TECHNISCHE UNIVERSITÄT
CHEMNITZ

Department of Computer Science

Data Management System Group

Master Thesis

Design and Implementation of a Web-Based Software for the OPC UA Communication Protocol Integrated into a Semantic Question Answering System

Orcun Oruc

Chemnitz, 26 February 2019

Examiner: Dr. Frank Seifert

Supervisor: MSc. Adrian Singer

Orcun Oruc, Design and Implementation of a Web-Application for OPC UA Protocol Integrated into a Semantic Question Answering, Master Thesis, Department of Computer Science Chemnitz University of Technology, 26 February 2019

Sperrvermerk

Diese Master Thesis enthält vertrauliche Daten der Fraunhofer IWU Institute. Eine Veröffentlichung dieser Arbeit, auch auszugsweise, ist ohne ausdrückliche Genehmigung der Fraunhofer IWU Institute nicht zulässig. Diese Arbeit darf nur den Korrektoren und dem Prüfungsausschuss zugänglich gemacht werden.

Abstract

Industry 4.0 has changed demands in terms of smart factories that promotes practices in manufacturing. In this context, smart factories have been created connected streaming data from the production system, which is hard to comment even by experts. Moreover, various smart factories and connected devices have developed a communication environment between devices and a production system so that a standardized communication shaped to solve the interoperability issue.

The OPC Unified Architecture works according to the server-client principle through which a server usually represents one or more machines with static and dynamic data models. Hence, this information model is not standardized; the internal structure can be changed individually for each server implementation. As a result, querying hierarchical structure of data or continuous data requires knowledge of the internal structure of manufacturing devices.

The service set of the OPC UA allows loading, querying and editing the data model, but relationships between nodes cannot be interpreted. This lack of semantic is recognized as one of the current challenges in "OPC Unified Architecture Pioneer of 4th Industrial (R) Evolution".

The goal of this thesis is to create a web-based application which is capable of loading, browsing and editing internal structures with regard to OPC UA Protocol to interpret

relationships between nodes and to implement a Semantic Question Answering connected to an internal Semantic Data Platform (eniLINK [1]) and the Dynamic Server. By doing so, this work provides an implementation to a human operator or experts using the system with limited knowledge of the inner OPC UA data model. In terms of Semantic Question Answering, main keywords which reside in Enilink Platform might be known to exploit semantic question answering, e.g. “linkedfactory” or “fofab”. In order to connect with the linked data, a Semantic Question Answering System performs on a different layer of the web application and comparing it to other solutions currently exist in the market. A semantic question answering provides information for an end-user, a human operator or an expert with natural questions convert into SPARQL in order to use linked data source of Fraunhofer IWU.

In the practical part of this thesis assessed the state-of-the-art toolkits and frameworks that are planning to use for implementation. In order to provide the best integration of question answering and OPC UA Web Service, Frontend and Backend technologies have been assessed with regard to the interoperability, scalability, loosely-coupled and synchronous/asynchronous development. A semantic question answering system is integrated into main web server software that can use linked data sources created by servers using OPC UA Protocol or real-time data source created by Fraunhofer IWU.

Acknowledgements

This research was supported by the Fraunhofer IWU Institute. I specially thank to my supervisor and colleague Msc. Adrian Singer. Without his guidance and helping, this thesis would not have been possible. I would like to express the deepest appreciation to my university supervisor Dr. Frank Seifert who provided insight and expertise that greatly assisted the research. I thank Diplom Inf. Ken Wenzel for assistance with LinkedFactory and Enilink Systems[1] and for his comments that remarkably improved the manuscript. Finally, I am deeply grateful to my family who supported whatever it costs through all my life.

Table of Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

NLP	Natural Language Processing
HMI	Human Machine Interface
REST	Representational State Transfer
JSON	JavaScript Object Notation
JWT	JSON Web Token
OPC UA	Open Platform Communication Unified Architecture
Fraunhofer IWU	Fraunhofer Institute for Machine Tools and Forming Technology
VDMA	Mechanical Engineering Industry Association
MVP	Minimum Viable Product
RFC	Request For Comments
RDF	Resource Description Framework
SOA	Service Oriented Architecture
OLE	Object Linking and Embedding
SDK	Software Development Kit
CLR	Common Language Runtime
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
W3C	World Wide Web Consortium

XML	Extensible Mark-up Language
DOM	Document Object Model
MVC	Model-View-Controller Pattern
SDK	Software Development Kit

1 Introduction

This study examines the use and implementation of web-based software for OPC – UA Protocol and a question answering system regarding the web-based software. To assess a question answering system, the source of data should be existed and examined as well. The web-based application for OPC UA Protocol Integrated to the Semantic Question Answering meets some requirements of smart factories initiated by Industry 4.0.

The rest of the chapter is organized as follows. Section 1.1 provides motivation for the research that outlines a case study. Importantly, this section builds a background that determines the importance of the research and relevant approaches. However, Section 2 will analyze previous studies in order to give a concrete idea to readers. Section 1.2 provides a problem statement, explaining the general obstacles of the research area. Section 1.3 briefly reviews the requirements, research questions, focus of the research and methods of research. Section 1.5 explains planned sections of the rest of the work.

1.1 Motivation

Industrial production characterized by the increasing interconnection of machines and devices (e.g. Industrial Internet of Things, IIoT). The IEC 62541 (OPC Unified Architecture) protocol is one of the most important representatives in the heterogeneous view of available communication protocols, which recommended by the Mechanical Engineering Industry Association (VDMA) as preferable M2M communication. Manufacturing technologies and systems that connected with them have evolved drastically according to the business world's requirements. These requirements such as interconnection elements of a factory, a distribution source of data, a continuous process planning etc. require a concept that enacted by the German Government as "Industrie 4.0" [2].

In light of Industry 4.0, it embodies all of the control networks and the network of the Cyber-Physical System that is controlled and monitored by the predefined algorithm. Industry 4.0 as an initial force to take the manufacturing industry forward through a digitalization by using various new technologies. Human Machine Interaction (HMI) provides interoperability of Industry 4.0; hence, the production system of factories

chiefly leverages a set of technologies such as Big Data, Internet of Things and Semantically Linked Data. In Fraunhofer Institute for Machine Tools and Forming Technology (Fraunhofer IWU), the department of digitalization in production aims to increase the efficiency of output processes of all manufacturing devices by implementing a Semantic Linked Data relevant to different factory processes.

To obtain a better quality in various manufacturing processes, we can benefit from a state-of-the-art protocol named OPC Unified Architecture (OPC UA) Protocol to fetch data from real-time environments such as sensors and save it into extensible markup language to convert into a semantic data. To facilitate interaction between end-users, human operators and machines, we present a question answering system that takes as a natural language input of statements returning a comprehensive answer. The issues that this thesis solves are important the aspect of integration a platform independent, scalable, flexible and robust solution for a Human Machine Interface in the context of limited architectural information aspect of the user. By using the web-based software, the thesis shows a clear indicative solving step of restricted-domain question answering to the other researchers by means of an artificial intelligence method. In addition, pre-existed solutions of the service-oriented protocol (OPC UA) evaluated and this work provides a better solution to us with an architectural pattern.

The difficulty of the area is a data scarcity issue in terms of a question answering system. In order to overcome this issue, data is used that created by the manufacturing devices of Fraunhofer IWU and quasi OPC UA Server of Fraunhofer IWU. The findings of this study suggest that the company provides streamed data or semantic data that can be collected from sensors and actuators to send a query against semantic endpoints. Human operators have difficulty in understanding of data source when they call for achieving daily tasks. Both OPC UA Web-based Application and Integrated the Semantic Question Answering envision that a human operator carries out a task by making less effort. Evaluation criterions are used which are different from open-domain question answering. The criterion numbers improved such as accuracy, user interface suitability, question formulations etc. to prove key points of a semantic-question answering the viewers. Not also, a question answering is evaluated with criterions, but also OPC UA web-based software and semantic representation tools are given by assessing with criterion regarding the thesis.

The results of this thesis exemplify the aggregated web-based application of OPC UA with a semantic question answering in the area of smart factories. Previous studies show

document-centered or keyword-based question answering system in a restricted-domain can be existed, and yet the gap between semantic data and question answering is still opened topic as shown in the overview of studies.

1.2 Problem Statement

Communication Protocols evolves continuously in accordance with interoperability and sustainability. Automation industry requires a next-generation standard such as OPC UA, which uses a scalable platform, multiple security models, multiple transport layers and an advanced information model to allow the smallest dedicated controller [1]. Real-time sensors and actuators' data should present in a semantic environment. By our research, we will investigate better applicability in Human-Machine Interface by using the OPC UA Platform with unstructured and structured data. Not only we will create a solution of web-based software, which works under OPC UA Protocol, but also we create a solution a question answering system communicates to an end user who knows limited information about a company's data structure. An end-user or a human operator can use the question answering that gives restricted information regarding the manufacturing system of Fraunhofer IWU. Any knowledge about a system architecture is required. A noteworthy research gap is how to transform OPC UA Address Model of Servers from unstructured information into structured information. Furthermore, it is not clear how to integrate a question answering system that harnesses semantic data resources from natural language questions. Many solutions exist in the research world that shown in the chapter "Overview of the Studies", however, Restricted Data Resource such as Fraunhofer IWU's Domain can partially exploit the researches that have defined. Thus, our priority produces a solution by utilizing a company-specific data resource. Due to limited structured data in Fraunhofer IWU, we also evaluate practical algorithms and give a comparison in the conclusion and evaluation chapter. The experience of implementing is helping other researchers who want to enhance the scope of their works in a suitable manner. OPC UA Web Service will be evaluated with the following parameters end-to-end productivity, interoperability, version ability, testability, and learnability. Integrated Question Answering will be evaluated questions and their answers. Due to the target-domain of this research, the question datasets like SQuAD, TREC QA or QALD Dataset cannot be used to evaluate restricted domain question answering. Working papers which studied evaluation of restricted domain question answering states that act of

ascertaining the results should set apart with dissimilar criterions then open-domain question answering [3].

1.3 Objectives and Scope

The objective of the thesis is to evaluate the viability of web technologies for industrial communication and suitability of semantic technologies to perform an assistance software.

This thesis reviews past literature to indicate the gap in knowledge and possible limitation while linking of an OPC UA Information Model with Semantic Web technologies. This work evaluates that allows querying a piece of structured information from an OPC UA Server and bringing time-series values from a remote server named EniLink [1] with natural language commands.

As a part of this work, we need to clarify the current state of research on scientific publications regarding meaningful researches that guide clearly in a literature review. Within the context of computer science, this work performs with informative theories and detailed implementations for a better understanding. The main objective of this work to assess pertinence of a web service for OPC Unified Architecture (OPC UA) Protocol and connecting to a Semantic Data Source that can be obtained from a server of OPC UA placed a device such as Siemens Step S7 or time-series data generated by eniLINK Platform. To focus is finding a way to connect a web service to a question answering system where a user can also send a natural language query even if a user does not have any prior knowledge about the architecture of the protocol or semantically linked data. This topic is required to evaluate how to implement lowest level communication with the OPC UA Open Source SDK. Since a web-based software that properly works in common operating systems, a semantic data source might be created from an OPC UA Server to evaluate compatibility. In spite of simulated data in the OPC UA Protocol, we utilize static and real-time eniLINK Platform's Data [1]. The overall goal is not to develop all-in-one web service suite, which can be integrated to Fraunhofer IWU Human Machine Interaction (HMI) System, rather showing a Minimum Viable Product (MVP) that might be used for the end user or experts of Fraunhofer IWU who benefits with results of real-time or static data.

Initially, we should define the requirements of our project. These requirements provide results upon which the goals that we want to achieve. We will reach our requirements with the following research questions:

1) How can a semantic question answering integrate to OPC UA Web-based software?

2) Is the research able to establish a new foundation for an OPC UA REST Integration and a Question Answering System?

3) What are the requirements of OPC UA Web-based software integrated into the semantic question answering?

4) How does assess and compare an OPC UA Web-based software integrated into the semantic question answering?

5) What are the characteristics associated with the architecture of general application?

6) What are the key components of the approach and how did the research contribute to the research area?

//Focus on the research problem, research objective, research approach, and research delivery.

Requirements 1: Implementation of a robust web-based software to employ human to machine and machine-to-machine communication technologies.

Firstly, the architecture of the work should dissect OPC UA Unified Architecture and OPC Legacy Architecture concisely. The international specifications defined by RFC and other sources define OPC UA Unified Architecture's members. When analyzed, Information Model, Address Space, Discovery and Subscription Services are the most important specifications, which need to be used by the proposed software solution.

Requirement 2: Semantic composition with linked data from the streamed data of a smart factory and OPC UA protocol and serialization mechanism.

Web-based Software mainly is broken down into two parts named REST – Based and SOA Based Architectures. Conceptual and practical differences should be dissected

with this requirement and this work should embrace drawbacks both the above-mentioned methods and advantages of them. Evaluation criteria will be created mainly depends on load testing, unit testing, and REST API functionality.

Requirement 3: A design and implementation of a question answering system that employs linked data provided by the semantic composition and evaluation of the semantic question answering.

In this requirement, over the last couple of years, there are not many types of research have been done. OPC UA Address Space Representation used to model a server structure rather than exporting into a semantic data source. In that case, semantic data can be created from a modeling language that represents an internal structure of any server. OPC UA Address Space Model mainly used for modeling in other address spaces; besides, an address space model can be converted its structure into a semantic language with all necessary details with sensors, actuators, and internal structures.

Requirement 4: A literature review of previous studies that comprises of requirement 1, 2, and 3.

The thesis breaks down into two parts, which connected one another. The first part is conducted assessment and drawing conclusions of existing OPC-UA Web Service, which has studied, by other teams. The second part we will examine the gap between OPC UA Address Space transforming into Semantic Data. Regarding research evaluates that the usability of Address Space Mark-up Language in the context of the Semantic Data Model.

Thirdly, we pursue a conclusion about the question answering system applied on restricted-domain resources with regard to generated source from OPC UA Server or eniLINK Platform [1].

1.4 Structure of the work

//This section will be defined again

This research organized as follows. Chapter 1 presents motivation, problem definition, scope, and related works. In Chapter 2, we organized the structure with Industry 4.0 and OPC Unified Architecture. In Chapter 3, we will explain our development technologies in Practical Development. In Chapter 4, Implementation Details of OPC UA web-based

software will be introduced and technical definitions will be enlightened to viewers. Chapter 6 conducts an experimental development phase for a web-based software both OPC UA Protocol and Semantic Question Answering by comparing existing frameworks of Back-End and Front-End. Chapter 7 evaluates the application performance and compares it to the existing solutions presented by previous chapters. Chapter 8 identifies a problem about how to represent semantic data which is extracted from OPC UA Address Space. Moreover, this chapter gives succinct details about the process of converting and transforming to a better format in order to use with the Question Answering System.

2 State of the Art

2.1 Review of Existing Researches

This thesis requires conducting a literature review providing novel methods a web service of OPC UA, serialization of OPC UA Data and Semantic Question Answering. This literature review comprises the state-of-art development of OPC Unified Architecture Applications, Time Series Data Mapping into Semantic Data, Address Space Representation, types of Semantic Question Answering and regarding developed applications aspect of various types of Question Answering

2.1.1 OPC Unified Architecture Concept and Existing Applications

OPC Unified Architecture (OPC UA) was developed by taking into consideration the drawbacks of the traditional OPC Classic Platform. Microsoft conceptualized OPC Framework with Component Object Model (COM) [4]. However, OPC Classic did not purpose for connecting end-user devices to the underlying protocol. To remedy this drawback, OPC UA Platform, which is a platform-independent service-oriented architecture that integrates all the functionality of the individual OPC Classic into one extensible framework was standardized and released in 2008 [5]. In the thesis, we will discuss the OPC UA Architecture in conjunction with the definition of layers of a schema, their usages and impact of our conclusion part.

[Salvatore Cavalieri, Marco Guiseppe Salafia, Marco Stefano Scroppo] [6] make an effort to enhance interoperability with web technologies to comply with Web Service using Representational State Transfer mechanism. The system purposed for the end user who does not know technical information about OPC UA Protocol and one can use by means of a web service which provides a token-based authentication[7]. The architecture of system [8] sets apart from other solution platform independence, loosely-coupling and monitoring of message-broker protocol such as MQTT, AMQP, and SignalR. According to the authors of the paper, main differences between the proposal and the solutions available in the literature is that all these solutions require, on the Front-End side, the knowledge of communication services and the data modeling provided by OPC UA Standard [6]. Moreover, by holding a variety of username-password pair, or X509 certificates, this platform assures granting of the services. Services split up in a distinct

manner such as SecureAccess, GetDataSources, ReadInfo, WriteInfo, and Monitoring. [Tatu Paronen] [8] states that examines the requirements for the generic client and concerned with the specific technology choices, architecture, interfaces between different components, and other technical decisions related to the implementation [8]. The research examined OPC UA connectivity from a web browser and organized as pre-rendering by a user interfaces, exposing with a web API or applying native communication stack for a particular programming language [8]. The author organized the structure with a tiered architecture such as Presentation Layer and Service Layer that consists of Server Connections, Address-Space Browser, Subscriptions, and Monitored Item Services [8]. It has composed with Service Layer and OPC UA Servers Layer concerning low-level communication as well.

2.1.2 Streamed Data Mapping into Semantic Data

Previous studies mostly defined Semantic Representation as a challenge that is supposed to map from the data based on Time Series or Internet of Things defined to linked data forms. Raw sensor data is useless unless without being properly annotated. Real-time data annotation is another problem when annotating data at the same time processing it. [Xiang Su Et al.] offers a mark-up language for representing device parameters and measurements [9]. Standardization is a solution for representing data in Semantic World; however, a mark-up language is challenging to define because different vendors use different standardization methods. [Xiang Su Et al.] defines two main rules to implement a semantic annotation, which is transformability to multiple RDF sources such as N3, Turtle and automatic assignment of a namespace to be defined on the Internet of Thing applications [9]. (Xiang Wang, 2015) defined semantic sensor networks characteristics that overlap the requirement of this study. (Xiang Wang, 2015) emphasized the sensor data is to be annotated is improved interoperability, the context information of time and space for sensor data should be integrated and support multi-type sensor data [10]. One major drawback of this approach is that the overall system should comply with a new standardized semantic sensor network. However, thanks to the data integration layer of this study [10], the proposed system can be detached easily from the sensor data source layer. The study points out that a semantic sensor network can create a three-tier architecture with a loosely-coupled and coarse-grained way in which can incorporate the application layer [10]. Establishing a way to extract automatically from unstructured time series data into structured data is a challenging problem. (Katrin Rodriguez Llanes, 2016) states that the real-time approaches of linked data suffer from main limitations which are [11]:

1) Triple storage cannot efficiently handle high update rates

2) Numeric reading has performance issues with complex SPARQL queries.

3) Extracting sensor data into triples are inefficient

As being a survey paper, (Katrin Rodriguez Llanes Et al., 2016) categorized real time series into linked data with a selection of ontologies, defining the mapping language, selection of continuous queries languages, choosing related datasets in Linked Open Data cloud and creating data linkages [11]. Each chapter have given a definition and current research in the market, and it can be summarized as below:

Selection of ontologies: Ontology selection is a crucial step to perform a continuous linked data stream from time series. Every platform has own specifics and it should be handled with proper RDF datasets such as OWL, Turtle or N3. This thesis uses mostly Turtle datasets. Lack of scalability from one semantic data source to another, platforms should use standard semantic dataset and annotations. (Katrin Rodriguez Llanes Et al., 2016) offers to use Semantic Sensor Networks that can describe capabilities, measurements, and resultant from sensor and actuators [11].

Defining the mapping language: To convert sensor data from time series into RDF needs extra layer by customizing mappings from relational or non-relational databases to RDF datasets [11]. (Katrin Rodriguez Llanes Et al.., 2016) demonstrates two approaches which are: R2O (Calbimnonte Et al.) and SASML (Zhang et. al. , 2015) [11]. In the context of R2O[1], it is an extension of S2O is to utilize reifying from Relational or Non-Relational Objects to RDF. A platform works in real time data series should have a mapping layer in order to send a SPARQL request.

Selection of continuous queries language: The authors stated that languages such as SPARQL are designed to execute RDF triples in a static way, however SPARQL query has no effect on continuously linked RDF triples so that a new RDF Stream Processors are implementing in the market by [Barbieri Et al., 2009], [Calbimonte Et al. 2011], [Anicic and Fodor, 2011] and [Phuoc, 2013] called C-SPARQL, Event Processing-SPARQL, Continuous Query Execution over Linked Stream (CQELS) respectively [11].

[1] http://oa.upm.es/5678/1/Workshop14.SWDB2004.pdf

Large scale federated linked continuous data extractor may provide a RESTful architecture exposing necessary annotations and semantic data. Our scope is implementing a web service so that this thesis examines which studies were conducted by (Henning Hasemann Et al., 2018) and (Heiko Müller Et al., 2013). (Henning Hasemann et. al. , 2018) proposed an RDF tuple store named Wiselib that attaches into sensors to collect data by means of RESTful architecture that can connect to Semantic Data [12]. The Wiselib on the lowest level, it uses a set of protocol that a sensor can understand at the same level. On the highest level, it uses HTTP protocol to understand semantic web documents as a proxy server. As an extra feature, the tuple store can behave as a SPARQL endpoint by basic query parameters such as query and insert[12].

// From Restful to SPARQL: A Case Study on Generating Semantic Sensor Data will
// be examined

(Heiko Müller et. al., 2013) created an ontology which can be used with cloud frameworks. After created ontology, the authors decided to create a transforming tool by using a set of JSON documents to convert into RDF documents. At the final stage, generated RDF graph with links to the resource was enhanced in order to create purpose-built functions in linked open data cloud [13]. The main goal of the work semantically augments the Sensor Cloud that enables the automated generation of RDF triples from sensor data [13].

2.1.3 OPC UA Address Space Representation in Semantic Data (Static Representation)

There is a research gap between OPC UA Web Service and Semantic Question Answering. To elucidate the research about OPC UA Information Model mapped into Semantic Data Link, it appears that researchers' communities have not conducted enough surveys, researches or statements. [Eero Laukkanen] [14] has conducted research about source code generations from OPC UA Servers to create a Source Code Generator in Java by using instances of metamodels and templates with metadata [6]. [Badarinath Katti, Christiane Plociennik and Michael Schweitzer] [15] emphasized the absence of standardization of information model for machines and this can lead us to each vendor has produced on their own standardization. The authors introduced a concept of a semantic markup language called OWL-S to the OPC-UA protocol particularly [7]. On the subject of research, a remarkable tool that has published by Python-FreeOpcUA used to

create an extensible markup language of its address space [16]. Researchers of Fraunhofer IOSB implemented a tool which supports OPC UA Address Space Extensible Markup Language Export and Import to substantiate Computer Aided Design in Manufacturing Execution System (MES) [17] This is one of the closest research which complies with our purpose except that they intend to create a semantic data format such as RDF/XML, RDF or Turtle. Although they published an article with definite goals, the research states limited usage with OPC UA Server to another. Rather, it does not meet requirements concerning a semantically represented data.

2.1.4 Open-Domain, Closed-Domain, and Semantic Question Answering System

Regarding Semantic Question Answering, researchers mostly focus on algorithms on how to transform from a natural language query to Structured Query Language (SQL) and SPARQL Protocol and Resource Description Framework Query Language (SPARQL). Principally, a question answering system is a system to answer a question by human interaction with respect to information retrieval and natural language processing theories. Two types of main question answering systems were briefly introduced in this work, which are:

Open-Domain Question Answering: The question can be asked to a general type of data sources such as DBpedia, Freebase, and Wikidata. Not only a specific topic can be asked but also a user may ask in any type of question so as to get an answer from a data source.

