relates to the algorithm design thinking and domains-specific requirements to fulfill information retrieval theory and natural language understanding. This question has to evaluate the practical application.
RQ-3: This research question examines the viability of the proposal in an aspect of the division of a plant or a smart factory. Generated new test questions set to evaluate our semantic question answering.

V. Implementation

We implement a mixed parsing based approach in order to define essential elements of a natural query. The major priority is to detect <*subject-predicate-object*> triples and then map the verbs and nouns onto template SPARQL. This template was created according to the requirements of a smart factory. For instance, dynamic queries that fetch information from streaming data possibly need *SUM*, *AVG*, and *MIN* filter statements of SPARQL language.

As for static queries, we have hiearchical triples that contain units of the smart factory and linked data of the *Information Model*. Listing 1 and 2 show examples regarding hierarchical triples of the smart factory of eniLINK and annotated OPC UA linked data. Such predicates <*factory:contains*> should be parsed and they need to be matched with verbs. However, this may lead us to a misconception to match synonym verb of predicates. Therefore, as illustrated in Figure 2, we inserted an extra step to identify the synonym of verbs.

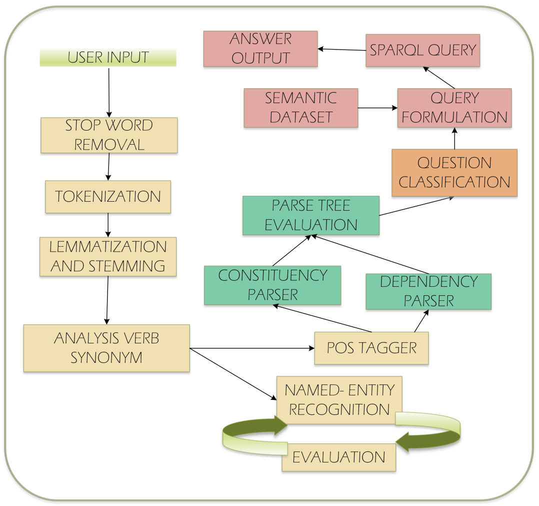

Fig. 2: Natural Language Processing for Question Answering

Listing 1: Sample triples of the eniLINK hierarchical data [10]

```
<http://linkedfactory.iwu.fraunhofer.de/
    ↪ linkedfact
ory/linkedfactory/demofactory/machine10>
factory:contains
<http://linkedfactory.iwu.fraunhofer.de/
    ↪ linkedfact
ory/demofactory/machine10/sensor1>,
```

Listing 2: Sample triples of the linked OPC UA Data

```
<unknown:namespace#UANodeSet/
    ↪ UAVariable_321> :BrowseName "0:
    ↪ MinSupportedSampleRate" ;
       :DataType "Duration" ;
       :NodeId "i=2017" ;
       :ParentNodeId "i=2013" ;
       :DisplayName <unknown:namespace#
           ↪ UANodeSet/UAVariable_321/
           ↪ DisplayName> .
<unknown:namespace#UANodeSet/
    ↪ UAVariable_321/DisplayName> rdf:
    ↪ value "MinSupportedSampleRate" .
```

After taking input from any user, stop-word preprocessing starts to filter unnecessary characters such as question marks, exclamation points, commas, dots, or determiners. Tokenization is the next step to reduce the size of characters to provide optimization in natural language processing and it reduces the complexity of instances of sequence characters. Lemmatization and stemming are fundamental steps before *WordNet* verb analysis since the primary target is to extract verb, nouns and related chunking to formulate a SPARQL query that can answer.

There is an if-else statement for the named-entity recognition after finding synonyms of the verb. As previously explained, it is a way of extracting the most common entities such as locations or names. A question answering application can face problems in identifying domain-specific names, locations, or organizations. For instance, the *linkedfactory* can be comprehensible for Fraunhofer IWU's smart factory, but another smart factory or different domain may not know what kind of entity this is. Therefore, if the question answering can catch the entity-relationship pair as shown in Figure 3, the question answering system inserts natural expressions into shallow and deep syntactic parsing.

For dynamic queries, the question answering system applies a similarity measurement. In Algorithm 2, the similarity flag employs a sentence similarity in the following case. *"Is the system in trouble ?"* is a reasoning query. The system should interpret this query, and the system needs to know exactly the semantic meaning of the sentence. However, the above-mentioned approach is similarity-based identification. When a user asked a question *"Is the system trouble for* *sensor1 in machine1?"* the semantic question answering can interpret a reasoning question through machine-readable annotations.