Closed-Domain Question Answering: A user can ask a question against a restricted type of data source which has defined in which a commercial domain resides. A user may not ask all type of questions so as to get an answer from a data source.

[Victor Zhang, Caiming Xiong, Richard Socher] [18] created a model to leverage the structure of SQL queries to significantly reduce the output space of generated queries by using reinforcement learning method [6]. SQL Queries used for a relational data source but the paper significantly important to understand how to transform a natural language query into a query language. Their algorithm Seq2SQL [18] requires a large number of question schemas, which need to be retrieved by the author. Seq2SQL takes input as a question and columns of a table. It generates the corresponding SQL query during a training phase, thus it executes against a database. The output space is representing via

a Softmax Function, which is a way of representation in a probabilistic view of the Machine Learning. However, the author [6] stated that the output space should be limited in order to do so. For the sake of brevity, Seq2SQL produces three basic structure for SQL query such as Aggregation classifier, SELECT column pointer and WHERE clause pointer. Aggregated input encoding such as COUNT, MIN, and MAX is applied by Softmax Function to obtain the distribution over the set of possible aggregation operations [6]. The authors have tested their system WikiSQL which is a large crowd-sourced dataset for developing natural language interfaces for relational databases [18]. Towards solving the problem, the SQLNet Algorithm is an approach to employ a sequence-to-sequence style model. In particular, [Xiaojun Xum Chang Lie, Dawn Song] employ a sketch-based approach where the sketch contains a dependency graph so that one prediction per-forms by taking into consideration only the previous predictions that it depends on [19]. Main contributions of the SQLNet Algorithm express as below:

Avoiding "order-matters" problems in a sequence-to-sequence model and thus avoids the necessity to employ a reinforcement learning algorithm [19].

A novel attention structure is called column attention, and show that this helps to further boost the performance over a raw sequence-to-set model [19]. To handle with column names, Stanford CoreNLP Tokenizer used for parsing the sentence [20]. Shortly, each token represents a vector and inserts into bi-directional Long Short-Term Memory. [F.F. Luz, M. Finger] used an LSTM encoder-decoder model capable of encoding natural language (English) and decoding query language (SPARQL) [21]. The algorithm uses a new type of encoder-decoder model that implements an alignment model in a feedforward neural network, which is concurrently trained with all of the components [21]. In the approach of [F.F. Luz, M.Finfer] is used to learn word vector representation and an LSTM neural network is used to encode natural language sentences and decode SPARQL Query [21]. The main contribution in the algorithm part that they used a novel approach in Recurrent Neural Network (RNN) with Neural Attention. Mainly, the researchers endeavored to solve the long-range dependency problem and the approach broken into two steps, which are described as follows:

- The first step is to find a good vector representation for the target language lexicon.

- The second step, [F.F. Luz, M. Finger] concerned with implementing and finding the settings so that the architecture can translate from natural language to SPARQL [21].

Most question answering systems translate questions into triples that are matched against the similarity metric. However, in many cases, triples do not represent a faithful representation of the semantic structure of the natural language question, with the result that complicated queries may not be answered [22]. [Christina Unger, Lorenz Bühnmann, Jens Lehnmann, Axel Cyrille Ngonga, Daniel Geber, Philipp Cimano] adopt the parsing and meaning construction mechanism of a question answering system which consists of two algorithms used Lexicalized Tree Adjoining Grammar and Underspecified Discourse Representation Theory respectively [22]. The authors used Stanford Part of Speech (POS) Tagger on the fly while parsing to sentences. Nouns entities are noun phrases and are usually modeled as resources, thus a lexical entry is built comprising a syntactic noun phrase representation together with a corresponding semantic representation containing resource slot [22]. Both algorithms serves one single purpose to identify light verbs (to be, to have and imperatives like give me), question words (what, which, how many, when, where) and other determiners (some, all, at least, more/less than, the most/least), together with negation words, coordination and the like [22]. [Christina Unger et. al.] primarily contributed to this paper as below:

- Pure comparisons categorized in natural language specific tools and broken into correct stages while creating a SPARQL Query.
- In particular, an active learning method was implemented for end users to give feedback on the presented query results [22].
- Clearly defined goals of a question answering system and designed for a large-scale heterogeneous knowledge base.

[Tommaso Soru, Edgard Marx, Diego Moussalle, Gustavo Publio] have implemented an architecture Neural Machine Translation (NMT) approach that relies on three main components: a generator, a learner and an interpreter [23]. The generator takes inputs to create a query template. A query template is an alignment between a natural language query and its corresponding SPARQL Query [23]. At the final step, the learner obtains a natural language question to encode SPARQL queries. The authors have published a preliminary result in accordance with epochs on the training phase. They planned to

address current limitations by investigating how to generate domain-independent templates and minimize the burden on the end user [23]. [Shanthi Palaniappan, U.K. Sridevi, Jayasudha Subburaj] focused on a question answering system by improving the semantic similarity with JavaScript Object Notation (JSON) – Linked Data (JSON-LD) [24]. This work aims at a different type of questions with different patterns which have to be matched with the ontology tree structure [24]. Chiefly, they specified an architecture in a clear manner to reach their goals. The main drawback of this architecture of the ontology mapping part was not outlined. As for the main contributions of this paper, it can be taken into consideration that the authors provide a clear-cut step that entirely can carry out with a specific semantic data source such as JSON-LD.

Hidden Markov Model (HMM) is discovered by [Cristina Giannone, Valentina Bellomaria and Roberto Basili] in conjunction with QALD-3 Question Answering Contest [25]. The authors clearly indicate how to parse an RDF Data Source with the HMM method in order to create a precise SPARQL Query. The architecture comprises HMM Initialization, Modeling, Decoding and Query Computation [25]. They have clearly indicated the combinations of sentences with dependency parse tree how impacted from changes of an order of items. Mainly, HMM Methods used for assigning properties of DBpedia source into their states and compilation of SPARQL queries [25].

[Diego Molla et. al] overview a main characteristic of question answering in restricted domains is the integration of domain-specific information that is either developed for question answering or disclosed for other purposes [26]. [Diego Molla, Jose Luis Vicedo] defines main characteristics of question answering system over limited domains as below [27]:

- It should be circumscribed
- It should be complex
- It should be practical

The authors have compared between open-domain and restricted-domain question answering by figuring out key points. According to their paper, they have defined three clear-cut subjects, which are:

- The size of data
- Domain Context

- Resources
- Use of Domain-Specific Resources

[Sebastian Ferre] has published one of the detailed reports that express common pitfalls of natural language processing, essential points while consolidating SPARQL query and morphological definitions [28]. SQUALL is a solution for querying and updating RDF graphs by exploiting a controlled natural language which restricts grammar structures of a sentence in order to diminish complexities [28]. It has grouped all substantial features of a morphological language and pointed out what type of features in a natural language harnesses with regarding priorities and orders. The main contribution of SQUALL is categorizing ambiguities of natural languages and how turned out an advantage when using a controlled natural language [28]. The authors sketched a translation from their intermediate language to SPARQL to gain more accuracy with their system [28].

2.1.5 Existing Applications in Question Answering

In this section, existing applications are surveyed without distinguishing of open-domain, closed-domain or restricted domain. AquaLog is a portable question-answering system that takes queries to express in natural language and an ontology as input, and returns answers drawn from one or more knowledge bases (KB). By using the NLP Platform, [Vanessa Lopez Garcia, Enrico Motta, Victoria Uren] aimed to choose an ontology and then ask queries with respect to its universe of discourse [29]. AquaLog uses a Relation Similarity Service that is purposed for dealing with the various sources of ambiguity. The authors give explicitly a comparison between open-domain question answering and AquaLog. The main contribution of AquaLog is capable of learning the user's jargon in order to improve the quality of the result [29]. The approach of NLP-Reduce is simple and does not use any complex natural language processing technologies [30]. Principally, it utilizes synonym expansion and stemming techniques. NLP-Reduce contains a user interface, query generator, and ontology mapper to send a query against open-domain resources. In the preliminary evaluation part, the more NLP query is trained by NLP-Reduce, in order to get precise results from associated to a geographical dataset. [Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham] developed a system named FREyA, which combines syntactic parsing with the knowledge encoded in ontologies in order to reduce the customization effect [31]. In the research world,

main contributions with FREyA can be broken into improving understanding of the question's semantic meaning, providing the concise answer to the user's question and communication the system's interpretation of the query to the user [31]. As the authors stated, they evaluated the FREyA with Mooney Geoquery Dataset[2] in accordance with correctness, ranked suggestions and learning mechanism [31]. Bio2RDF is creating a network of coherent linked data and linking together with semantic life science resources [32]. Bio2RDF initiates a full-text search in the graph resources, in that case, the server returns a graph containing the searched literally with matched resources [32]. The main contribution in the paper, Bio2RDF offers a reverse link service to determine whether a Uniform Resource Identifier (URI) has used in other sources [32]. This feature is significantly important while creating a SPARQL re-quest with prefixes that use URI tags. One of the brilliant research was published by [Vanessa Lopez, Enrico Motta] and they implemented a solution named PowerAqua [33]. PowerAqua follows the system architecture of AquaLog, and address its main limitation to leverage future properties in order to develop a better mechanism. PowerAqua's paper explains that all bottlenecks of the architecture of AquaLog, drawbacks which should be remedied and how to use the existed architecture of AquaLog in PowerAqua. The main contribution of the paper provides information on how to implement a Question Answering System against multi-ontologies based on In-formation Retrieval Sources. GFMed is another Biomedical Question Answering System that we are going to discuss contributions, benefits, and drawbacks. GFMed used to create a SPARQL request from a natural language with a specific English-Biology Lexion by concreting SPARQL Grammar with Sparql Lexicon [34]. English to SPARQL Algorithm has been clearly elaborated and supplied the question answering with a concrete SPARQL grammar built on top of a library [34]. The authors of the paper evaluate results in conjunction with Processed, Right, Partially, Recall, Precision and F-measure. Multilingualism (syntax, lexicon, and inflections in 36 languages [34]) is one of the contributions of the algorithm that taken into consideration. Pythia compositionally constructs meaning representations using a vocabulary aligned to the vocabulary of a given ontology [35]. The primary contribution of Pythia is generating a grammar from a given ontology (ontology-based grammar) to enrich a natural language query that shapes a formal query [35].

[2] http://www.cs.utexas.edu/users/ml/nldata/geoquery.html

2.2 Chapter Discussion

Chapter 2 break into five sections to easily investigate previous studies that have written with advantages, disadvantages, and contributions.

3 Smart Factories and OPC Unified Architecture

3.1 Towards Smart Factories

The definition of smart factories has evolved over the past few years. In the present study, a smart factory has defined an aspect of boosted technologies named Industry 4.0 and Human-Machine Interface. Impact of manufacturing development impacted economic growth over the last decade in Germany. Continuously improvement of Industry 4.0 brought the researchers to find cutting-edge technologies such as Question Answering System, Manufacturing Augmented Reality etc. The key aspect of Smart Factories can be list as follows: Towards Smart Factories, Industry 4.0 and Human Machine Interface in Smart Factories. Within the key aspects, this study informs the readers how it contributed to the Industry 4.0 area and what will be the benefits when used by Smart Factories.

A smart factory is a highly digitized and connected production facility that relies on smart manufacturing [36]. This concept one of the key outcome 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 making data collection possible. Additionally, sensors enable the monitoring of specific processes throughout the factory that increases awareness of what is happening on distinct levels [37].

3.2 Industry 4.0

The development of Industry 4.0 has a big influence on the manufacturing industry. In the era of smart manufacturing systems, Industry 4.0 is a necessity that should standardize all communication structures in smart factories. The primary objective of Industry 4.0 makes the manufacturing technologies of factories more intelligent, optimizing the chain of processes and enhancing capabilities of communication one to another. Possible solutions emerged through all manufacturing processes in Fraunhofer IWU and any other smart factories. Furthermore, the development towards an Industry 4.0 provides a vast of opportunities for realizing sustainable manufacturing using big data analytics in the context with Information and Communication Technologies [38].

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 [39]. Industry 4.0 is a new concept comes out a necessity, which is co-operating between machine and people. Hence, it is necessary to digitally integrate the value chain by using cyber-physical systems is required [39]. A cyber-physical system describes the relationship between humans and a Cyber-Physical System, which is again divided into a physical component by separating virtually from each other[40]. Taken as a whole, physical components and their virtual representations should standardize from the bottom to the top. The Cyber-Physical System embraces complex networking, integration of embedded systems and application systems, enabled by Human Machine Interface [2].

3.3 Human Machine Interface in Smart Factories

Manufacturing is one of the key areas that should communicate with humans clearly to increase the overall efficiency of a factory. Industry 4.0 could be a process increasingly complex, which should stay in touch with people properly to manage task efficiently. A Human Machine Interface is a term utilizing for associating any device to a machine, which can be controlled by a smartphone, terminal device or monitoring device. The input can be taken via keypad, keyboard or touch screen, however, the new concept of input is giving as voice control or natural language queries. A well-designed HMI solution not only increases the productivity of the human operator but also provides the system control or maintenance of a machine [41]. HMI Systems are responsible for interaction between a human operator and a machine. A good example of the importance of interaction is monitoring disturbances in HMI Systems Alarms in an assembly line shows a human operator which part should be intervened to prevent fatal damages. The web-based solution has an advantage over developing a smartphone application or tablet application in accordance with scalability. Since a mobile device can reach to a server, the HMI system can connect all device according to the web server's configuration. Hence, the system may enhance user experience with HMI legacy devices in order to interact with the data of industrial processes [42].

With the improvement in the OPC Legacy Standard, the Industrial World achieved the interoperability between heterogeneous devices at the communications level, regardless of the manufacturer [42]. Question Answering System increases the capability of transforming query languages. Semantic structured or relational data using to show result by means of a particular query language such as SQL, SPARQL. Owing to the difficulty of

learning query language, the HMI system should provide a mechanism for entering input as a natural language or a quasi-natural language. The role of human operators or experts are facilitated by language queries, in addition, it diminishes time-squandering error that occurred in elements of a machine in an efficient manner. Task-specific Human-Machine Interfaces provides a set of information with specific user interfaces such as condition monitoring, energy management, predictive maintenance or diagnosis [43]. In our case, experts may employ the average value of a specific sensor that resides in a machine to predict future maintenance or repair. The system can also provide an error situation with a threshold value when querying into time series data. Because of domain-dependent data, a question answering system complies with the factory specific data. The data may contain many specialized terms that experts use a keyword or plain text to search for an item related to a machine.

3.4 Overview of Open Platform Communication Unified Architecture

The following section gives brief information about the historical development process of OPC Unified Architecture. On the path of development, OLE OPC was formerly known as OLE for Process Control has been widely dispersed at the industrial scale. The term Object Linking and Embedding for Process Control traced back to initial development that was founded by Microsoft in order to communicate objects with Component Object Model (COM). By introducing the following chapter, the thesis provides comparative information on each technological steps in terms of drawbacks and benefits. It is also necessary here to clarify what is meant by security, scalability, service-oriented architecture of OPC UA communication protocol. In the context of service-oriented architecture,

3.4.1 Object Linking and Embedding for Process Control (OPC)

Object Linking & Embedding is a collection of operating system services that allow you to include component objects in an application or allow you to package component objects for use in other applications [44]. It is a communication standard based on Microsoft's OLE technology that fosters great support of interoperability in the industrial process control [45]. The principle of OLE is to develop modular applications that refer to a groundbreaking step for loose coupling in object-oriented programming. The use of object-oriented techniques encourages to provide applications which are reusable and

easy to maintain [44]. OLE uses component objects, which can be referenced with another object. Furthermore, the objects that belong to the OLE Model can support interfaces to deploy an abstraction layer for other objects. Each interface is a collection of related functions that can be invoked when a user need to interact with a component [44]. Main drawbacks of OLE components does not support inheritance to increase the integrity of the run-time development[44]. This issue led to another issue such as multiple vendors can create multiple types of objects and releases, however, OPC-UA development demanded to solve multiple releases issue by means of data integrity in the protocol layer.

OPC defines standard objects, methods, and properties built on OLE technologies that support data exchange in real-time information such as PLC and client-server software applications [45]. If any OPC based system wants to transmit data to another, OLE technology should be activated in both systems. The absence of interoperability requirements of real-time process automation applications prompted creating a solution for addressing information model by means of OPC UA.

3.4.2 OPC Classic

The OPC Foundation has defined a number of software interfaces to standardize the information flow from the process stages to the management stages [46]. Chiefly, OPC Foundation has created this standardization to implement Industrial automation application like HMIs and SCADA systems to consume current data from devices [46]. OPC Classic constructed in an architecture using the client-server model for information exchange. An OPC Client can connect to an OPC Server to consume data that resides in the server. The advantage of the approach used in OPC Classic works to the definition of different APIs for different specialized needs without a network protocol definition for inter-process communication [46]. OPC UA Classic has the biggest issue about how to enhance security in industrial networks when OPC UA Classic used. On the one hand, OPC Classic is based on Microsoft's Distributed Component Object Model (DCOM) technology, on the other hand, it is purely based on OPC-UA Standards [47]. DCOM uses Remote Procedure Call (RPC) to generate packets that can be shared across the networks. RPC was designed for the distributed networks because RPC can execute a routine in a remote address space. It is a type of inter-process communication, which is a principle term for communication systems. Despite the disadvantages of DCOM such hard to reconfigure, long timeouts and unpractical usage with Internet Technologies

[46]. With OPC Classic, the first data access modeling has been developed for reading, writing and monitoring the value changes. When an expert requires inserting a simulative model with metadata modeling, OPC Classic support this feature in a limited way. Taking into consideration that modeling limitation of OPC Classic, OPC UA was a result to solve the issue.

//Distributed COM

//COM Server --- COM Client

3.4.3 Migration to OPC UA and Service Oriented Architecture (OPC UA – SOA)

"Service-Oriented Architecture (SOA) is an architectural style for building systems based on interactions of loosely coupled, coarse-grained, and autonomous components called services." [48]. Each service discoverable addressed called endpoints and transmit composed messages to each other [48]. Services separate business function to provide a coarse-grained task implementation. Service performs as a coarse-grained task handler that provides breaking large tasks, low communication and synchronization, and assignment tasks into a large processor core. However, service-oriented architectures suffer from load misbalancing, which loads of data is not assigned equally on business logic. A Service Oriented Architecture should communicate business functions in the manner of having an asynchronous request/response. An endpoint is a universal resource identifier where the service can be found [48]. The primary motivation for migrating from OPC Classic is that its message protocol based on Microsoft's COM/DCOM (Mention about DCOM before), OPC UA supports multiple communication protocols and operating systems [49]. OPC UA Security is more enhanced and easier to configure. Updated security protocols and hash methods empower the data transmission between OPC UA Client and Servers. OPC UA application authenticity is provided by X509 certificates that assign to each application uniquely [49].

//Before OPC UA, every OPC Classic has its own address space and service definition. Along with OPC UA, the generic design of address space and service definition came out.

3.5 Services of OPC Unified Architecture

3.5.1 Overview

OPC Unified Architecture is breaking into several services that should be investigated. Our scope is to study services that would be an important step for the thesis. For instance, this study does not examine the Historical Data Access Service and Alarm Service because the web-based software only utilizes current values by means of OPC UA Data Access Interface. Our goal is to evaluate Data Access Interface and implement with an OPC UA SDK. In the following pages, this thesis will present the most important services used by the web service.

3.5.2 Address Space Model

The primary objective of the address space in OPC UA provides a standard way to the clients in terms of elements of OPC UA. More specifically, the Address Space provides a space of objects that can realize to exchange information. To exchange information, the address space act as permanent storage transforming from binary data to high-level objects. In the very beginning, OPC UA has specified as an object-oriented model and every element of OPC UA need to correspond for objects. OPC UA has to comply with this standard. Clients can browse, read and write by means of denoting the address space.

The smallest item in the address space of OPC UA is called Nodes which belongs to Objects [50]. A node comprises Attributes and References which can be reached by Node Class Browse Name [50]. Attributes define Nodes and a node can connect to other nodes with the interconnected information of References. OPC UA Nodes have several classes such as Object, Variable, Method, View, Object Type, Variable Type, Reference Type and Data Type [51]. When a user is endeavored to obtain values of the node, the address of the node in Address Space of OPC UA should be activated. Mainly, a browse name and node-id show to clients in the address space. In order to access attributes or other elements, clients must know the name of browsing and a related node ID. Due to a real-time data processor, the address space has a breakthrough feature where process data saved previously. This thesis is a review of a preliminary attempt to explain items of address space which are [52]:

View: All Nodes lives in a View when browsing in Address Space.

Object: It represents real-world objects and software components and it may use additionally References to define relationships of Nodes.

Variable: The purpose of a Variable is to provide a real-time value when a client is browsing in it.

Method: Method item correspond to callable events by returning a state

The above-mentioned items are defined as the general aim of OPC UA Address Space. There must be data containment when we use these items. Mandatory and Optional selection are contained in type definition so that one can decide how to apply a type.

Object Type: It consists of a definition for Objects

Reference Type: This type used for meta-modeling providing an inheritance of objects and defines meanings of a relationship among nodes.

Variable Type: It defines some types such as Historicization of Variables, Minimum Sampling Interval, Access Level, User Access Level, Array Dimensions, Value, and Object Type. Variable Type has a vital role in practical implementation because the definitions of Variable Type enables browsing, reading, writing, and subscription how to make them possible.

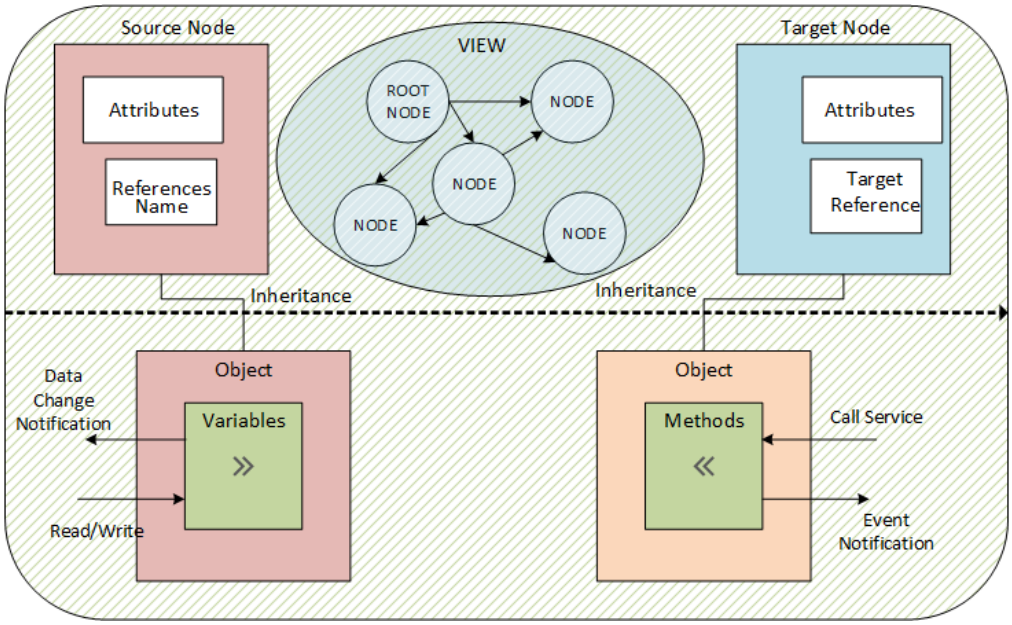

Figure 3-1: OPC UA Address Space [50] [53]

As shown above, an address space consists of nodes. Each node has fundamental attributes and references. Object classes have variables and methods to embody object-oriented architecture. Nodes are connected to each other in an address space through a service called View. Upper space that has separated by a dashed arrow shows an address space and below one demonstrates an information model that inherits from and links into address space. The View Service in OPC UA helps to navigate hierarchical references to search for information about nodes, attributes, and objects of nodes.

3.5.3 Information Model

The primary objective of the Information Model is to represent the structure of the objects that have relationships with Variables, Methods, and Events and provides a set of predefined types and rules which can be expandable [52]. Beyond this concept, a semantic modeling tool provides a two-way standardized communication version of an Address Space. Strictly speaking, it is a way of object-oriented representation of servers that can be reached by clients. The main difference between Address Space and Information Model is being suitable with meta-modeling languages such as UML and SysML. Information Model has a higher abstraction layer to simulate the Object-Oriented Paradigm of OPC UA Protocol.

As indicated previously, OPC UA is a protocol based on Service Oriented Architecture so that every object can communicate related service to exchange corresponding data. Object Types defines types of object dependent on the object and these types can be customized with multiple definitions. Variables are the main components of objects and it represents data values in the objects. Variable Type and regarding Data Type define a structure of variable. The main challenge of any OPC UA Software shows all data types that relate to Scalar (Fundamental Data Type), or Application-Defined Data Type. Our method is a clear improvement when a user requires reaching a scalar typed or application-defined typed objects.

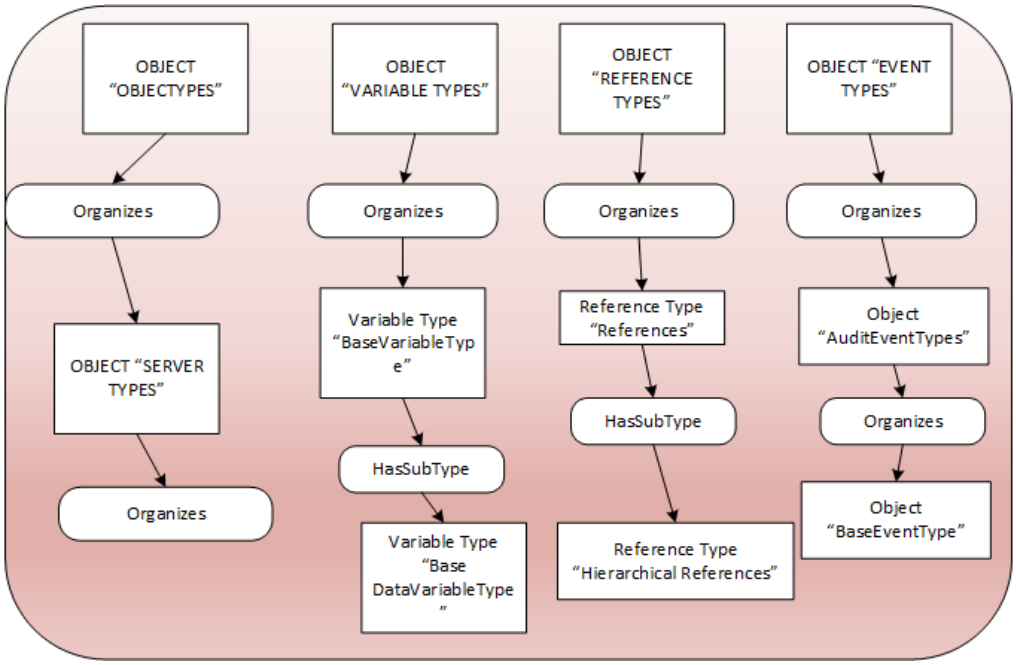

Figure 3-2: OPC UA Information Model [54]

Information Model defines objects, variables, events, and references relationship between them or inner structures. As illustrated in Figure 3-2, a reference organizes the relationship between object, more specifically Node Classes. Node classes are a subset of the abstraction method among nodes in the address space. Due to space limitation, Figure 3-2 indicates limited elements of the Information Model. For instance, reference has not only HasSubType, but also comprises of HasTypeDefinition, Organizes, HasProperty, and HasComponent. HasSubType defines subtypes and supertypes of references. While subtypes are defined explicitly, supertypes are identifying through HasSubType implicitly. For instance, the BaseDataType has multiple references as HasSubType and with a BrowseName and NodeClass, it is defined explicitly as primitive, structured or XML elements. It is not a mandatory type of definition in references; nevertheless, it is a mandatory schema structure build up a hierarchy in OPC UA. HasTypeDefinition is a definitive term for the type definition of an Object. Every object has a relationship with other objects so that HasTypeDefinition should occur more or less the number of relationships that an information model has. Organizes determine types of folders and their internal structures so as to group a set of objects. Nevertheless, Organizes reference may be used for Objects of the FolderType, which has a usage restriction [50]. OPC UA Servers have beneficial items called Folders that serve as

separator objects that have similar type definitions. HasProperty used to describe Properties, which properties have relationships with other properties of a Referent Type. HasComponent identifies the data variables, the Methods, and Objects contained in the Object [50].

Consequently, every element of the Information Model defines and has a relationship in a hierarchical way or a non-hierarchical way. In a general manner, the information model defines types and references, which are the essential component of abstraction. As a result, the web-based application depends on the information model to browse between nodes with their reference and to identify the types of the nodes, more specific objects, by using this service.

3.5.4 Publish/Subscribe Service

This service defines the way of communication by using message-oriented architecture or standard transmission protocol of Open Systems Interconnection. By using a middleware, publisher and subscriber can be de-coupled. Subscriber or Publisher may be an OPC UA Server to transmit information to clients; however, a serious weakness with this method is a necessity to write values temporarily to the address space. Due to the installation of broker infrastructure, message-oriented service brings more cost to any architecture. In the literature, middleware of Publish/Subscribe Service breaks up two various titles, which is Broker-Based Middleware and Broker-less Middleware [55]. It is generally accepted that the broker-based middleware has a common use in industrial internet of things. The fundamental characteristics of the broker-based are detaching different protocols from each other through a broker, confidentiality between publisher and subscriber, and integrity can be ensured among publisher-subscriber pairs.

Dynamic Server uses message-oriented architecture and it behaves as a publisher. Hence, it designed differently from other OPC UA Server to takes messages to be sent to particular receivers. Although the fundamental data collector is eniLINK with sensors, Dynamic Server send list of values with timestamps and topics to subscribers conveniently. This communication occurs asynchronously because synchronous communication is not suitable for broker unless it has a non-blocking input-output queue. Through the message broker architecture as depicted in Figure 3-3, each flow of OPC UA can transmit via a non-blocking queue, which is the fundamental step for the asynchronous communication. As shown in Table 3-1, pros and cons have been listed so that broker-based architecture reduces latency of streamed data to store data into any source such

as cloud service or real-time database without loss. The generated data set from Dynamic Server defines a structure through its Information Model and this model contains data from sensors and actuators that connect to eniLINK. Message Broker Service collects data from production and manufacturing system in the smart factory of Fraunhofer in helping creation of linked data.

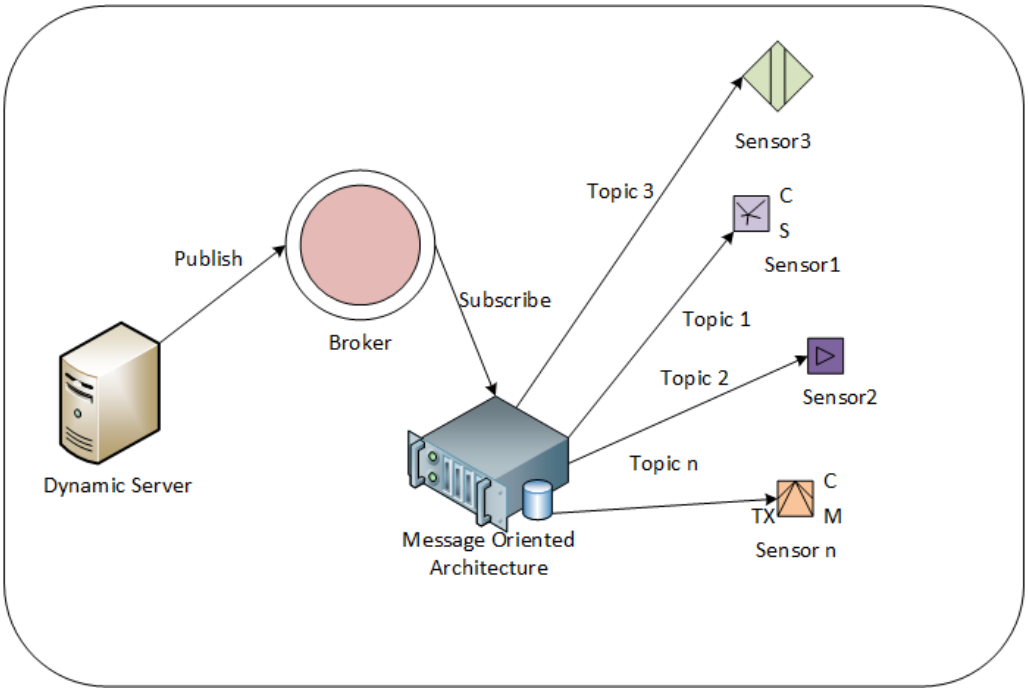

Figure 3-3: The Dynamic Server Publish/Subscribe Model

Types of Middleware	Advantages	Disadvantages
Broker-less Middleware	• No Central Point of Failure, • Legacy Devices Support • No additional software components like Broker	• Protocol Dependency because of non-existence a Broker

Broker-based Middleware	• Broker reduces latency and overhead generally	• A network bottleneck could be disastrous

Table 3-1: Types of Middleware Publish/Subscribe

3.5.5 Discovery and Aggregation Service

The principle of SOA states a service-oriented architecture should has a service consumer, a service provider and a discovery service. The discovery service is important to construct a micro service structure instead of statically typed endpoints. The practical implementation is able to use discovery service and then clear-cut key points described in service discovery of OPC UA. To overview better the discovery process, these scenarios should be examined as follows. A client and a server can be in a same host or in a same network. Moreover, a client can connect different servers which are in a different network.

In the discovery process of any network, a discovery service allows locating items of a network by a specified device of a network. For instance, a client can find a server via a proxy server without knowing any details except the address of a proxy server. OPC UA Discovery Services work with the same principle by using endpoints to establish communication between OPC UA Clients and Servers. Discovery Service at OPC UA Standard can be divided into two main topics in terms of application domain where application lodged in. These are "LocalDiscoveryServer" (LDS) maintains discovery requests for all applications if clients and servers are on the same domains and "Global Discovery Server" (GDS) preserves discovery information for all applications if clients and servers are on the remote domains [53]. GDS can be full-fledged OPC UA Server and organize other discovery services in a central manner. Conversely, LDS can only behave as a service or serve other LDS supporting multicast networking. In this work, a local discovery server is examined in terms of benefits and drawbacks on the existing architecture. A client that wants to connect to a real server through a discovery server should use a set of service sets which are "Register Server", "Find Servers", "Get Endpoints" and "Find Servers On Network" [53]. When a client requires establishing a connection, a session is not supposed to be created. To achieve this feature, every server has a Discovery Endpoint to connect clients without creating a Session [53]. However, this could be a security vulnerability because a client and a server do not share certificate among them and lack of a session creates an unsecured connection.

A Discovery Server has two types of endpoints, which are discovery endpoint and registration endpoint. While a discovery endpoint provides a connection to clients, registration endpoint awaits a result from discovery endpoint whether it has a connection with the client or not. After a client obtains a "GetEndPoints" service set from a discovery process, it can open a secure channel by providing a certificate, hashed authentication or anonymous way to perform opening a communication channel. Accordingly, the between finding an endpoint and sending the endpoint request has not authentication schema. Hence a discovery service implementation could cause a security vulnerability inter smart factories.

Types of Middleware	Advantages	Disadvantages
OPC UA C++ Unified Architecture	Very fast while registering server and find endpoints, Legacy Devices Support, Automatically initialized by Windows Service Manager	Only windows based service setup,
Unified Automation Discovery Service	Strong support for a variety of languages	Central Point of Failure, Bottleneck if there is not enough maintenance
Node OPC UA Discovery Service	Easy to use with web platform technologies	Dependency on Node Virtual Machine and npm package manager

Table 3-2: OPC UA Discovery Services

The other way of bringing together OPC UA Clients and Servers is to use an aggregation server. "The aggregating server establishes a separate session to its underlying servers for each of its servers for each of its users." [53]. Aggregation Server solves complexity problem of a mesh topology by diminishing complexity of the design of the system. An

aggregated server suffers from a single point of failure. The main drawback is that aggregation may change code size and complexity of OPC UA Server. The aggregation server can be solved in two different ways. First method is detaching the requests from the client to the server. Unlike the first method, the latter method imports relevant part of address space regarding OPC UA Server and traverse in a recursive way. However, "Dynamic Server" is our main semantic data source and quasi OPC UA server, which is not implementing any aggregation concept. This research gives information with advantages and drawbacks rather than implementing an aggregation server.

Aggregated OPC Server can be used as a client to fetch data from the OPC UA Server that resides in an external domain. Aggregated Server can behave as a proxy server to connect other servers and fetch data from an external domain within a blob of address space. This kind of server may utilize different Sessions as a proxy server against monitorable nodes that works under different OPC UA servers.

3.5.6 Subscription Service

When a stateless architecture such as RESTful API implemented onto a stateful architecture such as session-based protocol, there would be an issue for identifying alteration of data. As streamed data incoming from servers, the stateless architecture such as RESTful API has deficit to refresh data instantly. A simulated data that is continuously changed has a great overhead when sending a read request over again. Moreover, data fluctuation has an important role to analyse data by experts. Hence, instead of sending a request, a subscription might have sent to identify variable, attribute or object changes with a set of features. In order to remedy this repercussion, a subscription is sent with particular monitored items into a session and monitored items serve as a polling mechanism. As illustrated in Figure 3-4, a single monitored item and multiple monitored items attach to a subscription. This service reduces time and space complexity of reading request by showing all changes in a single subscription. Three types of changes can be observed in OPC UA Protocol to simulate data, which are data changes of Variables, Objects of Events or Attributes. The sampling interval is a key component of monitorable nodes to detect changes in a particular polling time. After assigned a sampling interval, OPC UA Server can notify OPC UA Client when an Attribute, an Object or a Variable has changed. The implementation of web-based application dispatch a binary indicator belongs to monitorable nodes, thus the web service sends a general subscription request without monitoring nodes' topics and ID. A filter decides whether the next notification

of a subscription should send or not. Filter can eliminate different type of item to be monitored in order that unnecessary notification cannot overflow in the system. A subscription service put all notifications into a queue that can transfer without blocking respective notification.

If a new notification has been entered to the queue, a former notification should be deleted to free the queue size. Monitored items should comply with the minimum sampling interval. As a result, minimum sampling interval defines the degree of sampling interval and this minimum value of the sampling differ from a node to another node. However, the underlying structure of update cycle is not synchronized, so the system should explicitly synchronize all sampling values in order to fetch correct notifications with the decent value. Accordingly, the amount of smallest minimum sampling interval can create a maximum load of traffics for OPC UA Server and lead to produce buffer overflow, which is a general malicious attack that used by harmful minds.

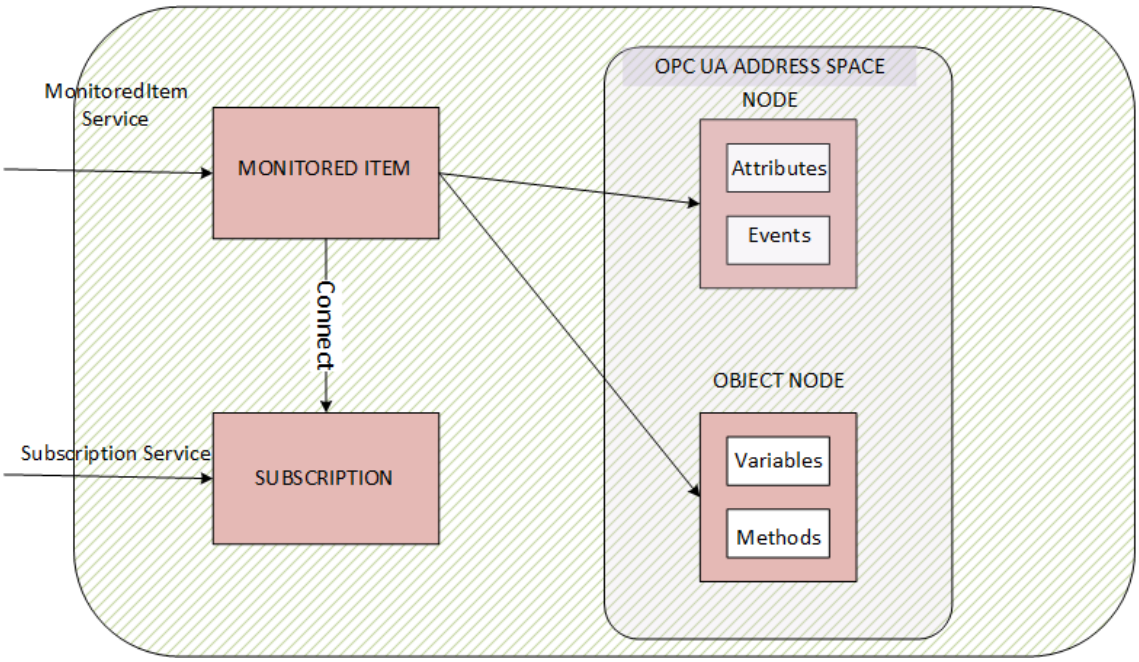

Figure 3-4: Subscription and Monitoring Item Services [53]

3.6 Chapter Discussion

Services of that design the concept of SOA architecture can alleviate design issues in a capable manner. However, the main issue is that a stateless web-based application enables communicating stateful OPC UA Servers. Discriminating the services of OPC UA may make the development phase easier for REST request. Session timeout could be a problem while reading or writing dataset to an OPC UA Server. If the operation timeout of a single request is above than a session timeout, the number of failing requests are tend to increase, which is an undesirable result for the web-based application. Address Space Model and Information Model are essential services of OPC UA so that one can create a semantic model from an internal model of OPC UA Server. Publish/Subscribe and Subscription Service process the streamed data that changes quickly within a minor period in the production system. If a server closes a session silently, the OPC UA server does not delete regarding subscription, however, the server cannot observe data changes anymore. After setup a reconnection, a subscription may continue where the session leave off. As far as subscription feature concerned, a stateful architecture offers more advantage than the stateless one. Service orientation enables integrating RESTful architecture so that the web-based architecture can enforce a microservice architecture instead of monolithic architecture.

Smart Factories enables more analysis of produced data, integrable communication protocol, and advanced controllable manufacturing devices. Our proposal of the web-based software can speed up the controlling of multiple stages in a manufacturing process with OPC UA Services. OPC UA Protocol provides a clear abstraction and service-orientiation to map a HTTP request onto the services.

//Discovery and Aggregation Service. Upon sessions it can be discussed what are those advantages.

//Subscription Service should be synchronized

//Session timeout and id problem of OPC UA

//Session and Monitorable Node Concept

//Importance of services

//Use your own words

//Significantly condense the original text

//Provide accurate representations of the main points of the text they summarize

//Avoid personal opinions

4 Serialization from Non-linked Data into Linked Data in Industry 4.0

4.1 Overview

When this research has been conducted, one of the research problems was serialization and representation of non-linked data in a linked way. OPC UA Protocol allows presenting the data from various devices and human operators can interpret the data in a specific domain. Essentially, the meaning of the data cannot interpreted by machines. This chapter summarize a theoretical background how a machine infer from structured and unstructured data. Hence, this research comprises time-series and static data values of eniLINK and evaluation of the linked data sources to conceptualize a particular problem in the sense of machines. Time series or real-time values from sensors, software logs or business packages are supposed to be converted into a resource description framework in order to use their linkages. Time-Series Data Source from devices of the Internet of Thing makes a disadvantage situation as for real-time environment. The major drawback is to extract meaning from a network of deployed sensors. Because raw sensor data is useless unless properly annotated. While transforming raw sensor data, another drawback comes up limited resources in terms of processing, storage capabilities and bandwidth of a network. Chapter 2.2.3 remarks establishing ways to automatically process of raw sensor data has been studied by previous researches. Limitation of storage can be alleviated with public or private clouds.

4.2 Resource Description Framework

RDF stands for Resource Description Framework that is a data model for interchanging web-based information. All of the members in RDF represents as triples and each triple might have connected to other triples. As understood from its name, it is a framework for supporting resource description and metadata in the Web. First RDF version provides a set of features that can be used interoperably with the extensible markup language (XML). RDF specifications are controlled by the authority of W3C in terms of update and maintenance of new requirements [3]. RDF consists of several types of models

[3] https://www.w3.org/standards/techs/rdf#w3c_all

that currently used in industry. The main part of an RDF data is a prefix so-called International Resource Identifier (IRI)[4]. Resource Identifier is rarely not feasible to every generated document from an extensible markup language. The main purpose of the Resource Description Framework is to integrate data in the web. Algorithmic representation of RDF is a graph data structure which has a set of vertices and edges. Navigational movement of RDF is allowed by graph traversal algorithms such as Breadth First Search or Depth First Search[5]. A fundamental property of RDF data that navigates internal structure with IRIs

Serialization stands for converting an RDF Format to another to use a variety of syntax notations, so the particular encoding can produce a variety of triples. After serialized an RDF resource, one can obtain the following formats. The consortium named World Wide Web (W3C) inspect RDF Serialization format observing the following goals [6]:

- All rules should be integrated smoothly to RDF
- URI Abbreviation should comply with namespace rules
- Repetition of another object for the same subject should be divided with special Unicode characters
- All languages need to be readable, natural and extendible scope of languages

//Put an image and explain more over the RDF structure

Turtle: Besides being a strong alternative for RDF/XML, the syntax of Turtle Semantic is similar to SPARQL queries. Turtle Notation is a compact and clear structure. Predicates and subjects can be marked as block

Notation 3: N3 triples are similar to Turtle RDF unlike it is supporting underscored namespaces. N3 triples syntactically is a subset of Turtle RDF because it was designed to be a simple format than Turtle RDF. As much as there are similar syntactic definitions, a variety of differences unlike Turtle RDF has been observed. Triples follow the pattern "subject-predicate-object" and terminal notation. Notation 3 has enlarged grammar structure with extra features more than Turtle RDF and NTriples.

[4] https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#introduction-1
[5] https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
[6] https://www.w3.org/TeamSubmission/n3/

N-Triples: Parsers and serializers can parse this format in an easy way because of simplicity. There is no complicated grammar rules with N-Triples but it is not a good format as human-readable. N-Triples has a trade-off to increase machine readability over human-readability. The simplest triple statement is a sequence of subject-predicate-object containing white spaces and dot-separated values. It has a tedious format has not abbreviation feature that makes hard to read by humans.

JSON-LD: To provide a lightweight linked data format, objects should be converted as a human-readable format. This format is a compact format that has compliance with JSON data. JSON-LD format has a compact dependency on JSON format and it can be used without prior information about RDF. Typically, JSON-LD contains the same structure as compared to RDF like primitive types for nodes and IRIs definition for edges. Standard parsing methods for JSON can be used for JSON-LD interchangeably.

//Explain types, aliasing, nesting, language

Web Ontology Language (OWL): OWL stands for Web Ontology Language represents a clear and compact way among relationships of data. Prefixes with IRI is one of the fundamental structure of OWL linked data. OWL can define a class to provide abstraction within the same linked data document. OWL definition is a standard with a Prefix IRI such as "http://www.w3.org/2002/07/owl#" and it is a mandatory field to define if an OWL source used in a linked data. This ontology language leverages RDF Schema to identify complex knowledge requires complex properties [56].