Algorithm 2 Query Formulation
```
1: function QUERY FORMULATION($naturalinput$)
2:     $query \leftarrow QueryWithPrefixes$
3:     $r \leftarrow contituent.parse.tree$
4:     $indirectdependency \leftarrow dependendency.parse.tree$
5:     while $nodes \neq leafs.terminal$ do          ▷ Until leaf nodes(Terminals)
6:         $verbs \leftarrow$ PARSER(nodes)
7:         $nouns \leftarrow$ PARSER(nodes)
8:         $similarityflag \leftarrow$ WORDLATENANALYSIS(verbs)
9:         if StaticInformation is True then
10:            $indirectdependencyFlag \leftarrow$ DEPENDENCYPARSER(nodes)
11:            if similarityflag and IndirectDependency is true then
12:                $object \leftarrow$nouns
13:                $predicate \leftarrow$verbs
14:                query += object + predicate + ?subject
15:            else
16:                $subject \leftarrow$nouns
17:                $predicate \leftarrow$verbs
18:                query += ?object + predicate + subject
19:            end if
20:        end if
21:        if DynamicInformation is True then
22:            $predicate \leftarrow$ PARSER(nodes)
23:            $object \leftarrow$ PARSER(nodes)
24:            $similarityflag \leftarrow$ SENTENCESIMILARITY(input)
25:            query += object + predicate + ?subject
26:        end if
27:    end while
28:    return $query$
29: end function
```

The architecture has provided a SPARQL endpoint for local static data, and the KVIN presents a SPARQL Endpoint for time-series data. We are using different techniques for different question types. In case of a given natural language expression as below, we can specify deep and shallow parsing diagram, as depicted in Figure 3:

"Could you tell me which one contains fofab?"

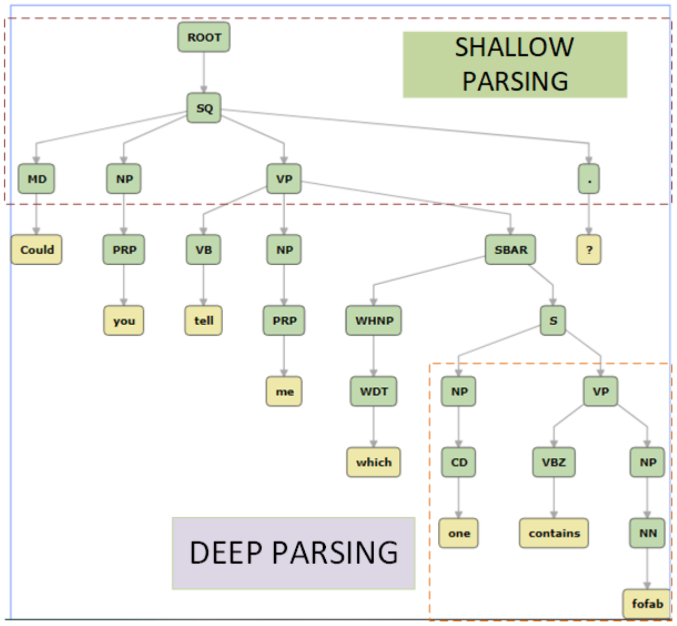

Fig. 3: An example sentence from Stanford CoreNLP [11].

We specified noun and verb phrases at a basic level so that they are using a shallow parsing that can alleviate the constituency-parsing disambiguities. If the system catches the right verb-noun pairs, it should eliminate expressions to reach the origin of the noun or verb. Such expressions may represent determiners, adjectives, or pronouns. As shown in Figure 4, the system has two verbs that it needs to map the predicate of triple onto the Turtle RDF data source. If it may find out the similarity level of *'contains'* and *'tell'*, the question answering could say the essential verb to be evaluated. However, the order of a verb is important for direct and indirect questions. As shown in Figures 4 and 5, multiple objects have relationships with the head verbs *'tell'* and *'contains'*. Subjects and objects can inverse the order of SPARQL query. In this case, the system needs to identify universal dependencies [14]. The named-entity recognition can show the types of relationships as illustrated in Figure 6 and Figure 7. A drawback of this identification is a particular keyword can perplex of the identifier, noun, etc. In essence, the question answering system needs more in-depth analyses to solve the perplexities of unique keywords and open-domain words.