//What is the benefits of OWL – Why did they invent it?

4.3 SPARQL Query Language

In the previous section, RDF was explained and detailed in order to analyze SPARQL. RDF has a collection of graphs and these graphs are directed and labeled. As a result, triples of graphs can be obtained with a query language from databases or files. The SPARQL Query Language is a declarative query language for performing data manipulation from RDF datasets[7]. The structure of SPARQL resembles Structured Query Language (SQL) very much but the SPARQL was designed using for semantically structured

[7] https://medium.com/virtuoso-blog/what-is-a-sparql-endpoint-and-why-is-it-important-b3c9e6a20a8b

triples, not for relational datasets. Additionally, SPARQL is a definition of a protocol working with HTTP Request by defining “User-Agent”, “Content-Type” and “Schema”.

PREFIX, SELECT and WHERE are three basic operators of SPARQL Protocol. PREFIX makes the serialization steps easier referencing IRIs. Prefixes are used for abbreviating of IRIs in a query. “SELECT” and “WHERE” statements used to find location of objects. IRIs has a wider range of characters to be used in order to accommodate a wider range of languages than URIs [57].

Mainly, SPARQL Requests are characterized with Remote Queries or Native Queries. Remote Queries define as sending a query against a remote SPARQL endpoint. Remote Queries needs an endpoint definition provided by Linked Data Source. As for Native Queries, they work mostly in a local database such as graph databases or files and need a query processor to carry on a query against local sources.

The SERVICE keyword reduces the complexity of queries and hands the complex query duty over the SPARQL Service. Real Time Data Annotation Service “KVIN” uses this keyword to prevent making a complex annotation by users.

Sometimes one needs to fetch multiple values in one single query with integrity known as Federated Query. UNION statement can help at this situation to provide federate property into queries. What can be clearly seen is the working principle of UNION is similar to Outer Join in SQL. It takes all Cartesian Product Multiplication, so one can state that the result of an answer impact issue of redundancy. To reduce redundancy, the UNION can be used with an OPTIONAL statement or query can be optimized with only an OPTIONAL statement. OPTIONAL used for allocating a particular portion of SPARQL into results of triples. OPTIONAL reduces redundancy of data and gives every match in any triples. It is a common usage OPTIONAL query with FILTER that allows measuring up a couple of criteria.

One of the biggest problems is searching with blank nodes among triples without clear IRIs information. A generated Turtle RDF can define blank nodes that have no clear identification while using with a SPARQL query. To solve this problem, a RDF file can be pre-processed assigning with a traversed property. A traversed property is a linkage between two properties to connect triples each other.

4.4 Serialization of the Address Space Model into Linked Data

Most of RDF Sources in the web has typed with RDF/XML. RDF/XML was a step from XML language to RDF and it has not even namespaces that are vital roles of semantic data. XML Schemas are other problems because of complex encoding and enlargement. In XML or XML-based documents such as RDF/XML, all items have strong hierarchies; hence, this leads to heavy parsing overload. On the contrary, RDF has collections of relations to traverse inside of documents or through documents in an efficient way. When creating an XML from OPC UA Server, the first task should be converting of namespaces. With browse name property of namespace, all initial base of nodes is inserting in an XML document. The second task is to browse among nodes with references and node ID. Display Name and Description will be inserted under OPC UA Objects or Variables in the XML Schema. References are converting as a sub-element of UA. Values of Variables are used for an object in semantic data such as Turtle RDF. While sending a SPARQL Query, a query attempt to obtain are using values in order to give a result to the Semantic Question Answering. XSL Transformation Language can be used for serialization from XML to RDF.

Algorithm 1 Node Extraction

```
1: function MAINFUNCTION()                                   ▷ Starting point
2:     export = ServerExport(serverurl, filename)
3:     export.IMPORT NODES(serverurl)
4:     export.EXPORT FILE(outputFile, namespaces)
5: function BUILD NODE TREE(nodes)                           ▷ Node Formatting
6:     client ← GETENDPOINT()
7:     client ← CLIENT(serverurl)
8:     nodecumulated ←None
9:     nodeID ←0
10:    for node < nodes do
11:        nodecumulated = node.nodeid.Namespaceindex
12:        for ref < node.getreferences() do
13:            nodecumulated.extend( ref.nodeid.Namespaceindex)
14:        nodecumulated = list(set(nodecumulated)    ▷ Clear duplicates
15:    return nodeID                                  ▷ Return node id list
16: function IMPORT NODES(serverurl)                         ▷ Traverse Node
17:    client = Client(serverurl)
18:    client.connect()
19:    for ns < client.getNamespaces() do
20:        namespaces[client.getNamespaceIndex(ns)] = ns
21:    root = client.getRootNode()
22:    child = client.iterateChildNodes()
23: function EXPORT FILE(outputFile, namespaces = None)   ▷ Export into
    XML
24:    if namespaces != None then
25:        for node != nodes do
26:            if node.nodeid.namespaceindex is namespaces
27:                nodes = [node]
28:            else
29:                nodes = list(nodes)
30:
31:        export = XmlExport(client) then
32:        export.BUILD NODE(nodes)
33:        export.appendXML(outputFile)
34:
```

Figure 4-1: Extraction Algorithm of OPC UA Address Space [16] [58]

As shown in Figure 1.1, the algorithm of extraction from OPC UA Address Space has been defined as a flowchart. Once a node list defined by OPC UA Address Space, all node-ids and namespaces might be saved into the list.

//Explain algorithm briefly

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://opcfoundation.org/UA/2011/03/UANodeSet.xsd#> .

<unknown:namespace> :UANodeSet <unknown:namespace#UANodeSet> .

<unknown:namespace#UANodeSet> :NamespaceUris
<unknown:namespace#UANodeSet/NamespaceUris> .

<unknown:namespace#UANodeSet/NamespaceUris> :Uri
<unknown:namespace#UANodeSet/NamespaceUris/Uri> .

<unknown:namespace#UANodeSet/NamespaceUris/Uri> rdf:value
"http://opcfoundation.org/iwu/DynamicServer" .

<unknown:namespace#UANodeSet/NamespaceUris> rdf:_1
<unknown:namespace#UANodeSet/NamespaceUris/Uri> ;
    :Uri <unknown:namespace#UANodeSet/NamespaceUris/Uri_2> .

<unknown:namespace#UANodeSet/NamespaceUris/Uri_2> rdf:value
"http://opcfoundation.org/UA/Diagnostics" .

<unknown:namespace#UANodeSet/NamespaceUris> rdf:_2
<unknown:namespace#UANodeSet/NamespaceUris/Uri_2> .
```

Listing 4-1: Preview of Generated Semantic Data from an OPC UA Server

As shown above, meaningful predicate names are crucial steps to employ with SPARQL queries. Converting from natural language question into triples, verbs often are mapping into predicates. Predicates are also edged labels that connect two nodes in the graph data structure. A missing predicate of a node stands for a blank node. A blank node is one of the evaluation methods while creating semantic data. Although a blank node is not a single evaluation method theoretically, nevertheless it is an essential measurement to evaluate for applicability of question answering with semantic data. Unknown namespace

In this study, SPARQL queries used with Turtle Data Source. The following SPARQL query has been used to fetch triples from generated data.

```
""" SELECT DISTINCT ?property
                       WHERE {
                         ?s ?property ?o .
                         OPTIONAL { ?s ?p rdfs:label. }
                       }
                    """
```

Listing 4-2: Sample SPARQL against a generated local source

```
(rdflib.term.Literal(u'linkedfactory.iwu.fraunhofer.de/linkedfactor
y/demofactory/machine1/sensor5'),)
(rdflib.term.Literal(u'AnonymousIdentityToken'),)
(rdflib.term.Literal(u'When the action triggering the event
occurred.'),)
(rdflib.term.Literal(u'linkedfactory.iwu.fraunhofer.de/linkedfactor
y/IWU/Rollex/PowerMeter'),)
(rdflib.term.Literal(u'Reports diagnostics about the server.'),)
(rdflib.term.Literal(u'ns=1;s=root_Demo_Scalar_SByte'),)
(rdflib.term.Literal(u'A numeric identifier for an object.'),)
(rdflib.term.Literal(u'i=2403'),)
(rdflib.term.Literal(u'Pure Python Client'),)
(rdflib.term.Literal(u'ns=2;i=1075791275'),)
(rdflib.term.Literal(u'i=11891'),)
(rdflib.term.Literal(u'i=3181'),)
(rdflib.term.Literal(u'i=290'),)
(rdflib.term.Literal(u'i=3094'),)
(rdflib.term.Literal(u'ns=1;s=root_linkedfactory.iwu.fraunhofer.de_
linkedfactory_demofactory_machine2_sensor7_value'),)
(rdflib.term.Literal(u'The type for non-looping hierarchical
references that are used to define sub types.'),)
(rdflib.term.Literal(u'i=11737'),)
(rdflib.term.Literal(u'An object that represents a file that can be
accessed via the server.'),)
(rdflib.term.Literal(u'i=298'),)
```

Listing 4-3: An answer from generated OPC UA Semantic Data

4.5 Serialization of Streamed Data into Linked Data

Continuous sensor data should be converted instantaneously into semantic data to be utilized with a SPARQL endpoint. Chapter 2.2.3 examines a couple of studies on how to accomplish with dynamic data. Mainly, one of the ways is extracting all RDF continuously and store into a non-relational database such as MongoDB[8] or NoSQL[9]

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix : <http://example.org/data/values.csv#>.

<http://linkedfactory.iwu.fraunhofer.de/linkedfactory/values.csv#row=1>
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#time> "2018-09-
28T06:49:16.9230000+00:00"^^xsd:dateTime;
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#value>
8.142857142857142.
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory/values.csv#row=10>
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#time> "2018-09-
28T06:49:43.9260000+00:00"^^xsd:dateTime;
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#value>
8.166666666666666.
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory/values.csv#row=100>
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#time> "2018-09-
28T06:54:13.9650000+00:00"^^xsd:dateTime;
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#value>
8.166666666666666.
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory/values.csv#row=1000
>
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#time> "2018-09-
28T07:39:14.3010000+00:00"^^xsd:dateTime;
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory#value>
4.166666666666667.
<http://linkedfactory.iwu.fraunhofer.de/linkedfactory/values.csv#row=101>
```

Listing 4-4: Generated RDF from Real-Time Data Source

[8] https://docs.mongodb.com/
[9] http://nosql-database.org/

As shown above Figure 4-5, a generated file was obtained from eniLINK without making extra IRI processing. Lack of clearly defined IRI complied with eniLINK, it is partly useful to send a SPARQL query with Semantic Question Answering.

```
@prefix : <enilink:model:users#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

Listing 4-5: Enilink Sample Prefixes

As demonstrated in Figure 1.1, it is possible to customize according to necessities of at the top line.

4.6 KVIN Continuous SPARQL Service

This thesis utilizes an API that has implemented as a service known as KVIN to send a SPARQL request into a specified endpoint. This service is an internal development that is based on a combination of a triple store (RDF4J) and a key-value storage library named LevelDB compatible with time-series data. Continuous SPARQL Service was examined with Chapter 2.2.3 which analyzed reviewing past literature with architectural differences in the market. KVIN is a continuous data stream service that holds limited annotated linked data so that we could use streaming for the purpose of fast prototyping. Annotated sensor data used for the semantic question answering system. Due to data scarcity, our functions of question answering is limited, however, the platform proves any kind of system utilizes natural query to get semantic data. Instead of using a continuous SPARQL language, KVIN is mapping semantic data with properties internal structure. A namespace is added in KVIN Service for the sake of clearness to convert easily SPARQL triples. To send a SPARQL query, a system requires an endpoint, e.g. "localhost:10080/sparql". A SPARQL endpoint process a request on HTTP protocol that is wrapping up SPARQL protocol that verifies the structure of query as syntactical correctness. Syntactical correctness has provided by a SPARQL validator so that a SPARQL endpoint should have a validator. In the practical work, the architecture of Semantic Question Answering uses SPARQL Endpoint with validator but the query is not sending with the address of the endpoint. Instead of direct-endpoint setup, KVIN tool uses a

testing framework "Selenium" with Python language to get triples. A sample SPARQL as shown below:

Relationships with other components of KVIN can be observed as below in Figure 1.1 "General KVIN Service". KVIN Architecture maps the continuous values onto graphs in linked data. Then, Key-value graph databases to connect the nodes each other through properties. Unlike relational and hierarchical databases, there is no primary key or parent nodes relationship between objects. Nodes might have properties and relationships to traverse from a start node to the end node. The execution time of a query increases proportionally according to size the path of traversing not all size of a graph in the store. This is one of the biggest advantages of a key-value database over relational or hierarchical databases. KVIN SPARQL Service has a strong relationship between Linkedfactory service which are continuous data provided by devices.

//Talk about KVIN Influx DB

//First – InfluxDB – time series to key-value mapper – LevelDB -- KVIN Service

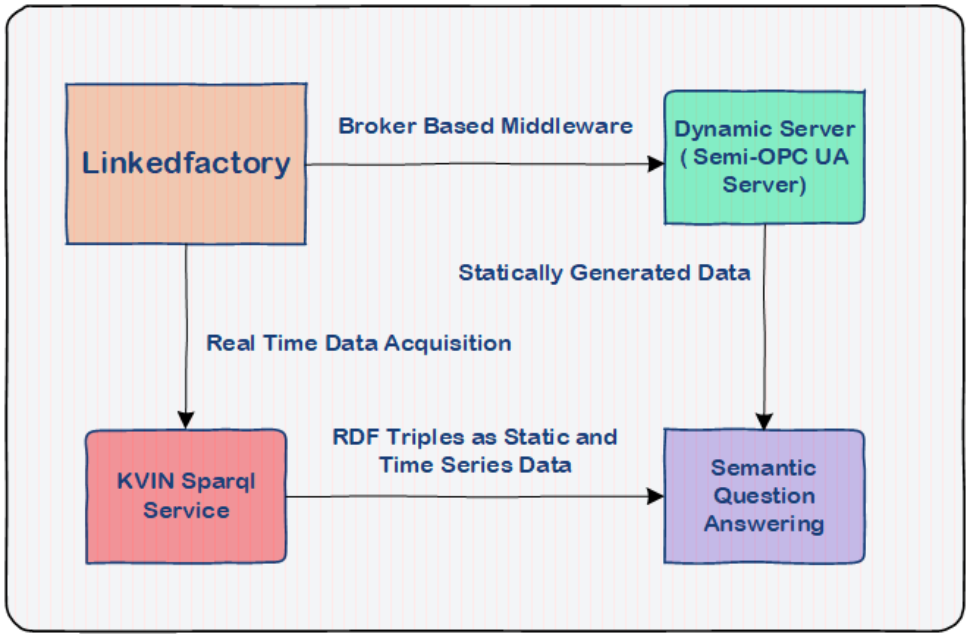

Figure 4-2: General KVIN Service

4.7 Chapter Discussion

Serialization from one to the other always has overhead for designing an architecture. The Resource Description Framework complies with the requirement that streamed data or generated data by an OPC UA Server. Nevertheless, the extensible mark-up language is an intermediate processing between a generated data and the resource description framework. As understood from testing of XSL Transformation, an address space of OPC UA that has complex structure and massive amount of data may not be a convenient way to serialize. Moreover, a serialization step could take time proportionally the size of data, which a server consider that regarding client connection could not respond back within the session timeout of the server. This leads to a serialization failure and the step may close the connection silently. XSL Transformation requires the extensible mark-up language format from an OPC UA Server and the tool starts from scratch when a request sent by a client with regards to serialization between XML and RDF. Luckily, an OPC UA Server can assure the language compliance, but it may not intervene the process of serialization. Hence, an OPC UA Server shall not take into consideration the small changes such as monitorable values, which it could cause a large repercussion in the serialization of hierarchical structure in the server.

Streamed Data Serialization may take benefits from either C-SPARQL, Instant Semantic Source Creation or Key-Value Mapping. KVIN Service follows key-value mapping that has the lowest overhead while performing against SPARQL Endpoint. However, limited description with linked data and extendibility of a substandard of SPARQL might have occurred. C-SPARQL. Key-Value Mapping is a quasi C-SPARQL solution, which the KVIN has a specified window size to collect data within a range of time.

As a result, combining static and streamed data as linked data source is the best way of representation and tackling the problem as a whole. Due to complexity of design, it is not possible to create such application without a large of effort. On the one hand, detaching static and streaming data serialization step would be beneficial if we want to reduce the graph size for traversing. On the other hand, describing a natural query regarding streamed data or static data is a difficult problem to solve so that we should a further step of natural language processing tool that distinguish the query related to which one.

5 Semantic Question Answering System

5.1 Overview of a Question Answering System

The question answering is a combination of natural language understanding and information retrieval theories. On the one side, a question answering performs a task on natural queries to observe syntactically and semantically, on the other side it is an activity to obtain relevant information model that has been searched. Therefore, the major aim of a semantic question answering is to identify an answer from a collection of semantic data such as RDF, RDF/XML, JSON-LD or N3. Question Answering is similar to Information Retrieval and Information Extraction. Since these keywords have perplexed definitions that one should examine with similar and dissimilar points, in order to better understand the question answering process. Information Retrieval is a term used for locating a document that is required by a user, but a user defines a relevant answer after obtaining a document. Information Extraction is a term for extracting a set of information from a user input so as to learn relationships between searched keywords and documents[10]. Broadly speaking, question answering is a balancing process with natural language understanding between natural language understanding and information retrieval. According to the domain type of question answering as shown below:

Open Domain Question Answering: A user can ask any topic that he wants to reach a result from a general domain.

Closed-Domain Question Answering: A user can only ask questions against domain-dependent document-based architectures. For instance, one can ask general questions against a specific text document which have collaborated by online sources.

Restricted-Domain Question Answering: It is more likely one can ask a question against semantic documents in order to obtain results. Main characteristics of restricted-domain are that data sources can be different from closed-domain and open-domain. In this domain, the answer and result sets are circumscribed and complex, so information retrieval capability is strictly relevant to natural language processing capabilities.

[10] https://www.ontotext.com/knowledgehub/fundamentals/information-extraction/

Question Answering System is broken into question types which will be explained within this chapter.

Question types handling is an essential step for any question answering. On the one side, closed-domain and restricted domain question answering systems used for eliminating the unrelated type of questions, on the other side open-domain question answering system takes types of questions to formulate with rule-based architecture. According to types of questions will be shown as below:

Factoid Questions: A factoid question is about providing concise facts. For instance, "What is the population of Berlin?" is a factoid question that should be narrowed down from a general topic into a specific one. Otherwise, an open domain question answering system would be ineffective against a factoid question.

Keyword Questions: Keywords are one of the basic search items which are used in search engines. In the early phase of the Internet, researches have been focused on how to extract documents from keywords. The simplicity of grammatical and semantical structures, a keyword-based search has always been a prominent topic for question answering. This work can use a keyword to extract information from the Turtle RDF data format with a specialized keyword. Verbs or more specifically predicates and objects counterpart nouns in RDF are based on Fraunhofer IWU's data source. So one needs to have limited information about the internal system.

Indicative Questions: An expert can make a sentence through request words in the sentences, e.g. "I would like to know what does linkedfactory contain?".

Reasoning and Notional Questions [11]: A user might be asked a question defining by notional keywords, e.g. "Can you tell me the system health in trouble?" or "Can our system stay alive?". These are special queries that this work did not implement. : "Why" and "How" questions also show reasoning to induce a result from a series of events. The main reason that this research did not implement is that the types of questions require a number of data more than we have.

Indirect Requests: A user can ask a question like "I would like to list all of the members in linkedfactory?" or "Give me the value of sensor1 in machine1". The main feature is listing of the information

[11] https://www.fer.unizg.hr/_download/repository/TAR-09-QA.pdf

Boolean Questions: Answer must be yes or no. We have used this type of question to understand system status. “Is the system health good for sensor1 in machine1?” and “Is the //

//Notes for questions type will be completed

5.2 Natural Language Understanding

The Natural Language Processing has two subsets, which are Natural Language Understanding and Natural Language Generation. Natural Language Generation is a concept to create from a language description to another description that may constitute a set of formal rules, rules of syntax and semantics. For instance, a machine translation system provides a language exchange interface to perform a set of linguistic rules by words and sentences transforming into another language. The scope of this work does not cover Natural Language Generation; consequently, this work examines normative methods of Natural Language Understanding.

Natural Language Understanding has a key role in Human-Machine Interaction Systems. It enables computers to understand a natural query or voice input without formulating any computer language in the form of binary representation and for computers to allow communication with humans with their own language. In this thesis, a variety of methods have been used to parse sentences and identify the main items of natural queries. Statistical Natural Language Processing is one of the research topics in Natural Language Understanding.

Natural Language Understanding is starting with the corpus. A corpus stands for the body of texts or collections of documents. Multiple sources of collections named corpora [59]. Generally, closed-domain question answering works with collections of texts can be from books, manuscripts or offline-scripted sources such as electronic publications. Natural Language Understanding Methods has to understand these texts to draw a conclusion to a machine. This thesis uses general methods of Natural Language Understanding with Semantic Data Sources. Characters of data sources do not change the result of knowledge-extraction except the methods.

Statistical Natural Language Processing explores a statistical and model-based approach with corpus-driven data sets. This study will use main methods of NLP such as Part-Of-

Speech Tagging, Syntactic Parsing and supply supervised learning methods such as SVM and Logistic Regression in terms of question classification.

A question answering system that works under a restricted-domain should be good at making clear the complexities of natural language word-sense disambiguation by using methods of natural language processing.

//Combine these two chapters

5.3 Statistical Natural Language Processing

Natural Language Processing is the critical part of the Question Answering System cause of deploying natural languages to any type of queries.

Stop word removal: It is one of the most common tasks in NLP across different implementations in order to simplify the input structures given a set of rules for stop-words. Stop-words have different and unique aspect of every language, so libraries of NLP should give a new stop-word list in every different language. For example, NLTK has a large of the list for stop-words while using the English Language. This could bring the NLP a drawback that makes usability lower stop-word sources from one language to another.

Language Modelling: Language modeling defines the overall performance of natural language processing methods. Different types of applications that benefit from natural language processing utilize n-gram language models such as spelling correction, machine translation or speech recognition. N-gram defines the size sequence of a given input. For instance, one can tokenize “Could you give me the average value of sensor1 in machine1?” as “Could you”, “give me”, “the average”, “value of”, “sensor1 in” “machine1, ?”. Therefore, we can call the above-mentioned gram model as bi-gram modeling. Because every output of tokenization is parsed as a two-word sequence. N-grams does not only parse inputs with sequences but also it calculates the probability of each sequence. N-gram defines the scope of analyzation for given a specific language. For instance, if an application requires deepest language property, a natural language system should parse as small as possible to model sequences. Unigram or bigram could take more time for modeling then bigger scope needs.

N = 1 (Unigram) has 20000 parameters in order to so. Respectively, N = 2 (bigram) has 20000 ^ 2 = 400 million, N=3(trigram) has 20000 ^ 3 = 8 billion, and N = 4 (four-gram) has 1.6 x 10 ^ 7 (referans gerekebilir). Apparently, the more n-gram model we have, the more complex system a question answering system or natural language processing tool need to solve.

Furthermore, an input can be dispersed to bigger sequences but the context of modeling would be messy. So one can say about language modeling is very relevant to application-specific. By using the chain rule formula, the n-gram model predicts the conditional probability of the next word [60]. As depicted in Figure 5-1, language modeling can be estimated with Maximum Likelihood Estimation.

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

Figure 5-1: Maximum Likelihood Estimation [60]

For instance, "I would like to know where the error is. " represents a probabilistic method as Maximum Likelihood Estimation with P("I") x P("would | I") x P("like | I would") x P("to | I would like") x P("where | I would like to know") x P (the | I would like to know where") x P (error | I would like to know where the") x P(is | I would like to know where the error"). The main problem of this approach is to calculate the long-chain probability of total length. As the size of sentence grows, a system needs more processing time for the calculation of probability.

On the other hand, the Markov Model can say the last few words affect the order of next few words. Broadly speaking, Markov Assumption interests n – 1 number of words in an n-gram model but this assumption does not concern from that further. N-gram language models help the creation of corpora. While creating a language model, testing and training data sets evaluate the correctness of language model. In the practical implementation, libraries assess the corpora through the n-gram model so that the libraries can produce better results in the statistical natural language processing.

Therefore, the next question is about natural language processing how to evaluate n-gram language modeling. Extrinsic and intrinsic evaluations mainly used in the phase of evaluation for language modeling [60]. The extrinsic evaluation stands for end-to-end testing by performing all the system functions over again. For example, if we want to

assess the performance of a language model in a software library, the system can be performed multiple times to see the results. However, it takes enormous time when a corpus is big enough especially, four-gram or further. Intrinsic evaluation separates the data set into a training and test set.

//Perplexity used for evaluation of language models.

Test set tells how the given model predicts the results well. The more results truly predict the lower perplexity a natural processing system can get. Broadly speaking, lower perplexity can be a better model. As shown in Figure 5-2, perplexity shows an inverse probability of a model. At some conditions, dividend goes to zero value in case that a test set could not be matched in a training set. In this case, perplexity cannot be evaluated. Additionally, a machine learning approach always suffers from overfitting issue. If a training phase were occurred more than average, a system would not give the right results given a test set and behave like generalizing every test set.

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Figure 5-2: Perplexity formula of a language modeling [60]

Normalization Process: Normalization used for eliminating low-level frequent usage of the words with regard to applications. Unnecessary words are supposed to be eliminated seeing the way clear to doing a less-overloaded application. Every corpus or any data sources should be refined before a process has implemented in natural language processing.

Stemming and Lemmatization: A normalization process cleansing unnecessary prefix, suffix or other morphological appendixes. Subject-predicate-object should map onto noun-verb pairs in order to create a SPARQL query. If a predicate has a prefix e.g. “lf:contain”, given verb are cleared surpluses by implementing a lemmatization. Normally, stemming can cleanse suffix, prefix or influx from nouns to reach the pure version of a word. However, a restricted-domain question answering has special words with

suffixes that can have a different meaning for the system or these special words can belong to a different hierarchy of a tree. So a lemmatization and wordnet synonym analyzation have been enforced on verbs.

Part of Speech Tagger: A sentence consists of a couple of structure including words like noun, verb, pronoun, preposition, adverb, conjunction, participle and article that are main categories of part of speech processing [60]. Part of Speech Tagger mostly uses a Markov Model that is a part of statistical natural language understanding. Markov model stands for a state can depend on a previous step but there is no dependency on states of historical steps more than one. For instance, a noun or a verb tells us about its neighbors, e.g. nouns are preceded by determiners, adjectives, verbs [60]. For example, a chess player makes a movement according to the last movement of a rival rather than guessing from the first movement of the rival. In this step, pre-saved corpora which has a million words has to be tagged by POS Taggers. One of the common list that has an identifier for POS named as Penn Treebank. A treebank used for annotating syntactic and semantic structure of a sentence with million words of part-of-speech tagged text. Selection of a corpus equally important to achieve a result with a parsing process.

A major concern of the Penn Treebank is to provide multiple syntactic bracketing if necessary [61]. Multiple brackets are important for example Brown Corpus tags “one” and “the one” as Cardinal Numbers but it “the one” case could be an important determiner in any sentence. Every tagger named as labels, which are clause level, phrase level, and word level taggers. However, it is important to annotate as a common noun (NN) for detecting the head of a noun phrase in a sentence. So “the linkedfactory” and “linkedfactory” are assigned as a common noun or an adjective phrase but those could be identified differently with tagger according to Markov Model of the item in a sentence.

// Explain tagger types

The simplest tagger uses a method called by NLTK library as NN_CD_Tagger which assigns a tag to each token on its basis type but it has a quite low performance because this method tagging only as Noun Phrase(NN) and Cardinal Numbers(CD) [62]. A sequential tagger could give us a better performance such as Unigram and N-gram tagger.

Parsing: When a natural query is given, a question answering system should understand the grammar behind it. POS tagger is not enough to identify a grammatical structure for complex natural queries. Relationships among noun phrases, adjective phrases, adverb

phrases, and verb phrases should be examined in order to map correctly subject-predicate-object triples in linked data. The approach of parsing separated into two main sections, which are the rule-based approach and the probabilistic approach [63]. The rule-based approach is a top-down approach to solve problems via predefined rules such as the way of Regex-parsing. Therefore, a question answering system should define rules precisely to get a correct answer. Open-domain question answering systems use this approach because of the complexity of the bottom-up approach and broadened question types. Nevertheless, a rule-based approach could give an undesirable results in restricted domain question answering or semantic question answering and could be time-wasting parse approach. The probabilistic

Syntactic parsing commences parsing sentences with chunking that is a shallow parsing without analyzing the deepest element of the parsing tree. Items can be assigned as a noun phrase and a verb phrase. In our case, this method could be practicable, for instance, "linkedfactory" keyword might be combined as an adjective "linked" and a noun "factory". If the parser would go into the deepest leaf, it would have been relatively faster operation.

Various types of probabilistic parser have been prevailed since the natural language processing research started. It depends on the grammar of the English language and how much profoundly information source that required by a question answering system. Formal Grammars of English defines a constituency parse approach, which can identify noun, verb or adjectives in a big chunk like shallow parsing. This approach eliminates of item relationship among nouns, verbs, and adjectives by providing an abstraction method. If a question answering system needs a relationship between subjects and objects, a constituency approach is not suitable to utilize because of shallow parsing. In the case of syntactic parsing, the task of recognizing sentence and its grammatical structure [60]. Syntactic parsing suffers from "*word-sense disambiguity*" problem. This problem denotes that a word can represent different meaning in the sense of location in a sentence. For instance, "*What does linkedfactory contains*" could be differentiated "Could you give me the members a factory which has linked?". Both sentences are semantically similar but hard to recognize by lemmatization and sentence similarity methods.

The method in practical implementation is partial parsing known as chunking. Parsing can check the grammar of language according to correctness. To handle the ambiguity better, the system utilizes a probabilistic parser provided by Stanford Core NLP, Spacy, NLTK, Textacy, and TextBlob libraries.

Syntactic Parsing: This is the stage of recognizing syntactic structure of inputs (sentence, keywords) by means of shallow or deep parsing.

Dependency Parser and Constituency(Syntactic) Parser: Explain Parsing with mathematical methods

Spell Checking and Abbreviation Correction: Spell checker is an evaluation criterion for restricted question answering system. It is not necessary to provide advanced spell checker controlling all aspect of morphological, semantical and syntactical rather preferring at least a simple checker. Industrial based spell checker is hard to implement due to some restrictions such as … .

As for abbreviation correction, it is difficult to find an acronym because of punctuation at the end of the acronym. Domain dependency could be another issue such as computer science, medical, or currency domain. Types of domain mainly used in open-domain question answering system due to a variety of questions. To infer an acronym, a system expects to utilize a well-formed dictionary overlapping the domain of semantic question answering. A smart factory entirely has different vocabularies and acronyms than a medical domain. In this case, the best way to classify properly an acronym using a simple look-up dictionary or hash table. A Bayes Theorem and Levenshtein Distance Algorithm would be useful to find both on spell correction and abbreviation checker. However, the spell correction gives better results than the abbreviation checker does under the Bayes Algorithm (given a result set)

Named Entity Recognition: It is a subtask of information extraction to locate and distinctive named entities with pre-classified labels such as names of people, organizations, locations, quantities etc. Named-entity recognition is a method that identifies the item of a sentence as a domain-specific. It identifies all structures mainly as a person, a location, an organization, and an entity. As shown in Figure 5-3, “sensor1” and “machine1” named as an entity and found a relation between each other. Unfortunately, the named-entity recognition of Stanford CoreNLP could not identify entity as person or organization presented by AllenNLP as depicted in Figure 5-4.

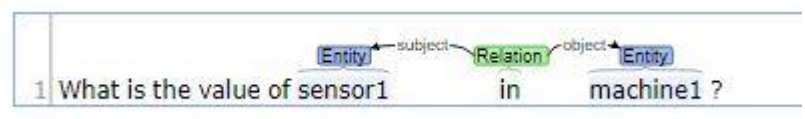

Figure 5-3: Named-Entity Recognition by Stanford CoreNLP

Figure 5-4: Person and Organization assignment by AllenNLP

This evidence shows us named-entity recognition is an application-specific task. An NER Method that is created for a different domain may not be reused for another domain. In order to create a named-entity recognition for a smart factory, a model can be trained to satisfy the requirements of a smart factory. In this context, a model can be created by statistical methods or a rule-based model. In context with the rule-based model, the character regex method can identify the structure of a natural query. For instance, a named-entity recognizer can employ a model that contains a combination of "HeatMeter" or "HeatingWater", which it can assert given items with the character started by "Heat" in a smart factory. Open Domain Question Answering

Word Vectors (Word2Vec and Glove Data): Words can be represented as vector spaces. To convert a word into a vector, the meaning of words table can be created. For instance, if we discuss two main phrases such as *"internet of things"* – "the network of physical objects with electronics, software, sensors, and connectivity", *"Mesh Network"* – "The topology of a network whose components are all connected directly to every other component", these two phrases similar in terms of frequency matrix of their meanings. "connected" and "network" are equal semantically. Therefore, one can create a word vector from corpora in order to identify word similarity. Due to the data size of corpora, a word vector can reduce the feature space of corpora.

Sentence Similarity: Sentence similarity used for comparing two string inputs in order to achieve indicative questions like "Is the system health good?". Mainly, this method leverages averaging word vectors such as word2vec or glove implementing Euclidian and Manhattan Distances or Cosine Similarity algorithm. In order to calculate distances of word, n-gram model or more specifically bag-of-words concept can be implemented. It is a subset concept of n-gram modeling. In practice, every single element is assigned into an array, for instance when comparing the following sentences:

"Is the system health good for sensor1 in machine1" and "system health status sensor1 machine1"

BagOfWords1 = {“Is”:1, “the”:1, “system”:1, “health”:1, “good”:1, “for”:1, “sensor1”:1, “in”: 1, “machine1”:1}

BagOfWords2 = {“show”:1, “the“: 1, “system”:1, “health”:1, “status”:1, “sensor1”:1, “machine1”:1}

Vector1 = [1, 1, 1, 1, 1, 1, 1, 1, 1]

Vector1 = [0, 1, 1, 1, 0, 1, 0, 0, 0]

Vectors scored as binary whether a word is exactly the same with its counterpart or not. In most cases, this sort of evaluation is unpractical because semantic structures of words are not taken into consideration. Therefore, the following methods can precisely calculate the word similarities of vectors can

Jaccard Similarity: This algorithm uses a procedure to calculate the similarity between sets of data defining as the size of intersection divided by the size of a union of two sets [64].

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Figure 5-5: Jaccard Similarity Formula [64]

Jaro Winkler: This algorithm calculates transposition of matrix t, and the number of common characters by putting into a formula as below:

$$sim_{jaro}(s_1, s_2) = \frac{1}{3}\left(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c}\right)$$

Figure 5-6: Jaro Formula [65]

The Winkler algorithm increases the Jaro similarity by means of initial characters and gives a similarity measurement [65]. For example, Jaro Winkler takes head characters of a string such as “health” and “heal” to perform the Winkler formula.

Levenshtein: Levenshtein algorithm has a variety of application areas such as spell checking, acronym finder or sentence similarity. This algorithm calculates cosine distance of given two strings and divided by the maximum value of absolute value of given two strings.

$$sim_{ld}(s_1, s_2) = 1.0 - \frac{dist_{ld}(s_1, s_2)}{max(|s_1|, |s_2|)}$$

Figure 5-7: Levenshtein Formula [65]

Wordnet Analysis: Wordnet is one of the largest databases for English lexicon that can be used for word and sentence similarity analysis. Depends on the domain of question answering, Wordnet Analysis could be used for sentence similarity or verb-noun analysis. In essence, it is a combination of two major algorithms known as Wu-Palmer Similarity and Leacock-Chodorow Similarity.

Wu-Palmer Similarity (wup_similarity): This measure calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of Least Common Subsumer [66]. With the following formula as shown in Figure 5-7,

$$\delta_{\text{Wu\_Palmer}}(c_p, c_q) = \frac{2d}{L_p + L_q + 2d}.$$

Figure 5-8: Wu Palmer Formula [67]

After defining Least Common Subsumer, which is a tree-based semantic relatedness measure extracting from "is-a" relationship of a tree. For example, "contain" and "incorporate" synsets are identical according to Wu Palmer algorithm. First of all, WordNet finds the first Tree with categories like [66] :