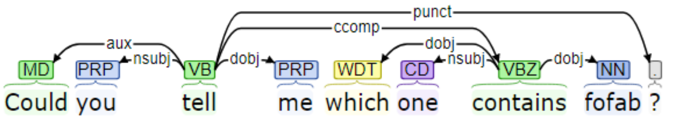

Fig. 4: A Reversed Query Dependency Parser [11]

[14]https://universaldependencies.org/

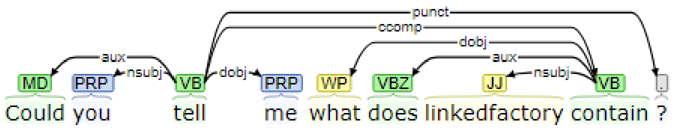

Fig. 5: A Direct Query with Dependency Parser [11]

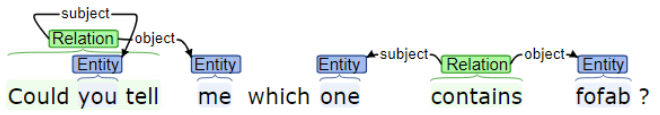

Fig. 6: Named-Entity Recognition Stanford CoreNLP [11]

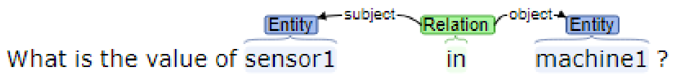

Fig. 7: Named-Entity Recognition with OpenIE [11]

VI. Evaluation

A. *Test Environment*

In the evaluation phase, the data sources linked data from the OPC UA Server, eniLINK linked data that consist of elements under the linkedfactory [12] and streaming data that resides in eniLINK. As previously detailed in Linked Data Serialization (II-B), we have a heterogeneous data source for the semantic question answering. Generated data from OPC UA has no particular namespace definition unless we define it explicitly. However, the user-defined IRIs definition has drawbacks such as collision or non-extendibility. Linked data that has been instantly generated triples makes the structure complex so that two subjects of the list can collide with identical-defined IRIs. In this case, all namespaces are generated with http://www.example.org/ and "<unknown_namespace>". In Table II, answer return rate means that an answer takes round-trip time after prompting a question or keyword in the system. Querying style indicates the type of queries that we can enter and coverage shows the source of data that has been created. As for size parameter in Table II, the size of dataset that we generated from OPC UA Server has 19,687, which is 2 MB sized Turtle File. The Linkedfactory triples relate to hierarchical triples that have 70 triples as Turtle format and we test the question answering with manually generated questions through *Intel Core i7-2720QM CPU @ 2.20 GHz, 2201 MHz, and x64 based Windows 10 Pro*.

As a result, up-to-dateness supports update statement in SPARQL in a question answering system supports. Lastly, query formulation assistance displays to the end-users about the type of assistant modulethat is used in a question answering system.

B. *Result*

Evaluation criteria exhibits recall; accuracy, precision, and F1 Score of answers against semantic question answering system, as shown in Table III. General evaluation parameters for a restricted domain question answering are not only limited to responding to questions but also we can assess with speed, user interaction, querying style (keywords, browsing, spell checker, abbreviation recognition). In the following formulas, TP, TN, FN, and FP denotes true positive, true negative, false negative, false positive respectively.

$$Prediction = TP/(TP + FP) \tag{1}$$

$$Recall = TP/(TP + FP) \tag{2}$$

$$F1 - Score = 2x(PrecisionxRecall)/(Precision + Recall) \tag{3}$$

$$Accuracy = (TP + TN)/(TP + FP + FN + TN) \tag{4}$$

The precision (1) presents an expected answer that was correctly predicted against the total responses. F1 Score (3) is a balanced weight average between the Recall and Precision. The recall (2) is the proportion of correctly answered questions with respect to the number of questions. The accuracy of the model (4) explains the model that has a ratio of accurately predicted observation to the entire inspection.

Test questions were created with a combination of keywords and elements of sentences. Due to the domain restriction, the generation of test questionss has a goal that responds to the questions precisely ranging from keywords to complex natural input. The target data source is a mixed source that combines static and streaming data. In the appendix, readers can observe combinations of test questions to use for further improvements.