```
1) Tree1 = ROOT → Include → Contain
2) Tree2 = ROOT → Include → Incorporate
3) Least Common Subsumer(s) = argmax(depth(subsumer(Tree1,
   Tree2)))
4) Depth of Least Common Subsumer = depth(*ROOT*) = 1
5) Depth1 = min(depth({tree in T1 | tree contains LCS} )) = 3
6) Depth1 = min(depth({tree in T1 | tree contains LCS} )) = 3
7) Score = 2 * Depth of Least Common Subsumer / (Depth1 +
   Depth2) = 2 * 1 / (3 + 3) = 0.3333333333
```

Listing 5-1: Wu Palmer Sample Calculation[66]

Leacock-Chodorow Similarity (lch_similarity): This algorithm is very similar to the Wu-Palmer Algorithm except it calculates a negative logarithm of the path similarity. Let's give the same example comparing to "contain" with "incorporate":

```
1) Tree1 = ROOT -> <include> -> <contain>
2) Tree2 = ROOT -> <include> -> <incorporate>
3) Lowest Common Subsumer(s) = argmin(length(subsumer(Tree1,
   Tree2))
4) Length(incorporate) = 1 and MaxDepth (v) = 14
5) Score = -log(length(Lowest Common Subsumer) / (2 *
   max_depth(LCS.pos)) = -log( 1 / (2 * 14)) = 3.332204510175204
   > lch_threshold (equal to 2.15)
```

Listing 5-2: Leacock-Chodorow Sample Calculation[66]

Question Classification: A question answering system regardless of domain type needs a question classification algorithm to choose the best answer match. There are a couple of methods based on logistic regression and support vector machine that a question classification can use

Logistic Regression with newton-cg: Logistic Regression is a predictive analysis method that uses a binary classification method wrapped the combination with range [0, 1].

Logistic Regression with lbfgs: Limited Memory BFGS is an optimization algorithm of Newton-methods. We should understand what the Broyden-Fletcher-Goldfarb-Shanno method is

Logistic Regression with Cross Validation: To classify more than one categories, multinomial logistic regression method. Regression models are useful for continuous data, however, can be used when required a categorical dependent variable. Cross-validation defines the same data set as training and test data.

Linear Support Vector Classification:

5.4 Software Packages for Natural Language Processing

TextBlob: TextBlob is a tool based NLTK to process a natural query without providing NLTK's function overhead. It was written in Python 2 but also compatible with the Python 3 version. It provides a simple API for diving into common tasks of Natural Language Processing such as part-of-speech tagging, sentiment analysis, classification etc. [12]

Stanford CoreNLP: One of the fastest and robust libraries for Natural Language Processing provided by Stanford University. The only drawback of this library is limited support for Python programming language. However, it has been solved this problem with Rest – Compatible Web service by sending external HTTP queries from Python programming language. Stanford CoreNLP works based on Java Virtual Machine so that it can be conceptualized as model-view-controller pattern. As shown in Figure 1.1, Stanford CoreNLP supports diversity of implementation that is a vital role for natural language processing. Stanford CoreNLP provides an API which annotation-based that suitable underlying models or resource are available for the different languages [20]. The main drawback of CoreNLP is that one needs to use other programming languages except for Java by wrapping up Java compiled packages to specific languages. This reduces supports of full-feature such as sentiment analysis, dcoref, regexner [13].

[12] https://textblob.readthedocs.io/en/dev/

[13] https://github.com/Lynten/stanford-corenlp

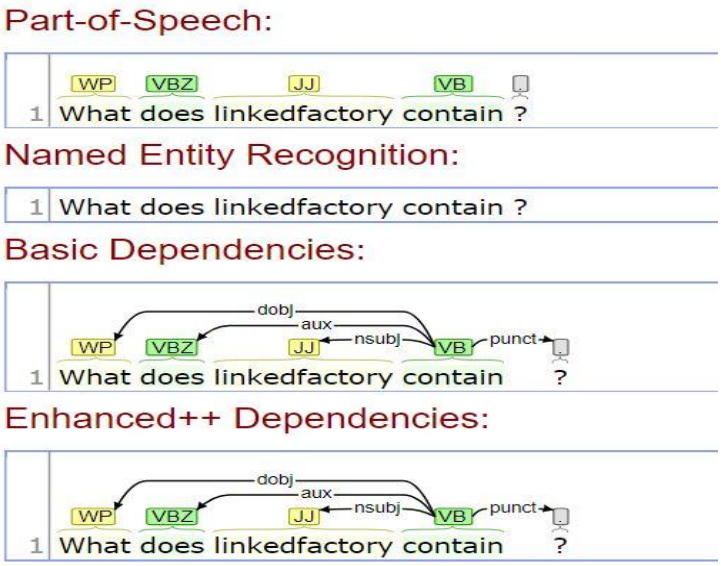

Figure 5-9: RESTful NLP of Stanford CoreNLP

Spacy: It is an open source library for Natural Language Processing which is written in Python and C-counterpart Cython [14] It utilizes the convolutional neural model for tagging, parsing, and entity recognition to increase the precision of findings in natural language processing. When a request is sent to spacy, it calls a language pipeline, which it is brought into line tokenizer, tagger, parser, and named-entity recognition respectively.

AllenNLP: It is a scientific based NLP library compatible with Python. AllenNLP also provides a demo tool which has used in this work to demonstrate the development steps of NLP. AllenNLP has advanced features to use that not only industrial scale application but also scientific purpose tools such as coreference resolution, semantic role labeling, open information extraction or textual entailment.

SyntaxNet: It is a library provided by Google that works with a deep neural network based on Tensorflow. The main purpose of this library is to serve as a syntactic parser. Moreover, this library focuses on dependency parsing more than constituency parser.

[14] https://spacy.io/api/

Natural Language Toolkit: NLTK is one of the fundamental languages that consists of many features such as tokenization, parsing, tagging published as an open source project.

//Comparison of toolkits

Libraries	Advantages	Disadvantages
TextBlob	Low overhead while doing a natural language understanding	Only windows based service setup,
Stanford CoreNLP	Strong support for variety of languages	Central Point of Failure, Bottleneck if there is not enough maintenance
Spacy	Easy to use with web platform technologies	Dependency on Node Virtual Machine and npm package manager Pipelining creates repercussion in NLP Limited Architecture Support(64 bit OS)
AllenNLP	Large supported-features for Natural Language Processing	No support for windows
SyntaxNet	Purely Deep Learning Based Stack based dependency parsing	No backward compatibly. No asynchronous support for the language version. Python 2.x

Table 5-1: NLP Toolkits Advantages and Drawbacks

5.5 Chapter Discussion

6 Practical Background

6.1 Front-End Development

6.1.1 Script Languages for User Interface Development

Fundamentally, script languages are a subset of programming languages. However, a crucial step sets script languages apart programming languages, which is at the compilation level. Script languages rather interpret where implemented.

JavaScript: JavaScript is an entry point for Rich Internet Application, which provides a content-rich application to an end-user. Event handler concept started with JavaScript language. The functional programming paradigm started with JavaScript for script languages. ECMA Standard is a de facto standard of JavaScript and these properties affect other languages that based on JavaScript language features. Such features include hoisting, callback functions, promises, lazy evaluations, generators, asynchronous iteration, and advanced regular expressions.

Typescript: Typescript is an object-oriented version of JavaScript and pretty much closer to JavaScript language. According to the essential characteristics of JavaScript, it should provide a functional language that supports callback functions without object and class. Javascript libraries can be compiled within Typescript language without occurring any problem. TypeScript sets apart from JavaScript with an object-oriented paradigm, static typing, and compile time type checking. Thus, TypeScript is a statically typed language because it should compile all types in compile-time to check the correctness of data. However, JavaScript is a dynamically typed language, which a dynamically typed language should have an interpreter layer, not a compiled one. TypeScript can implement object-oriented paradigms such as interfaces, classes, abstract classes or objects.

CoffeeScript: It is kind of similar context with JavaScript but it gives a concise and compact structure when compared to JavaScript. CoffeeScript reduces coding time thanks to short-cut version of its language property but the drawback of CoffeeScript is that needs a step to compile from CoffeeScript to JavaScript. After conversion to Coffescript, it makes look complicated than Javascript because of pre-processor codes. Basically, Coffeescript reduces code complexity (Counterpart stuff) but it makes a harder syntactic

structure for a JavaScript Developer. While Coffeescript is suitable for small module development which can easily be integrated into a bigger module, JavaScript and TypeScript can handle with a large scale of a code base. It makes available to enable an object-oriented script development like TypeScript contrary to JavaScript language.

PureScript: It is one of the largest cross-compiler support as compared to the last three. PureScript aims to be a language in front-end technologies. Large scalability support is one of the fundamental features of PureScript. It can be used in multiple operating systems, C and Field Programmable Gate Array. As compared to other script languages, PureScript has the largest scalability support for multiple platforms. This script language is based on Haskell Functional Programming Language. PureScript support polyglot programming, for instance, a core part of an application could be written in PureScript as well as JavaScript could be used for another module. By using Records can be handled with more complex Object structures in PureScript. PureScript can modify Classes and instances with keywords with "class" and "instance" by creating advanced typed-data. The type definition is more enhanced than JavaScript because PureScript can preserve data with keywords and it is an *indentation-sensitive,* unlike other script languages. It provides worker threads, which is reducing the number of threads by putting all into a pool mechanism in order to use in case of need.

6.1.2 Front-End Frameworks

Angular 2: Angular 2 is an extension framework that develops a variety of properties of Angular JS. Libraries of Angular 2 are not suitable to work legacy usage, so Angular JS Framework cannot use any library from Angular 2. The main reason for the underlying framework has been written in TypeScript, not in the JavaScript library. Angular 2 has a new command line feature that extracts essential information from "package.json". All sort of libraries is saved into "package.json" to detect discrepancies between versions of libraries. Angular 2 allows embedding dynamic bootstrapping features into a pure HTML Page. The biggest drawback of Angular 2 is that is not backward compatible and the differences between versions can be immense. While Angular JS follows the pattern of MVC, Angular 2 implements a component pattern. Besides, Angular 2 splits component by component to increase code reusability and implement the object-oriented paradigm in script languages. Angular 2 is faster than AngularJS versions because Angular 2 uses different hierarchical dependency for each module. When a module updated by a developer, only regarding hierarchy are updated so that the front-end framework can

enhance the running and compilation performance. A developer can use an external asynchronous event library in AngularJS, but Angular 2 provides an internal library implementing an asynchronous feature.

Ember.Js: Ember.JS is a JavaScript MVC Framework that helps to organize large web applications. The structure of Ember.js is based on micro-libraries [72]. MVC pattern fully complies with Ember.Js in terms of bindings, computed properties and automatically updated templates [72]. Bindings enable the change of a variable propagating to another variable. Computed Properties and Automatically Updated Templates ensure the framework stay up to date with regarding data source of Ember.js. One of the major advantages of Ember.js is Ember Data Library which stores all values of a process by means of caching into an In-Browser Store [72]. Ember.js supports all end-to-end testing tool such as Karma and Mocha. Testability is an important step to develop bug-free codes so that one can state Ember.js has a variety of compliance with test tools.

The main purpose of Ember.JS is to support a Single Page Application, thus it has no architectural layer for server-side rendering. Server-Side Rendering is an old transfer technology for HTML Websites and brings a big overhead in case of minor changes. In addition, Server Side Rendering works with static sites that need to load the entire structure of web pages. However, the initial page loading time of Server Side Rendering is shorter than Client Side Rendering does. Ember.Js is fully backward compatible that means one can use a function from an old version in a new version.

React: The React Framework serves the purpose as a full-viewer of a front-end library. React is primarily concerned with the view aspect of UI and it is not suitable to use as a framework or library in a large-scaled application [73]. React does not enlarged support for the following necessities: HTTP Calls, Routing, Dependency Injection are robust components when implementing a Web Service, so React cannot be taken into account a good solution for full-scale web service but the viewer. This could be a big drawback while comparing with AngularJS framework. React has been posited that front-end developers can leverage its features to create the part of a viewer in MVC. React does not follow the MVC Pattern. More likely known as component based architecture, it is traditionally different from MVC pattern

//Component based architecture

//React.js focused on the view of part of the model view controller

MeteorJS: MeteorJS is an open source project which has built on a stack of MongoDB, Node.js, Angular, and Express.js have consistent client-server applications, reactive modules, and rapid prototyping [74]. The underlying structure is based on Node.js and its virtual box named Google V8 Engine. The underlying mechanism of MeteorJS detects the changes of the object and automatically set the results before a developer made. Angular2 and React have observables to ensure this set of property.

VueJS: VueJS is a frontend framework that has a similar grammatical structure to ReactJS and AngularJS. Templates are one of the powerful features that used by VueJS. By means of templates, the VueJS provides data bindings. Templates support two-way data binding, that means when you changed an input, VueJS will update the corresponding element. After combining VueJS element with HTML, every element of VueJS will be reactive, which inputs are rendered immediately accordingly. VueJS is a component-based system that the abstraction mechanism of language works with components. Methods can be called in VueJS through cached memory. Thus, a cached method is not compiled in multiple calls so that a VueJS application can reduce the memory complexity of method calls.

Feature	Angular 2	React	Ember.js	MeteorJS	VueJS
Dynamic UI Binding	B2	B3			
Reusable Component	+	+	+		
Routing	Async Routing				
Data binding		State binding	One-way bindings	Template Binding	Two-way bindings
Performance					
Feature Advantage	Object-oriented script development, Independent Library Dependency				

Dependent Pattern	Model-View-Component	Component-based Archute

Table 6-1: Script Languages

One-way bindings only propagate the changes into one single direction. Two way data binding allows to implement data flows two directions.

6.2 Back-End Development

The following section is a brief description of back-end development process in terms of framework that has used in experimental development of OPC UA, Address Space Mapper for Semantic Data and Semantic Question Answering. All of these development cycles are examined with comparison between frameworks, languages, libraries and toolkits. Regarding OPC UA Web Application, frameworks, languages, and software toolkits are taken into account. As far as Semantic Question Answering and Address Space Extractor of OPC UA will be taken into account libraries and their performance. Software Development Kits was split up open source and commercial kits due to licencing co

6.2.1 Software Development Kits for OPC UA

As commercial or open source software development kits (SDK), OPC UA world provides many opportunities to developers. In this part, we will evaluate those SDK in terms of speed, scalability, and supported features. Toolkits or Software Development Kits are fastest way to produce a prototype with wide-range of programming languages.

UA.Net Standard OPC UA Stack [15]: The implementation of UA .Net Standard based on C# language supported by Microsoft Incorporate. Architecture of Net provides a scalable solution between languages that use this architecture as well. Thus, Microsoft Company targeted a great uniformity among high-level languages such as Visual Basic, C#, C++

[15] https://github.com/OPCFoundation/UA-.NETStandard

etc. .NET framework provides a virtual machine to establish garbage collection, security, exception handling, low-level (bare metal) thread management and asynchronous messaging based on Events. Common Language Runtime (CLR) is a set of technologies support implementing Web Services, Web Forms and Windows Application. The Standard Platform of .NET endorses intermediate language such as Java Runtime Environment in order to obtain cross-platform support. Furthermore, by using UA .NET Standard, all applications can be ported to cloud architectures (Azure, Google Cloud, and Amazon Web Services) which has established by software companies. Many samples can be used to develop a new application such as Global Discovery Server, Simple Server and Client [68].

Free OPC-UA and Python-OPCUA [16] [17]: A python implementation of OPC UA Stack can be used various types of development. This work uses a library of Python-OPCUA to fetch address space model into an extensible mark-up language. Later the extensible mark-up language that we generated is using for creating a semantic data file such as Turtle, RDF/XML, N3. Another version of Python-OPC UA Library is Free OPC-UA which is written C++ at the base level and it works with Python language on above levels. The main advantage of this method is performing tasks in an effective manner as compared with that of the Pure Python (Python-OPCUA) library.

Open62451 C ANSI Stack : It has been developed with old C standard which named C99 ANSI Standard. According to features, a developer can use this library as per his/her requirements such as multi-thread management, event-based architecture, publish/subscribe or cross platform support [69].

Node-OPCUA [18]: This library uses extensive asynchronous library support that presented by event-based threaded programming language named NodeJs. It supports variety of features which are essential on any OPC UA Stack. Main advantage of the library is fully-integrated Node.js library and it might be used with other asynchronous library such as JavaScript-based Promises library. Beyond as an extension of JavaScript language, Node.js uses also a virtual machine provided by Google. This library has an extracurricular characteristic is being an event loop based thread management. In Node.js, only a single thread works, however, this single thread creates multiple event loops by

[16] https://github.com/FreeOpcUa
[17] https://github.com/FreeOpcUa/python-opcua
[18] https://github.com/node-opcua

means of event triggers. Asynchronous communication is widely considered to be the most important infrastructure for client-server communication.

Eclipse Milo: Eclipse Milo is a Java Toolkit that provided by Eclipse Foundation. It is currently supporting Java 9, which is the last version of Java Virtual Machine. Except vendor-specific types of nodes, Eclipse Milo supports main types of nodes such as Object Nodes, Variable Nodes and View Nodes.

Unified Automation OPC UA Software Development Kit [19]**:** Software Development Kit provided by Unified Automation is a commercial and restricted to use core function without commercial licence. The central problem of using commercial licence is not suitable for research purpose.

Quick OPC [20]: It is a type of commercial toolkit works with a wide range of programming languages. It also supports for asynchronous and synchronous reads and writes internally, subscription and browsing for access details of nodes. The commercial licence of Quick OPC is more suitable to survey a research than other software development kits made.

Prosys OPC UA Development Kit [21]**:** Prosys is another commercial toolkit which provides variety set of tools to implement OPC UA Server, Client and Discovery Server. The pitfall of Prosys tool is that the licence requirements are very strict which could be intervene the development process. Prosys company provides a stack that is interoperable OPC UA communication.

Matrikon OPC UA Development Kit: This is a commercial development kit produced by Matrikon. Matrikon has limited support for development kit

Tookit Name	Advantages	Disadvantages
UA .Net OPC UA Stack	B2	B3
UA .NET OPC UA Legacy	A1	A2
FreeOPCUA and Python-OPCUA	C2	C3

[19] https://www.unified-automation.com/
[20] http://www.opclabs.com/products/quickopc
[21] https://www.prosysopc.com/

Open62451 C ANSI	It is fast because of C language.	Platform dependency
Node OPC UA	Asynchronous based OPC UA Server	E2
Quick OPC UA	D1	D2
Eclipse Milo	N	
Prosys OPC UA	A set of tools to implement OPC UA Server/Client and Discovery Server	Licence requirements very strict
Unified Automation OPC UA SDK	C1	Low performance of OPC UA Server under async tests.
Matrikon OPC		

Table 6-2: OPC UA Software Development Kits

6.2.2 Architectural Decision for the Web-based Software

Web services apply two types of communication method where a communication endpoint is reached. When a client invokes a web service synchronously, the client application should wait until it gets a result from the web service. Otherwise, the client application fell into a timeout of a session or frozen application. There is a solution for those issues named asynchronous communication. In this way, a user can send a request to a web service without waiting for the result of queries. The Asynchronous communication is implementing by multi-thread management, finite state machine or event-based callback functions. Event-based callback functions are the fundamental elements of script languages such as JavaScript, Typescript etc. callback functions have a nature of being asynchronous. Asynchronous features have more overhead than synchronous features. If a developer wants to implement an asynchronous web service, the framework should save the state of each request to continue from which a service left.

In this work, synchronous communication for a web user to get a JSON Web Token authentication access. When a user has the right access, all requests are sending in an asynchronous way. Due to the nature of JSON Data parsing and Publish / Subscribe architecture, the usage of asynchronous requests are necessary for the return of requests inner architecture.

//asagidaki paragraph baslikti birlestirdik

With regards to design patterns, the Model-View-Controller (MVC) one of the architectural pattern used for building up a web service. Basically, the client application is a view, back-end application which provides details of Rest Interface and Data Mapping is a controller and database application is a model [22]. The web application composed of view and controller purely and partly provides a model. The model has been used in the work handled the way of in-memory mapping. ASP.NET Core MVC has model binding, routing, dependency injection, filters model validation

ASP.Net Core MVC and Java MVC comparative study are provided in this work by explaining the afore-mentioned context such as model binding, routing etc. ASP.Net Core. Model-View-Controller has comprised a Model, a View and a Controller used to separate the application's logic.

Model: This layer encapsulates business logic and data. In computer science, business logic is the part of a program that encodes real-world requirements in terms of creating, reading, and updating. All of the items have dynamic nature in an application so that other layer of an application may concerns changes that are presented by a model.

View: This layer demonstrates a view of the modelling of data presented by the same data. It also closely relevant to visualization, beyond that it has a purpose for showing multiple views regarding the same data modeling.

Controller: This layer acts on both the model and view. It also copes with data changes and provides an endpoint for a view to visualize data's content. Main features of ASP.Net Core used with Controller in this thesis and the Controller base class for an MVC Controller with view support [23].

The aspect of architectural view, a system can comprise of monolithic or microservice architecture. Most of the applications apply monolithic architectures, which are (monolithic acikla) , (microservice acikla)

//Load Balancing burada aciklanabilir

[22] https://www.futurice.com/blog/api-services-mvc/

[23] https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controller?view=aspnetcore-2.1

//Async-Sync kismi buraya alinabilir.

6.2.3 Back-End Frameworks and Languages

ASP.Net Framework (Active Server Pages .NET): ASP .NET Framework one of the oldest framework used by developers to implement web applications. The oldest framework named as ASP.NET Web Forms. ASP.NET Web Forms was strongly dependent on Windows Operating System because of Internet Information Service (IIS). This server used for deploying web applications that can work only within Windows Operating System. Moreover, this framework was limiting changes due to an internal file of Windows Operating System as known as Web.dll [70]. Web Forms evolved to ASP.NET MVC Framework to comply with Model-View-Controller design pattern.

ASP.NET Core: Besides continuous improvement of this technology, Microsoft Company decides to scale this framework Unix-based architecture. Therefore, the name of the technology changed as ASP.NET Core, which brings to the developer worlds lightweight features. One of the most prominent features of ASP.NET Core is the routing framework to control Rest API calls. When an HTTP call arrives, it should be parsed as schema and host path. Schema path decides which protocol used an underlying structure to deploy a call. An aspect of the query string to understand specific element, host part contains path and query structure to discriminate an HTTP call from each other. ASP.NET Core mainly used for a production environment because of the immature step of development like ASP.NET Framework (Think about). To deploy a web application rapidly, ASP.NET Core is a better choice thanks to its lightweight functions, the code size of the virtual machine, open source code, and interoperability with Unix-based operating systems. Moreover, ASP.NET Core has legacy support with ASP.NET MVC and Web Forms so that the framework can extend internal functions with legacy projects.

Node.js: Node.js is a framework based on JavaScript language that leverages a virtual machine developed by Google Inc. [24] Node.js is an event-driven server-side development framework. By leveraging a virtual machine named V8 Engine, the framework sends an event signal to the virtual machine rather than communicating an operating system itself. Node.js has a broader support for multiple operating systems because the virtual machine has been compiled for multiple targets. Not only the framework is compatible

[24] https://www.w3schools.com/nodejs/nodejs_intro.asp

with a client-side script language such as JavaScript, but also it can integrate callback functions with low-level compiled language such as C++ or C. This paradigm has named as Native Call in the software world that is very useful when a function result returned from a programming language managed by a virtual machine into a native language. A finite state machine implemented by C++ creates asynchronous callbacks until the objects of callbacks are eliminated by a garbage collection.

Java Spring Framework: Java Spring Framework is the closest architecture to ASP.NET Core in terms of package management, virtual machine based garbage collection, routing and dependency injection. It has an object mapper such as Entity Framework in ASP.NET Core that ability to connect with databases mapping object into database objects. By supporting modular development, the code can be split into modules and it is getting easier to handle with code size when a project source code's size increased. Spring Framework work with Plain Java Objects.

Flask Micro-framework: Flask is a micro-framework using by many software developers benefits of rapid prototyping. For the reason that we need to develop a rapid-prototyped solution, Flask helps developers in many aspects. The Flask is working with many versions of Python 2.x and 3.x. There is no complicated routing mechanism and completely compatible with microservice development. Annotators are bounded but compact, which is making the learning curve of the framework higher. A Flask Application can ensure JWT Authentication and other security policies with external libraries. In the research phase, we faced that the biggest problem of Flask is a non-asynchronous structure. One of our proposal to remedy the problem is using a non-blocking input-output queue with an asynchronous task queue. A non-blocking input-output queue can create an internal load balancer by balancing all requests into a queue before it reached to the application. It is far more than making a routine asynchronous. When an asynchronous queue works, it selects a message broker to connect a non-blocking input-output queue. This overall system creates an internal "message-broker system". To change simply version from 2.x to 3.x would be the second solution. Because Flask application does not have the "async" keyword that makes the routines asynchronous call.

Django Framework: Django Web Framework is a loosely coupled, high-level Python Web framework along with supporting Model View Controller pattern. It leverages the language property of Python allowing indented programming and implicit data types. Underlying pattern is a bit different from MVC because the view part could present

views through templates. The framework operates across multiple tiers such as Business Logic, Application, and Presentation Tiers. Django uses a special template to fetch iterative values from template engines and it is believed that the templates shorten the code complexity of Front-End. Django has a package manager called "pip" to organize libraries in a virtually separated folder. Main issues about Django are not occurring from the architecture of the framework, rather it is associated with versions of Python 2.x and 3.x creates discrepancies among libraries. Routing Mechanism provided by regular expressions with precedence rules. When an URL is matched, all other requests are dropped in accordance with precedence rules. Django can consolidate URL Patterns by including a URL one into another. In this way, developers can easily manage the URLs and HTTP Requests within a single base URL entry point.

In the following Table 6-2, the thesis examined with a couple of parameters such as Performance with Multiple and Single Queries, JSON Serialization Fortunes etc. It has been referred to platforms, micro-frameworks, and full-stack frameworks as "frameworks" [71].

Criteria	ASP.NET Core	Spring IO	Django Framework	Node.js	Flask
Multiple Queries	46.4%	13.6%	%3.63	24.9%	21.7%
Latency of Multiple Queries	0.5 ms	80.9 ms	287.8 ms	44.4 ms	226 ms
Platform Support	All Platform	All Platform	All Platform	All Platform	All Platform
JSON Serialization	80.8%	%8.7	%10.8	%46.7	12.2%
Latency of JSON Serialization	0.6 ms	4.8 ms	5.8 ms	0.9 ms	3.8 ms
Single Queries	54.9%	12.8%	3.7%	28.7%	10.8%

Latency of Single Queries	0.5 ms	3.8 ms	1.4 ms	0.4 ms	3.5 ms
Plaintext Query	99.7%	2.3 %	2.1%	12.7%	4.1%
Latency of Plaintext Query	1.4 ms	397.7 ms	24.1 ms	65.9 ms	45.8 ms
Compatibility	Backend Compatible with minor versions	Backward Compatible with minor versions	No Backward Compatibility between Python 2.7 and Python 3.0	Backend compatible with major versions	No Backend Compatible

Table 6-3: Backend Development Framework [71]

The test suite provided by TechEmpower [71] contains multiple query tests, single query tests, Plain Text test, JSON Serialization tests. JSON Serialization test includes request routing, request header parsing, object instantiation and representation, the creation of a response header regarding a request [71]. Single and multiple queries sent against the database to test connection pool and performance of input/output given a set of a framework. A simple SQL query is prepared regarding context and header with aid of an HTTP POST request. Instead of JSON query, a plain text such as "Test Framework" sent by a request header with content through HTTP 1.1 at different concurrency levels [71]. The higher scores of percentages show how a framework perform better under equal circumstances. In the same way, the latency is a parameter showing the duration of the delay within consecutive queries. In this test, each request is processed to fetch a single row from a database and the data is serialized as JSON [71]. Hence, low latency gives better performance for back-end frameworks.

6.3 Practical Implementation

6.3.1 The RESTful Implementation of OPC UA Web Application

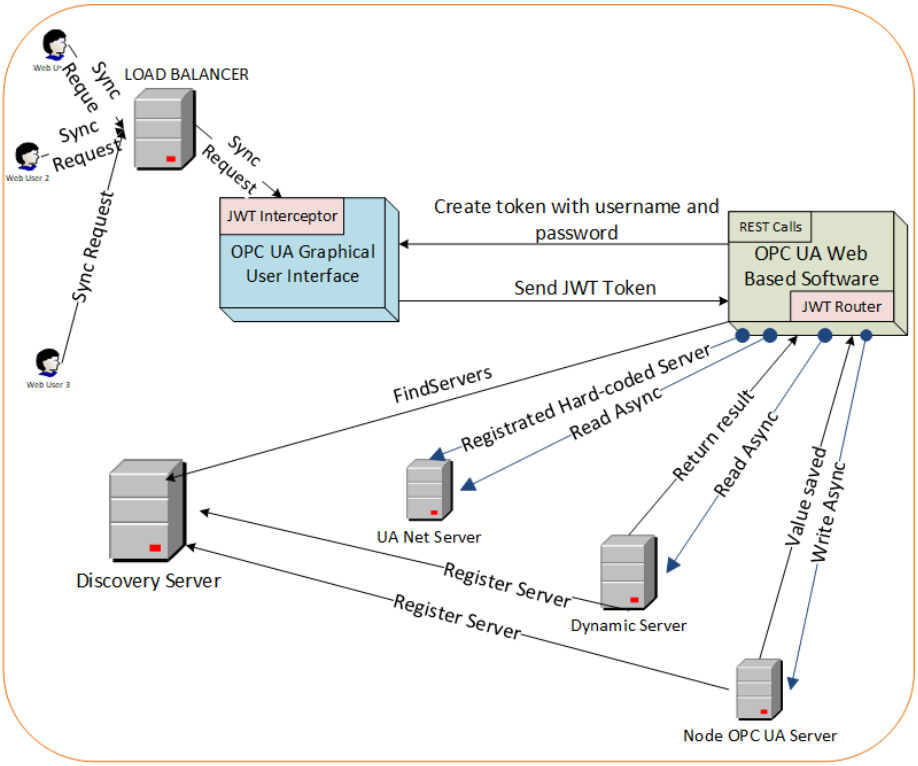

Figure 6-1: General Architecture of OPC UA Web Application //Add Question Answering

At this chapter, OPC UA Web and Integrated Semantic Question Answering Architectures are giving to examine structure that comprises software elements and relations among them. JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact way to transmit information among generated JSON Objects. Before all requests taking from web users, a load balancer can balance the volume of requests and split up the resource of the system regarding queries. The practical work of thesis provides a load balancer to give an ability to assigning multiple resources into equal space of cores regardless of the domain and scope of the web-based software. The architecture resembles a monolithic application that contains all modules connected to one single load-balanced endpoint, unlike Microservices. This thesis uses the approach of API

gateway which improves the usability of libraries contains a more complex structure with a simple API entry point.

As shown in Figure 1.1, the architecture of this thesis comprises an authentication mechanism, RESTful service handler. OPC UA Protocol handler and Semantic Question Answering. OPC UA web-based software can support every type of data structure which is used in communication stack defined by an XSD document.

ASP.NET Core commences the API entry point with "API" keyword. All regardless sort of requests is sent with "api".

```
[HttpGet("api/authenticate/) Body of Request {username, password}]
```

Listing 6-1: HTTP Get Request for token-based authentication

As show Listing 6-3, the system has a get request for authentication with user name and password. The practical implementation incorporates a hard-coded username and password, but ASP.Net Core can implement a temporary username-password pairs with an in-memory database. Authentication HTTP Request is an initial point to get access from API Gateway. A sample request of authentication as shown Listing 6-3. After matching username-password pair, a JWT token is being created to direct other requests into a controller. A developer can define the JWT Token for a short duration unless the time to live value (TTL) of server expires. Other routings of ASP.NET Core are protected with ASP.Net Core Headers such as [Authenticate] and [Allow Anonymous]. A malicious request without a proper JWT token cannot be sent that way. After an HTTP Request authenticated with a token, the request bypass the Header named [Authenticate] by letting the request anonymously.

An HTTP GET request for node information has been reformatted again from the work of [Scroppo Et al., 2017] [7]. In this work, the expiration time of a JWT token restricted with minutes; however, when the practical implementation has been tested under heavy load testing, a point of failure could create a bottleneck if JWT token authentication would have expired within minutes. So expiration date of the JWT Authentication has been prolonged in the practical implementation. In particular, a node-id is a mandatory field in the request as below in Listing 6-2.

```
[HttpGet("api/serverconf/DataSetID/allnodes/{node_id}) Authentication Bearer
{JWT}]
```

Listing 6-2: Http Get Request [7] [6]

To connect OPC UA Servers, a client can use

//Talk about the way to connection of OPC UA

//Websockets, TCP, HTTPS

//Websocket needs high rate of polling to stay the connection up rather than having a repetitive connection setup for each request. Good match for real time

//Mixed connection with HTTP and HTTPs are vulnerable to attack. HTTPS never be cached. When an HTTPS used between browser and server, proxies cannot see the shared cache, which can be dangerous for the communication environment. Encryption has overhead for both server and client. Certification creation overhead between client and server. However, OPC UA CA sharing solves this issue

//TCP connection provides a basic level connection without overhead.

The web-based application is capable of writing a value into a writable object. This is a limited feature against servers because most of the OPC UA Servers have some restrictions in order to protect from vulnerable attack. However, simulated data in any object can allow writing in the structure for testing purpose.

```
[HttpPost("api/serverconf/DataSetID/allnodes/{node_id})  {value:”message”}
Authentication Bearer {JWT}]
```

Listing 6-3: Http Post Request [7] [6]

```
[HttpGet("/integratedstaticmessage/{question} Authentication Bearer {JWT}]
```

Listing 6-4: Question Answering Static Message HTTP

```
[HttpGet("/integrateddynamicmessage/{question} Authentication Bearer {JWT}]
```

Listing 6-5: Question Answering Dynamic Message HTTP

//Talk about load balancing with NGINX and RabbitMQ non-blocking IO queue.

HTTP Request can be repetitive and consecutive methods triggered by a user. Taking into consideration an increasing amount of data in industrial networks on the daily basis, an architecture should comply with the load balancing and non-blocking input-output queues.

6.3.2 Authentication and Authorization of the Web-Based Application

A Web Service must have compliance with an authentication standard anyhow. One can mainly observer two kinds of authentication, which are certificate-based authentication and token-based authentication. Authorization is a higher-level representation so that one can implement a role-based authentication after authenticating a system. These roles can be broken into administrative and user roles. A system can assign different rights to these roles in order to provide a system's security or integrity. OPC UA Protocol introduces a certificate-based authentication before establishing a session. A Web-based software may perform security between its platform and end user. This called mainly API security and OPC UA Web-Based Software provides a JWT Authentication. With JWT Authentication, a token created by the back-end application of web-based software and is sent to the client side after a client performed an HTTP Request to an endpoint. A client or front-end application should send this token with every request that he wants to authorize while a process performing. The carrier system called Authentication Bearer, which is carrying out a body of a request in HTTP Protocol. A user types username and a password to get access a token to fetch data from a web-based system. After initiated username-password pair check, JwtSecurityTokenHandler creates a handler of a token and SecurityTokenDescriptor initiates a description of a token. The latter called SecurityTokenDescriptor defines expiration date and type of credentials such as Aes128, HmacSha384 or RsaSha256Signature. In our case, the practical implementation of a symmetric key has with Hmac Sha1 256 Bit Cipher.

JWT Authentication may work with Claim-Based Authentication that allows users to authenticate with claims. For instance, OAuth2 and OAuth Single-Sign-On Authentication Methods leverage Claim-Based Authentication by routing to an external layer of software. An Issuer envelopes information such as Roles, User Domain or Account Name with a token by means of an Issuer Server. In order to authenticate multiple time with the same token, a system requires an Issue Server with simple information about a

user to distinguish domain of application to be granted. In the general case, claims identify an expiration time of a token in the view of the fact that the system calculates the interval between the current value of time and token-validation time.

A compact way to provide security is to implement an authentication method within the low-level protocol area. Sessions have a variety of service set in OPC UA to create Session between a server and a client. Before calling a service set from a Server, OPC UA Client should create a session for the integrity of the communication. Firstly, a Session Service Set should provide an endpoint and a security model for constructing session management. A session can close a secure channel with timeouts to protect from an unnecessary idle state of servers. Sessions should have encrypted natively before they send information into the upper level. With respect to catering to protocol-level security, OPC UA Client employs a certificate-based authentication through an X509 Certificate to Certification Routing. This routing system prepares an authentication initiative to decrypt parameters of a certificate. At the same time, an OPC UA Client sends a secret message through Open Secure Channel initiated by a Session and message-certificate item pairs are verified with an asymmetric signature.

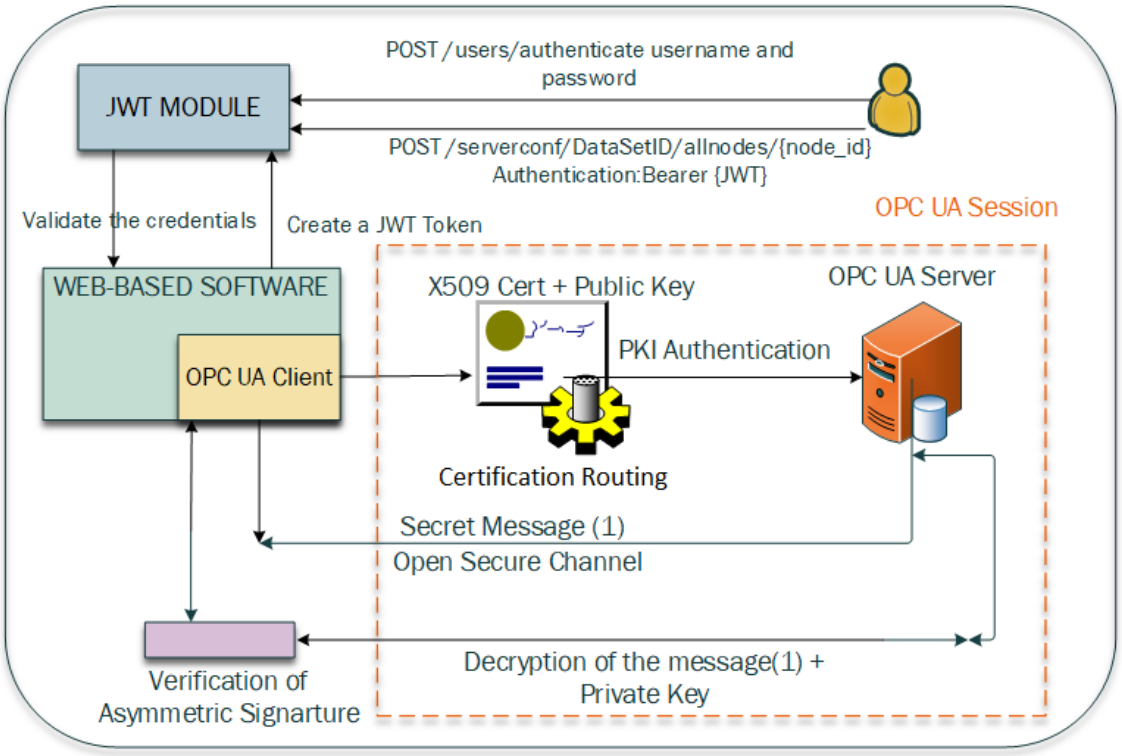

Figure 6-2: Authentication System in the Practical Implementation [75]

JWT Authentication is not the only method that a system can utilize through an authentication concept. Kerberos and OAuth can be thought other versions for the authentication and authorization concept. In common, all of three authentications employ issued tokens but there are different from each other slightly. JWT is an unloading authentication, which means that an authentication system interrogates tokens in incoming requests. Furthermore, a request that consists of JWT Token is not requiring extra configuration against endpoints. However, Kerberos Authentication needs a key distribution service and an authentication server. Initially, an incoming request sent to an Authentication Server to authenticate the request by user name and password pair. If a symmetric key used for authentication, the key distribution server should handle all requests to exchange keys instead of public-key infrastructure. This could lead to point of failure and security vulnerabilities. Another drawback about Kerberos, time limitation of a ticket does not allow time-to-live value such as JWT and OAuth2 and this time restriction must synchronize with clocks between servers. Consequently, a Kerberos system expects more configurations than JWT and OAuth2 needs.

//Talk about Kerberos, OAuth authentication and benefits, drawbacks

//These authentication methods are described in the document called OPC UA Mappings

6.3.3 Data Manipulation and Navigation through OPC UA Protocol

For the part of Data Manipulation in OPC UA, the previous study has been used that has published by [6] [7] and Free OPCUA [16]. OPC UA utilizes tree-based hierarchical architecture to traverse among nodes with their references. Folders organize Address Space and they can abstract objects into Information Model. Complex type as predefined structures should comprise primitive type that can be reachable by OPC UA Client.

OPC UA Protocol defines its own data structures that break up into two main sections: Built-In and Structured Data Structure. When an OPC UA Client demand navigating OPC UA Server, he should start from a root folder that consists of a root node. By sending a browse request to root note id that is equal for all standard OPC UA Server is id=85, OPC UA Client reach the terminal nodes of the tree structure in OPC UA Server. Generally, leaf nodes give standard information about folder, object and variables and continuous simulated data saved into the leaf nodes of OPC UA Server. OPC UA supports various data types up until top-level nodes in terms of object-oriented network design.

There might be a misconception when sent a Read Request without parameter such as "namespace = 0" and "root node id = 85". (Scroppo et. al. , 2017) stated that an OPC UA Client can send a read request without parameter, so the client will connect the object root node [7]. The practical implementation follows another way around like there is no hard-coded root node and namespace pairs for lucidity. An aspect of the web-based application, each user can send separate requests by creating a new session.

Navigation between nodes should consist of attributes and references as mandatory. Principally, a node attribute comprises the Browse Name and Node Id. "BrowseName" and "Guid" parameters as defined in OPC UA Protocol show initial names of nodes. Browse Names are matched onto Values and Datatypes. Browse Name and Display Name similar to each other except that Browse Name represents itself with a namespace index. Practical implementation exhibits a node id – namespace pairs, browse name, type of data and type of reference. However, an aggregation server can be created as a cumulative address space. All connected servers to the aggregation server create own address space in order to so. In this case, the aggregation server defines multiple root nodes with namespace and a GUID. The practical application can be extended to show multiple roots but the scope does not cover this point. Root Object has three items, which are Objects, Types, and Views. Views is a restricted address space created by an OPC UA Server. Restricted spaces have different definitions for Views that limits Nodes and References. Views are more useful when used a cumulative address space in opposition to a single address space because multiple address space should discriminate more dynamic and static data from multiple OPC UA Servers.

Serialization idea for reading and writing requests have been taken from the studies [7] [6]. The idea behind of serialization is converting OPC UA Server built-in types into JSON Schemas or Values. By way of JSON format, a web-based service can use the information through HTTP Request Payload to communicate between front-end and back-end architectures. Even though the features of built-in data such as description, binary schema, field type etc. saved as an XML schema in OPC UA Protocol, the XML format is not suitable for neither OPC UA Web-based software communication among modules nor a semantic question answering. By this means, an overhead of conversion of syntactic XML is not a problem while developing a web-based application or a semantic question answering. The approach comprises the Structured Data Type and Built-In Data Type which is converting into a JSON Format through serialization to send a proper response from OPC UA Servers [7] [76]. To navigate between Structure Types used an XPath navigation includes over 200 built-in functions for string values, numeric values,

Booleans and node manipulations [77]. Most built-in types are encoded in XML Schema of OPC UA Standard Definition. A Client holds application configurations, data types of nodes, and security information with certificates as encoded in XML Schema. Due to being a structured data type, information parsing are relatively easy with labels within an XML Schema.

As a result, our findings are that XML Schema can handle the internal operation of OPC UA such as a definition of Node Types or extracting data types and JSON Serialization is a more valuable step to interoperate with web applications. Our application of generalization can convert XML Schema into RDF/XML through XLST processor. At transport level, JSON Encoded messages exchange over an HTTP connection with a specific "Content-Type" and "Content-Length". Another finding is about that XML Schema can define clearly encoded Messages for SOA based applications. Bindings and Port Types for SOA Based applications have already defined within XML Encoding. Moreover, an HTTP Request can envelop a predefined XML Schema files such as Data Types within a body section.

6.3.4 Development of Monitoring Application in the OPC UA

```
[HttpGet("/api/serverconf/DataSetID/subscribeNodes/monitor_id) Authentication
Bearer {JWT}
```

Listing 6-6: HTTP Get Request for Monitoring Node (Should be posted)

This study handles a subscription request with a minimum sampling time interval of monitoring node id to register either a variable, an attribute, a node or an event. The Web-based application does not specify any minimum sampling time interval. In the practical implementation, one can prepare a packaged notification message with required monitored node id. The limitation of the monitored node is not all OPC UA Servers provide monitorable structure for variables or events. This restricts to follow all changes for manufacturing device and the only solution could be redesigning OPC UA Server in order to subscript changes within a particular time interval. Subscription connects to an existed session to prevent creating a redundant number of sessions. An existed session can be controlled from a common pool that has all session's identification numbers with their generic configuration.

After subscribing a request, one can fill up the subscription with a monitored node id. At this stage, the system should assign a sampling interval rate. The rate implements a cyclic rate that the server can sample data from real items. Sampling rate selection could be problematic in the implementation. A user or an inner application can make the selection. For instance, monitored part of the application select 1000 ms sampling interval, which is the most common interval rate selected by OPC UA Servers. If the 1000 ms has been selected although a server does not support the rate, the server assigns the most applicable rate in order to apply a sampling rate. The selection process may be different server by server. Regardless of a sampling rate below than a particular value or upper than that, server creates a subscription to insert a monitored node into the subscription. User can take an exception from protocol stack regarding mismatching interval rate, either way a subscription is produced. Other finding is that the underlying structure of subscription mechanism of OPC UA Servers are not thread safe. That means a client implementation should be aware while multiple monitoring node could create delay for results. Consequently,

//Why Monitoring Nodes

//How to implement Nodes

//What exactly it is

//What benefits we can get

//What are the drawbacks

6.3.5 The Design of Semantic Question Answering

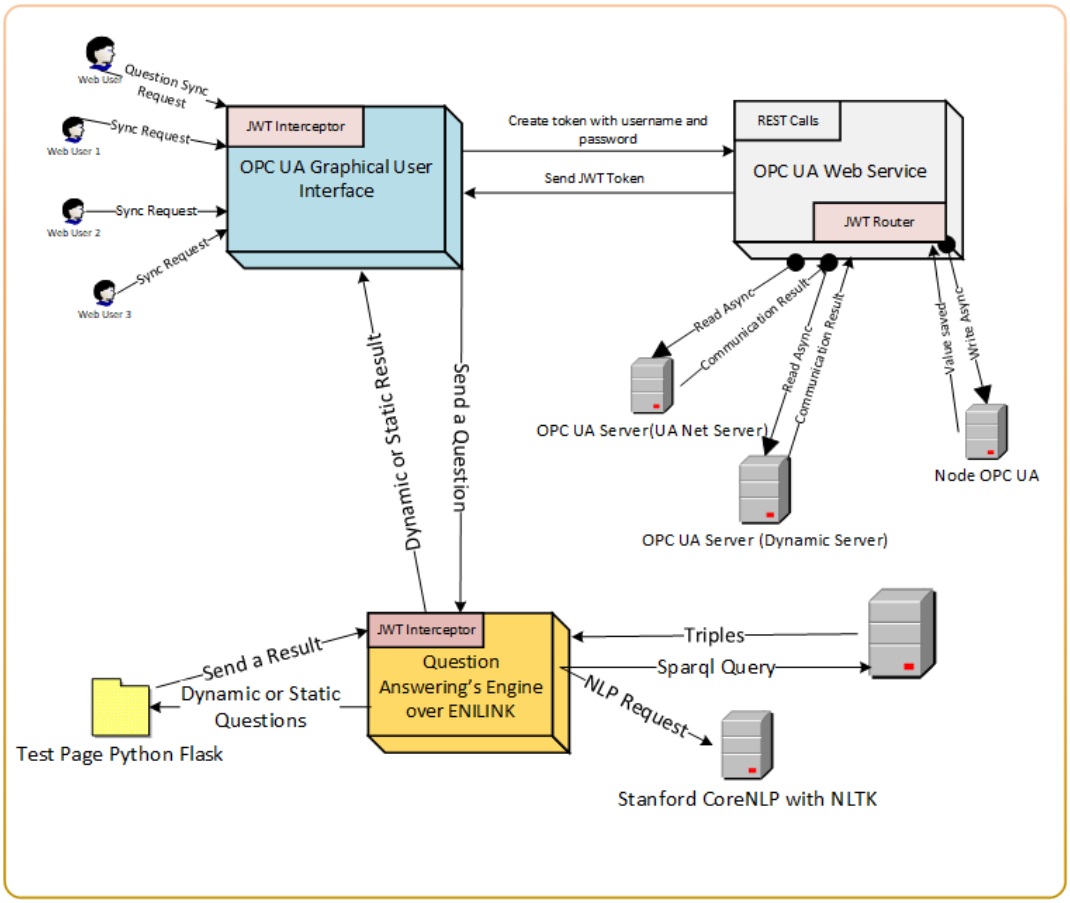

Figure 6-3: RESTful Semantic Question Answering System

Semantic Question Answering System is a detached module that complies with Model-View-Controller Pattern. In a similar manner of OPC UA Web-Service, a platform can send a rest request independently unless they have a proper token created by the service. A Test Page for Question Answering System works with the link under "http://localhost:5000". Semantic Question Answering System can take HTTP Request from OPC UA Web Service so as to have an integration through modules. Python Flask handles with all request coming from users with daemon threads. With respect to daemon threads, the system presents a multi-thread environment without paying attention to the termination of threads. For instance, when two of HTTP Request into both module OPC UA Web Service and Semantic Question Answering, the system without daemon threads should take care of termination manually. This causes a bottleneck for a system

known as multi-thread resource termination. Daemon threads make respectively stop all threads after exiting with their resource from HTTP Requests. As shown in Figure 1.1, Semantic Question Answering has multiple steps to achieve a result from its resources such as stop word removal, tokenization, lemmatization and stemming, WordNet analysis, question classification. Chapter 5.3 explains the theoretical treatise of every step and this chapter will introduce the practical implementations with different libraries. Many applications of question answering use the answer scoring method before taking answer with SPARQL queries. Mainly, natural understanding part of this thesis used a bunch of libraries such as Spacy, Textblob, NLTK, Stanford CoreNLP with annotators and Restful services. NLTK mostly used for tokenization part in order to handle inputs without an overload of library functions. Principally, tokenization and stop word removal is not memory overhead operation because these two steps only need a list of English lexicon to identify words.

We have used partial parsing and parsing based on machine learning through natural language processing libraries altogether. Firstly, the semantic question answering applies a shallow parser to identify an essential sequence in order to combine with a subject-predicate-object pattern. If such a pattern found in a given natural query, the question answering should chop the unnecessary items such as adjectives and adverbs except for nouns and verbs.

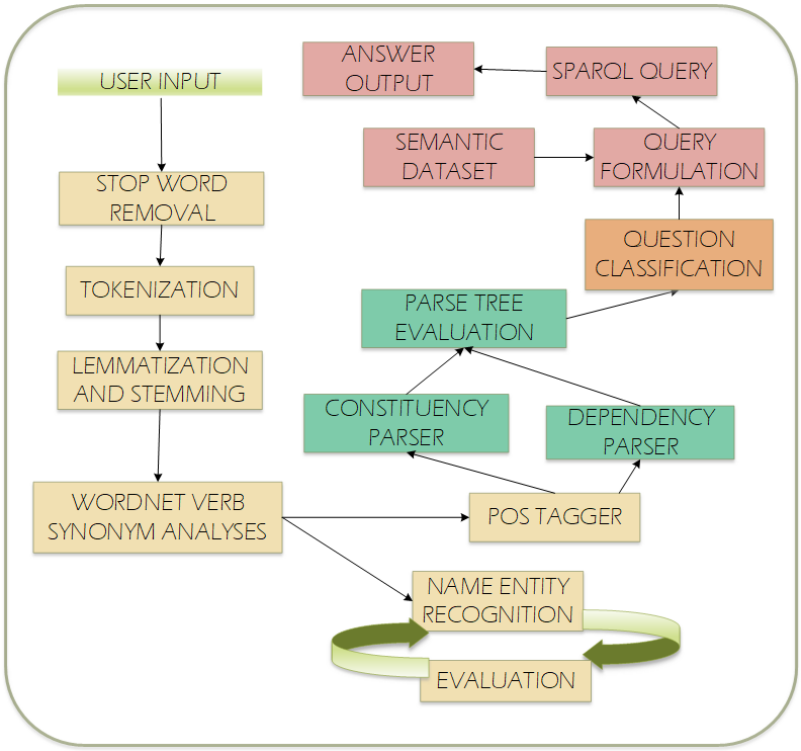

Figure 6-4: The Algorithm of the Semantic Question Answering

Every query formulation phase should follow the flowchart as shown in Figure 6-4. An input should clean unnecessary characters with stop word removal and tokenization functions. All of the rectangles illustrated in Figure 6-4 has represented in different classes in an application. The lemmatization step clears the prefixes to compare the synonym of the word. Unlike an open-domain question answering, we could not implement a rule-based approach. For instance, a template-based approach gives precise answers with a predefined set of a statement such as <?noun, property, ?object> [22]. Moreover, the question classification phase affects answer ranking. If a question answering system gets a large scale of data, the system can score answers of questions to indicate in a better way. These two aforementioned approach is not suitable for restricted-domain question answering according to our experiment. Data scarcity does not only affect implementing a machine-learning algorithm but also makes a question answering based on information retrieval harder. That means a semantic question answering that exploits restricted source should focus on a deep parsing approach. After implementing the semantic question answering, a question classification used for eliminating answer types rather than scoring the answers.

The semantic question answering employed a limited linked data represented as Turtle RDF for static queries. The statements i.e. “What does linkedfactory contain” or “Please give me all of its members” were asked to the question answering. One of our findings is that an upper or lower case of question can produce different results from constituency parser unless we turned into a lower case of them.

Algorithm 1 Query Formulation