Evaluation Parameters	Properties
Answer Return Rate	QA against generated data from OPC UA - 23.25 seconds average QA against static query from RDF file of the eniLINK - 18.92 seconds average QA against dynamic query from streaming data - 17.48 seconds QA against Template Based Open-Domain Questions - 20.55 seconds
Querying Style	Keywords-Based Search and Question-Based Search
Coverage	The eniLINK data, the linked-factory streaming data
Size	Static data relatively small size Streaming data relatively large size
Up-to-dateness	No update statement provided by SPARQL
Query Formulation Assistance	Voice Input Recognition, Spell Checker

TABLE II: The semantic question answering evaluation criterion

Question Answering Parameters	Total Questions
True Positive	34
False Negative	13
False Positive	3
Precision	94.44%
Recall	72.34%
F1 Score	81.92%
Accuracy	68.00%

TABLE III: The Evaluation of the Question Answering (QA)

VII. Related Work

[Molla, Vicedo 2007] reviewed primary characteristics of question answering in restricted domain according to integration of domain-specific information [13]. [Molla, Vicedo 2007] defined main characteristics of question answering system over limited domains, e.g. circumscription of question answering, the complexity of question answering, and practical usage of question answering [13]. The authors have compared between open-domain and restricted-domain question answering by figuring out key points. [Molla, Vicedo 2007] offers four clear-cut aspects such as the *size of data*, *domain context*, *resources*, and *use of domain-specific resources*.

[Ferre 2012] published one of the detailed reports that express common pitfalls of natural language processing and essential points while consolidating SPARQL query and morphological definitions [14]. SQUALL is a solution for querying and updating RDF graphs by exploiting controlled natural language expressions that restrict grammar structures of a sentence in order to diminish complexities [14]. It has been grouped all substantial features of a morphological language, and the author pointed out what type of features in a natural language harnessed with regarding priorities and orders. The main contribution of SQUALL is categorizing ambiguities of natural expressions and how they turned an advantage out when using a controlled natural language [14].

[Biswas, Sharan and Malik 2004] proposed an architecture that extracts precise answers for a given question [15]. The authors described the module distinctly and defined the types of questions that can be asked to the question answering. The authors sketched a translation from their intermediate language to SPARQL to gain more accuracy with their system [14]. Template-based solutions were commented on for a restricted domain and open domain question answering systems. [Unger et. al. 2012] proposed a template-based solution that produces a SPARQL template, which it directly matches in the internal morphological features of the question [16].

Evaluation of a semantic question answering is still a cumbersome and hard problem. Lack of test questions that belong to a specific domain is one of the major problems. [Diekerma, Yilmazel and D. Liddy 2004] [17] offer different methodologies from an open-domain question answering while evaluating the restricted domain question answering. The authors specify the evaluation methodology as below [17]:

System Performance: Speed and availability **Answers**: Accuracy, Completeness **Display User Interface**: Querying styles, natural language queries, keywords, browsing, and the question formulation assistance (spell checker, abbreviation solver)

The authors stated that the *TREC* style question answering evaluation might not be suited for their restricted domain system so that user-based evaluation can be more viable in order to evaluate the system [17].

VIII. Discussion

In this chapter, we will discuss the significance of the findings relevant to the research problem being investigated. Taking our findings into consideration, we will summarize insights about problems.

First, RQ-1 and RQ-2 address distinct architectures for the use of semantic question answering. The proposal is implementing a service called KVIN that employs key-value mapping with windowed time-series data. The time-series data has been windowed with the size of data as well as the extent of the data size. Although the information structure is limited to be mapped onto Turtle triples, it can be useful for rapid prototyping. No cost will be arise from designing a new language onto SPARQL or overhead of instant linked data creation from streamed data.

Generating test datasets still is a problematic topic for the restricted domain question answering systems because there could be some bias. For instance, the test dataset for the information technology domain is not valuable for a manufacturing domain, which restricts the testability; however, we have used the parameters of referenced research

[17]. One of the findings is that the answer return rate is similar to template-based open-domain question answering [18]. If we want to get an answer relevant to *node id*, *node parent id*, *references*, and connected devices to OPC UA Servers, we need to convert the *Information Model of the OPC UA* to the linked data. Converting from the *root node* to the leaf nodes with namespaces of nodes would be enough to map onto <*subject-predicate-object*> triples. The semantic question answering should give precise answers for dynamic data and list the results of the answer against static data. Previous studies tried to solve the restricted domain question answering problem with template-based solutions by implementing a generic solution. Whereas, we perform a heuristic-based syntactic parsing to a smart factory domain. This heuristic-based approach does not guarantee optimal results in similar statements; however, it can give a high accuracy and F1-Score, as shown in Table III.

Showing the test results of the semantic question answering and question classification, this study guides researchers of Industry 4.0 regarding how to develop an advanced dialogue system. RQ-2 defines the main features of the semantic question answering in the smart factory domain, which consists of short-listed answering, deep and shallow parsing methods and the use of heterogeneous data sources. The display interface may reduce the time that a human operator spends on typing and correcting spelling mistakes so that the efficiency of query processing may increase.

Consequently, as referred to RQ-3, the generalization of a semantic question answering that works in a restricted domain to another one is not an easy solution. Although the algorithm and architecture generalizations are possible; however, the drawbacks are the particular keywords in unstructured data and streaming data. Moreover, this research contributes to the research circle with algorithms regarding test set generation and features of a semantic question answering to be used against heterogeneous sources.

The major problem of this proposal is that the question answering solely depends on the predicates of the data set defined by the smart factory. To solve the dependency problem, *subject-predicate-object* pairs can be recognized by deep learning methods with unstructured data. Correspondingly, the first finding was that the named-entity recognition had shown poor performance compared to the parsing method aspect of identifying noun and verb phrases.

The second finding is that complex paragraphs need a complicated mechanism such as co-reference resolution. Speed is another factor that we can infer when it comes to the customization of the semantic question answering. Accordingly, a technical operator or expert cannot get an answer from streaming data within the time-constraint of a mission-critical system.

The third finding is the serialization of the OPC UA can be a time-consuming task; moreover, there must be a control script to detect unaltered semantic triples. We propose the source code [15] so that one could recognize simulation data in OPC UA Server with a script to stave off the repercussion during serializing. The last finding is that the implementation of a generalized algorithm could degrade the precision of answers but increase the scalability at the various departments in a smart factory.

IX. Conclusion

The operator assistant system increases the productivity of human operators and experts in smart factories. In this paper, we have proposed an application for a restricted domain question answering that utilizes generated data from OPC Unified Architecture and streaming data. This application can reduce the total amount of time for searching through a large number of triples. The significant findings, that are, the proposed novel approach can be used effectively to create a supervisor tool for manufacturing technologies and a synthesized human operator assistant system, which caters to a robust architecture for the aimed platform. The proposed model reduces the complexity of the normalization process and employs state-of-the-art natural language understanding toolkits.

For future improvements, we plan to implement advanced semantic question answering that can be extended for time-constrained tasks such as soft-critical software systems. Furthermore, a question autocomplete system can be designed. Such a system would be efficient because it prevents the obligation of pattern or template-based question types. By scoring correctness of answers, the system can give better insight to the end-users. Lastly, named-entity recognition for smart factory and manufacturing lexicons can be added in order to eliminate a set of natural language processing steps shown in Figure 2. The more annotated data is inserted in the smart factory domain, the higher accurate reason induction, which is compatible with the question answering.

ACKNOWLEDGMENTS

This work is funded by the German Research (DFG) within the Research Training Group Role-Based Software Infrastructures for continuous-context-sensitive Systems (GRK 1907).

References

[1] R. Margaret and D. Daniel, “Definition of Smart Factory.” [Online]. Available: https://searcherp.techtarget.com/definition/smart-factory

[2] K. D. Thoben, S. A. Wiesner, and T. Wuest, ““Industrie 4.0” and smart manufacturing-a review of research issues and application examples,” 2017.

[3] C. Team, “What is the smart factory and its impact on manufacturing?” [Online]. Available: https://ottomotors.com/blog/what-is-the-smart-factorymanufacturing

[15] https://github.com/zointblackbriar/QuestionAnswering

[4] T. D. Oesterreich and F. Teuteberg, “Understanding the implications of digitisation and automation in the context of Industry 4.0: A triangulation approach and elements of a research agenda for the construction industry,” 2016.

[5] L. D. P. IWU and F., “eniLink.” [Online]. Available: http://platform.enilink.net/

[6] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus, “C-SPARQL,” in *Proc. 18th Int. Conf. World wide web - WWW ’09*, no. May 2014. New York, New York, USA: ACM Press, 2009, p. 1061. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1526709.1526856

[7] D. Jurafsky and J. H. Martin, *Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, third edit ed., Stanford University, 2019.

[8] J. Perkins, D. Chopra and N. Hardeniya, *Natural Language Processing : Python and NLTK*, 2016.

[9] P. Christen, “A comparison of personal name matching: Techniques and practical issues,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2006.

[10] F. IWU, “eniLINK,” 2020. [Online]. Available: http://platform.enilink.net/

[11] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, “The Stanford CoreNLP Natural Language Processing Toolkit,” 2015.

[12] F. IWU, “Linkedfactory Intro,” 2018. [Online]. Available: http://linkedfactory.iwu.fraunhofer.de/linkedfactory/view

[13] D. Mollá and J. L. Vicedo, “Question Answering in Restricted Domains: An Overview,” *Computational Linguistics*, vol. 33, no. 1, pp. 41–61, mar 2007. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/coli.2007.33.1.41

[14] S. Ferré, “SQUALL: A Controlled Natural Language for Querying and Updating RDF Graphs,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, pp. 11–25. [Online]. Available: http://link.springer.com/10.1007/978-3-642-32612-7{_}2

[15] P. Biswas, A. Sharan, and N. Malik, “A framework for restricted domain Question Answering System,” in *Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques, ICICT 2014*, 2014.

[16] C. Unger, L. Bühmann, J. Lehmann, A. C. N. Ngomo, D. Gerber, and P. Cimiano, “Template-based question answering over RDF data,” in *WWW’12 - Proceedings of the 21st Annual Conference on World Wide Web*, 2012.

[17] A. R. Diekerma and E. D. Liddy, “Evaluation of restricted domain Question- Answering systems,” *Center for Natural Language Processing*, 2004.

[18] Machinalis Group, “Quepy Question Answering.” [Online]. Available: http://quepy.machinalis.com/

Appendix

Question ID	Sample Questions	Precision	Recall
1	What do linkedfactory,heatmeter, and e3fabrik incorporate exactly ?	0.0	0.0
2	Provide me a combined result for IWU and e3sim	1.0	1.0
3	I want to know which one carries fofab ?	1.0	1.0
4	There is a member named fofab. Please give me all of its members	1.0	1.0
5	I am a customer of this company. Could you tell me please what the value of sensor1 of machine1 is ?	0.0	0.0
6	Could you tell me please what is the current value of sensor2 in machine2 ?	1.0	1.0
7	What POWERMETER holds ?	1.0	1.0
8	What does FOFAB incorporate ?	1.0	1.0
9	What does machine5 HOLD ?	1.0	1.0
10	What does gmx comprise ?	1.0	1.0
11	What comprises karobau?	1.0	1.0
12	System health for sensor2 in machine6	1.0	1.0
13	Tell me the health of system for sensor2 in machine1	0.0	0.0
14	Could you browse generated data ?	1.0	1.0
15	Give me all of the members of gmxspanen4	0.0	0.0
16	What holds coolingwater ?	1.0	1.0
17	What is the hierarchical structure of fofab ?	1.0	1.0
18	What contains IWU?	0.0	0.0
19	Could you give me the members in which contained by versuchsfeld ?	1.0	1.0
20	Could you give me the members in which linkedfactory has ?	1.0	1.0
21	What is the value of sensor1 in machine6 ?	1.0	1.0
22	What is the minimum that we can calculate for sensor1 of machine1 ?	1.0	1.0
23	What is the value of the maximum can be calculated by the sensor1 of machine1 ?	1.0	1.0
24	Could you tell me what the average for sensor3 in machine1 is ?	1.0	1.0
25	I need to learn an average value for sensor5 in machine2	0.0	0.0
26	What is the average of sensor3 in machine3 ?	1.0	1.0
27	Could you get me the references of nodes ?	1.0	1.0
28	Could you browse generated data ?	1.0	1.0
29	Is the E3-Sim member of linkedfactory ?	0.0	0.0
30	Could you take me all members of generated data ?	0.0	0.0
31	Give me all registered node id	1.0	1.0
32	I need to learn parent node id in generated data	0.5	0.5
33	Could you give me parent nodeID in the file of generated data ?	1.0	1.0
34	Give me all data blocks	1.0	1.0
35	Data blocks in generated OPC file	0.0	0.0
36	Give me the name of stations in generated data	0.0	0.0
37	All stations which are in generated data or new data	0.0	0.0
38	Registered node id	0.0	0.0
39	Who is Fofab ?	0.0	0.0
40	How is the system status for sensor1 in machine1?	1.0	1.0

TABLE IV: 40 Test Questions in order to test the application