```
1:  function QUERY FORMULATION(a, b)                          ▷ Explain here
2:      query ← QueryWithPrefixes
3:      r ← contituent.parse.tree
4:      indirectdependency ← dependendency.parse.tree
5:      while nodes ≠ leafs.terminal do          ▷ Until leaf nodes(Terminals)
6:          verbs ← PARSER(nodes)
7:          nouns ← PARSER(nodes)
8:          similarityflag ← WORDLATENANALYSIS(verbs)
9:          if StaticInformation is True then
10:             indirectdependencyFlag ← DEPENDENCYPARSER(nodes)
11:             if similarityflag and IndirectDependency is true then
12:                 object ←nouns
13:                 predicate ←verbs
14:                 query += object + predicate + ?subject
15:             else
16:                 subject ←nouns
17:                 predicate ←verbs
18:                 query += ?object + predicate + subject
19:         if DynamicInformation is True then
20:             predicate ← PARSER(nodes)
21:             object ← PARSER(nodes)
22:             similarityflag ← SIMILARITYLEVENSHTEIN(input)
23:             query += object + predicate + ?subject
24:     return query                          ▷ The last query has been constructed
```

Figure 6-5: Query Formulation Algorithm

The algorithm as shown in Figure 6-5 is evaluating the queries within two main sections. While static queries are sent against local linked data endpoint, dynamic queries need an API to get through federated services. This separation must be done by an expert when they asked. Numerical keywords such as “value”, “average”, “minimum” or

"maximum" are key points of a semantical separation between static queries or dynamic queries.

6.4 Chapter Discussion

Like all natural language processing applications suffer from word-sense disambiguity, our application could fall into a failed situation after initiating some queries.

Selection of the back-end and front-end libraries might have changed according to the requirement of a web project. In fact, different versions of a framework may create different results under certain and same test conditions. However, Chapter 6.1.4 indicated to which one displays the best performance and worse performance. Rather than comparing the performance of front-end framework, Chapter 6.2 explained the script languages and front-end frameworks in terms of modular applicability, loose coupling, and scalability. Using software development kits can improve the quality of an OPC UA Software because of the difficulty of writing an OPC UA Stack.

In order to create the semantic question answering, we have used regex-methods, POS Tagger, Parsing,

Type of domain plays a very important role to decide natural language processing algorithms where an algorithm will be used. The semantic question answering is a part of restricted domain question answering that takes linked data defined as triples. Details of an Experimental Development

//Talk about the importance of deployment while using a library for the natural language understanding.

Based on the extensive support of SDK aspect of OPC UA, Frontend and Backend of OPC UA Web Based Software can be developed in any programming language. Moreover it should be evaluated in terms of End to End Productivity, Interoperability, Versionability, Testability and Mockability, Learnability. In this chapter OPC UA SDK (Software Development Kit) will be discussed. Both open source and commercial, there are extensive support to develop OPC UA Stack and its applications. Not only SDK will

be evaluated aspect of essential properties related to SDK, and also will be assessed extensibility of SDK regarding the feature itself.

7 Discussion

- Comments on your results
- Explains what your result mean
- Interprets your results in a wider context; indicates which results were expected or unexpected
- Provides explanations for unexpected results.

7.1 Introduction

In Chapter 7, we will explain the test parameters for the general suite of Web-based application. We evaluate the OPC UA Client feature of the web-based application with some parameters, e.g. viability of unit testing, mock testing, and load testing. We separately evaluated the Semantic Question Answering with the precision of answers and usability of the question answering. Our main theoretical motivation is to provide an assessment of performance for the general system. OPC UA Client is hard to test with mock testing because creation a session at the protocol level for each request could freeze the system. A load testing would use for testing main functions of the web-based software, e.g. opening a session, sending a request, serialization of objects, and closing a session. A timeout value of testing tool and OPC UA Client should overlap, otherwise, results of performance or load testing can give us wrong results in terms of failed requests. A question answering needs the Precision, Recall and F1-Score for evaluation. The list of meanings has been listed in Chapter 7.2 Materials and Methods. RESTful API requests are equally important for testing as performance testing. Randomly selected requests with JWT authentication and their results are shown in Listing 7-1.

7.2 Materials and Methods

OPC UA Web-Based Application is an all-in-one application that combines OPC UA Client over Web, generator tool for OPC UA Servers and a semantic question answering.

We will evaluate the web-based application in terms of latency and load testing. As for the question answering, we have parameters to assess system with precision, accuracy, and recall. Question Classification method also assessed with machine learning methods and given results as presented in Chapter 7.3. Unit testing applicability of the system is a test parameter for a web-based application. Given RESTful, calls in Chapter 6.3 were tested and results are shown in a table.

ASP.NET Core and Flask Backend Frameworks have unit testing to test important features of an application. Due to the dependency injection and routing mechanism of ASP.NET Core, we have tested through mock testing by creating a tampered controller with sample options. Such options added a different JWT authentication for test purposes. The aspect of the Semantic Question Answering, we have test cases RESTful API, Asynchronous and Synchronous Communications, and natural language processing toolkit over language inputs. We have created a session-based evaluation to test the OPC UA Client and OPC UA Web-based application together. In these tests, we have a time parameter that shows the duration of the test to run in seconds and a session parameter that indicates the number of sessions to run simultaneously. By increasing the number of threads slowly, we test how well the system reacted to parallel sessions. The main intentions are to show the degree of efficiency of the system under peak loading and burst to load and indicate how the architectural changes like adding load balancer affect the results. In a single session, the OPC UA Web Based Application Front-End will send a session that contains a JWT deserialization, a read request with node id, JSON deserialization, and an HTTP Response. Whether considering that OPC UA Client has a Session Instance separately from the web-based application, the request timeout of testing should be equal with OPC UA Default Session Timeout.

Firstly, ---- talk about first REST testing in the result page.

Secondly, we determined a set of conditions that we will test whether the system may give us a satisfying result. Within 60 seconds, we send multiple requests to different OPC UA Servers at second within sessions simultaneously 20, 25, 30, 35, 40, 45, 50, and 200 respectively. Then within 30 seconds, we send multiple requests to different OPC UA Servers at second within sessions simultaneously in the same order of session numbers. In the section titled Results 7.3, the graphs depict the number of requests and average request time. This combo chart comprises a line chart and a bar chart so that the line chart depicts the total amount of request time and the bar chart illustrates the number of requests.

//To evaluate the Semantic Question Answering, we should also assess a web server //because the nature of the question answering simply is a web server. Multitasking is //one of the important testing parameters while assessing.

As for the Semantic Question Answering System, we calculate system performance with precision, recall and F1 Score values. Moreover, we analysed the research of [Ozgur Yilmazel et. al.] [3] introduced the parameters of a restricted question answering might have "answer return rate", "querying style", "user interaction", "coverage", "size", "up-to-dateness" and "formulation assistance". Querying style can consist of keyword-based and sentence-based queries. Answer return rate measures how long an answer takes time after submitting a query by a user [3]. We can define the metrics of "coverage" with the research-domain that we have elaborated. "Size" could be measured with generated data, static data, and continuous-data produced by the eniLINK system.

Precision = True positives / (True positives + False Positives)

Recall = True positives / (True positives + False Negatives)

F1-Score = 2 x Precision x Recall / (Precision + Recall)

Accuracy of the Model = (True Positive + True Negative) / (True Positive + False Negative + False Positive + True Negative)

Moreover, the performance of question classification affect the results of the question answering system. In the practical implementation, the semantic question answering expose Li&Roth taxonomy classification and "Wh" typed question classification. The goal of evaluation for question answering to show the percentage of elimination by eliminating unnecessary answer selection.

Question Classification is the final step before implementing a query formulation. We have grouped Li & Roth Taxonomy and Wh-Question Taxonomy, which are essential categorization, used in question classification. Wh-Typed Question comprises of "wh-typed", "who", "why", "affirmation" and "unknown" types of questions. "Who" and "why" question is the type of "wh-typed question" but it is irrelevant for the semantic question answering. Li & Roth taxonomy are divided into 6 coarse-grained categories such as "Abbreviation", "Entity", "Description", "Human", "Location", and "Numeric". Dynamic Questions utilizes to classify with Li & Roth taxonomy and static questions expose to classify "Wh-Question Taxonomy" in order to identify unrelated questions.

//Talk about dataset size

//Talk about serialization results

7.3 Results

We defined RESTful HTTP calls for OPC UA Client Web-based Backend and the Semantic Question Answering Backend. RESTful calls should allow requests after authenticating with JWT tokens. As shown in Listing 7-1, every request followed the basic principle. Mock testing and unit testing implemented to the application and results are as depicted in Listing ..

Tests were selected randomly that can represent basic functionality of the web-based application.

HTTP Call	Test	Expected Output	Result
`[HttpGet("api/authenticate/)]`	Send the request without JWT.	HTTP Not Found 404. Action S	OK
`[HttpGet("api/serverconf/DataSetID/allnodes/{node_id})]`	Send the request without JWT.	OK 200. Reroute to the login page	OK
`[HttpPost("api/serverconf/DataSetID/allnodes/{node_id}) {value:"message"} ) ]`	Send multiple consecutive requests	OK 200. HTTP Response with JSON	OK
`[HttpGet("/integratedstaticmessage/{question})]`	Send the request without JWT	404 Not Found. Reroute to the login page	OK
`[HttpGet("/api/serverconf/DataSetID/subscribeNodes/monitor_id" )]`	Send the request with the wrong JWT	404 Not Found. Reroute to the login page	OK
`[HttpGet("/integrateddynamicmessage/{question}")]`	Send a PUT Request with right JWT:	405 Method Not allowed.	OK

Listing 7-1:

Results were grouped with the duration of the test that we have done. For instance, a test application creates a variety number of sessions to test a single instance of a web-based application for 30 seconds. As shown in Figure 7-2, there are no failed requests under specific circumstances. These failed requests also depend on the type of OPC UA Server. Hence, we should take into consideration to use the same types of a server so that we can provide equal conditions. In our case, we tested the system with two different OPC UA Servers. Due to the scope of the testing phase, the type of servers is irrelevant in order to reach precise results. The total amount of requests are changeable because the server threads can respond back within unspecified delayed. As illustrated in Figure 7.2, maximum request time, minimum request time and average request time are increasing when we reached 50 and 200 sessions. Nevertheless, there are no failed requests even if the number of requests increased from 40 sessions to 200 sessions.

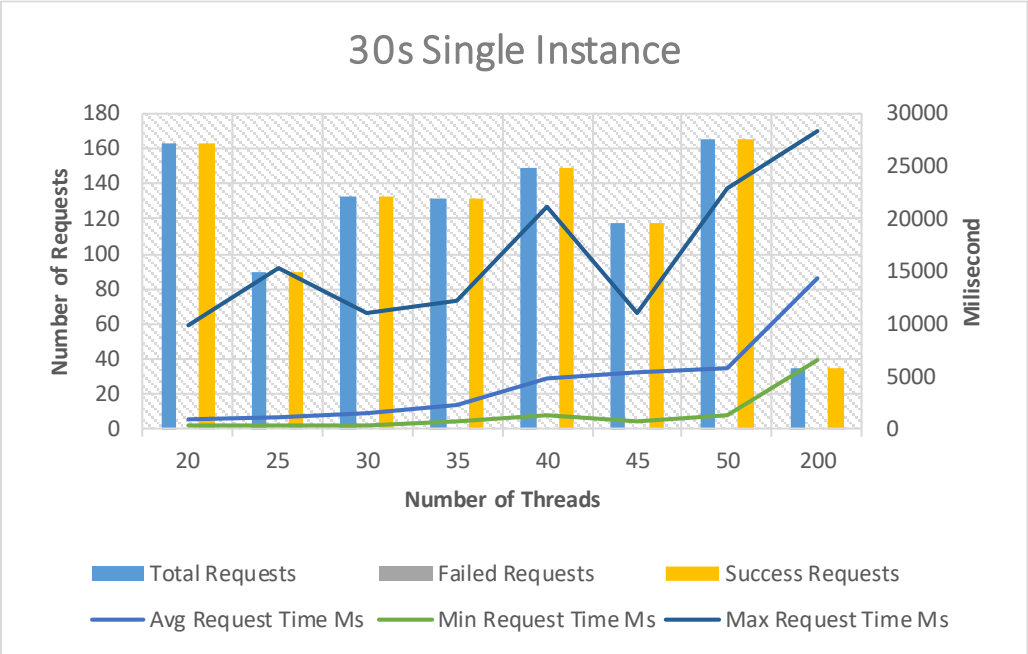

Figure 7-1: 30 second without load balancing single instance

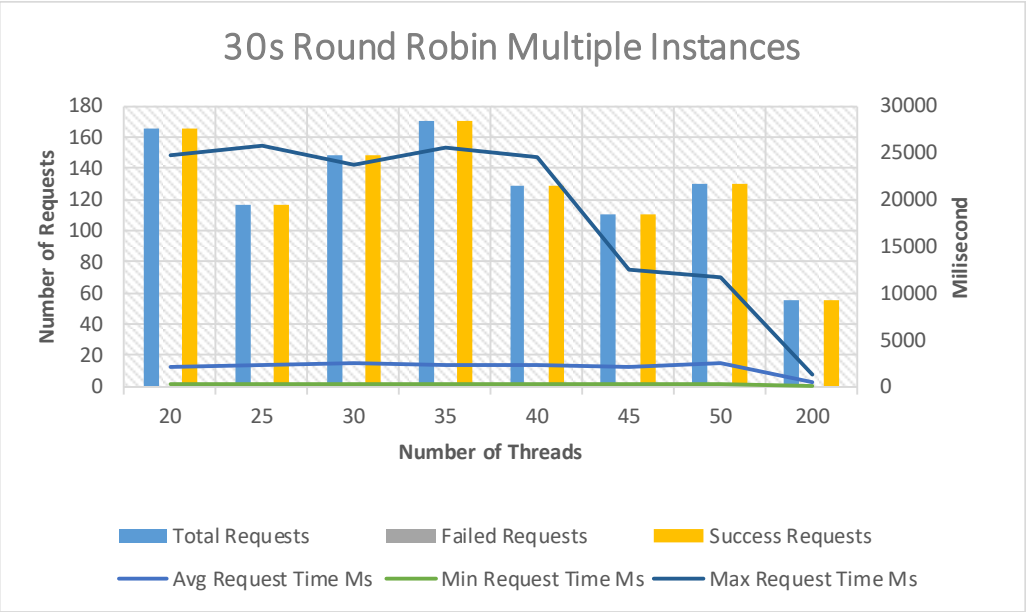

Figure 7-2: 30 second under the Round Robin Algorithm Multi-Instance

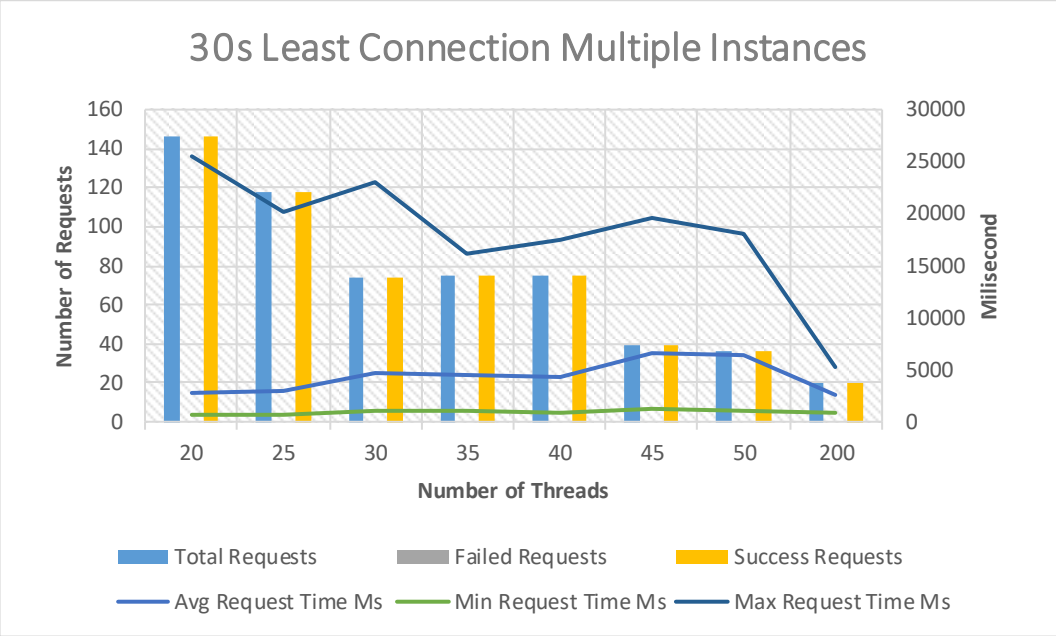

Figure 7-3: 30 second under the Least Connection Algorithm Multi-Instance

The number of total requests is similar to Figure 7-2, but maximum request time tends to decrease 40 sessions later. Average time and Max request time decreased dramatically

with the round robin load balancing algorithm from 40 to 45 and 50 to 200. The Round robin and least connection load balancing reduces the amount of average time after 50 sessions. While the OPC-UA web-based application creates maximum throughput for round robin and non-load balanced application under the number of sessions 20, 35, and 50 respectively. Unexpectedly, the least connection within the 30s as shown in Figure 7-3 could not create the same amount of throughput as compared to Figure 7-2 and Figure 7-1. Under the round-robin algorithm, the system created the least amount of minimum number requests so that it reduced the average time of requests.

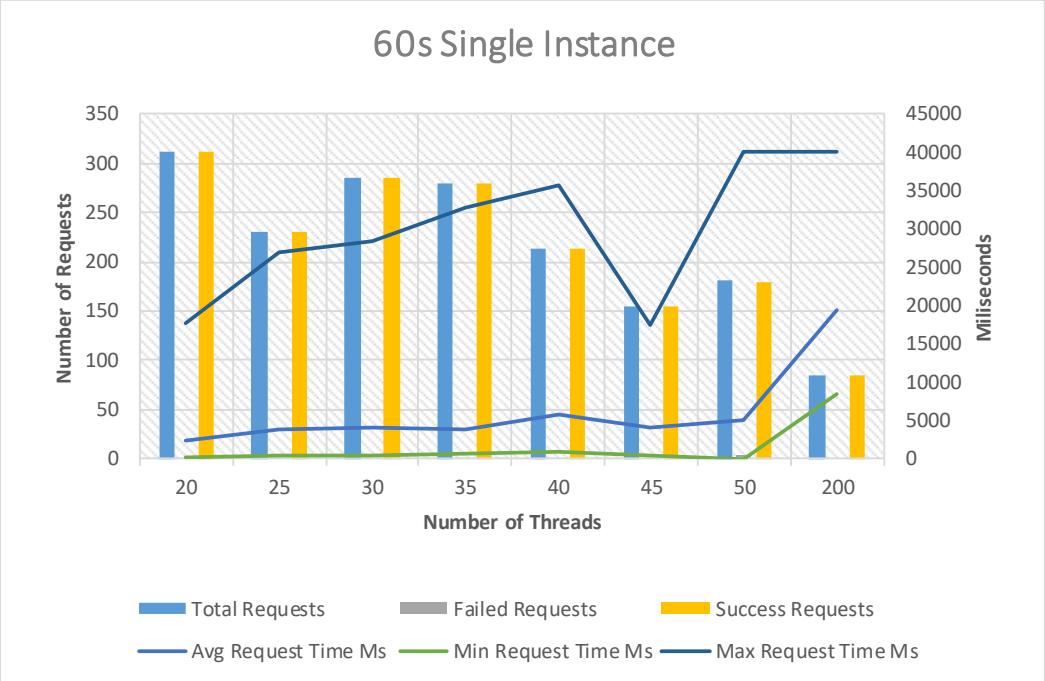

Figure 7-4: 60s Single Instance

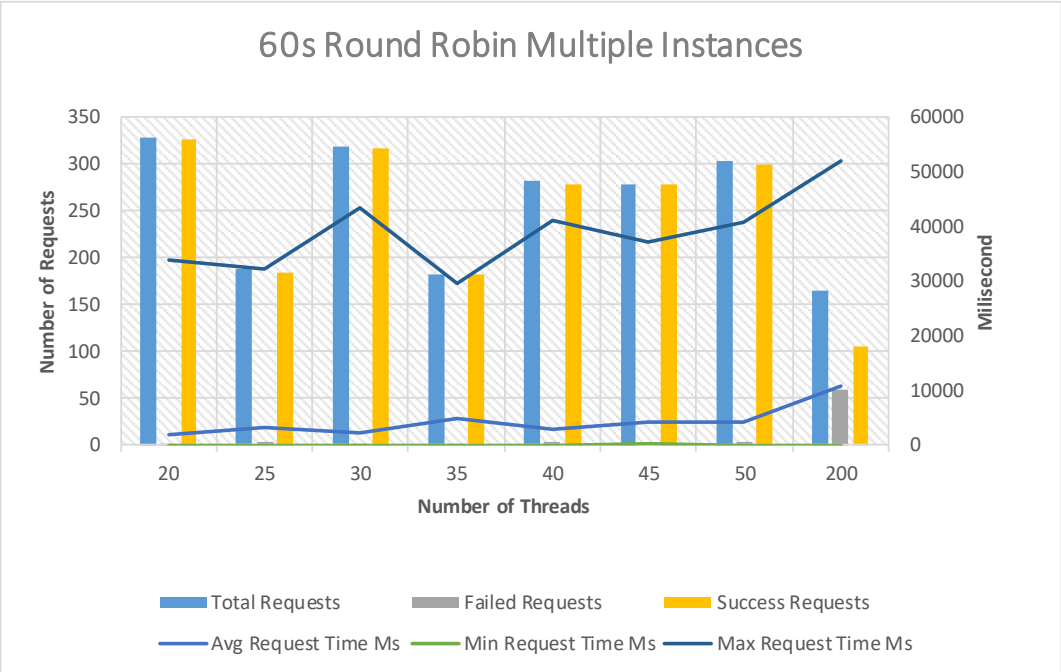

Figure 7-5: 60s Round Robin Multiple Instances

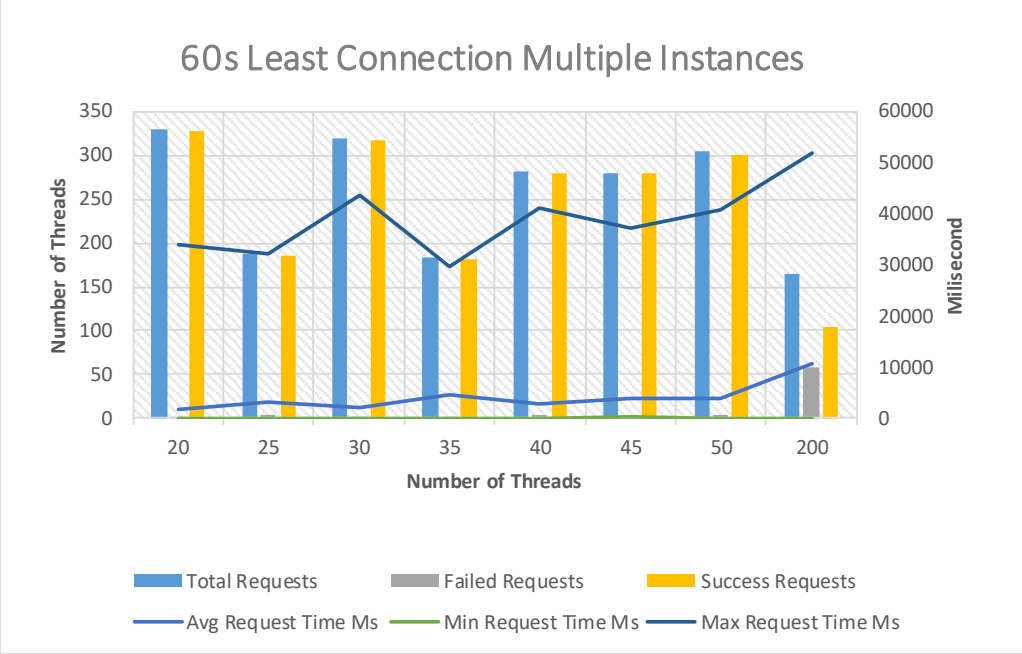

Figure 7-6: 60s Least Connection Multiple Instances

The tests during 60 second give different results aspect of request time and the number of requests.

In the semantic question application, we have a voice recognition that we use for natural queries by voice inputs. Voice-recognition is struggling to identify domain-specific words such as "fofab, gmx or glt". The reason that generic voice recognition could not identify domain-specific keyword is named-entity recognition. We should train named-entity recognition with our domain-specific data in order to use voice recognition with the semantic question answering in an efficient way. However, a voice recognition alleviates inputs size and it is more efficient while we are typing nouns, verbs, question words or adverbs. An abbreviation solver and a spell checker could be a question assistance system. Abbreviation solver is a recognition task for general abbreviations such as "A.S.A.P. – As soon as possible" or "A.A.A – The Agricultural Adjustment Act" domain-specific task as well. Compound words such as "linkedfactory" and "heatmeter" must be trained domain-specific keywords by means of named-entity recognition, otherwise regex-based or character based might not give precise results more than train-based did. The practical application provides a spell checker without a guidance system.

Evaluation Parameters	Properties
Answer Return Rate	Generated Data from OPC UA – 39.88 second – Consecutive Query of Generated Data 12.31 second Static query from RDF file of eniLINK – 19.33 second Dynamic Query – 17.48 second Open-Domain Question Answering Query – 20.55 s
Querying Style	Keyword-Based Search and Semantic Search
Coverage	eniLINK data, linkedfactory streaming data
Size	Static data relatively small size Continuous data relatively large size
Up-to-dateness	No update statement provided by SPARQL
Query Formulation Assistance	Voice Input Recognition, Spell Checker

Listing 7-2: Evaluation parameters of the Semantic Question Answering

Parameters	Precision	F1	Recall	Accuracy
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	

Listing 7-3: The Question Classification of Li&Roth and Wh-Question Taxonomy

Question Answering	True Positive	False Negative	False Positive	Precision	Recall	F1	The Accuracy of the Model
Total Questions	34	13	3	%94.44	%72.34	%81.92	%68

Listing 7-4: Total answers from Semantic Question Answering

As listed in Listing 7-1, answer return rate of the semantic question answering is close to open domain question answering. The environment of the practical part should load the vector and dependency dataset to avoid repetitive loading. Hence, consecutive queries are faster than the first query sent by experts or users. A user can search with keywords or semantic (sentence or question) words the results within the question-answering module. Data size is relatively changeable between a static search and a dynamic search. Dynamic search exploits continuous data generated by a time series and a key-value database. Sending a query to a SPARQL endpoint takes time due to the processing time of the SPARQL engine. Nevertheless, the dynamic search can respond within 30 minutes, which is a similar time value as compared to the static search. The semantic

question answering does not allow updating to a triple because changes of triples in Turtle Source can create a vulnerability. The data source covers statically linked data generated by the Dynamic Server or eniLINK and dynamically linked data by KVIN Service. Last but not least, query assistance has been provided by the semantic question system to improve the search quality and ease of use.

As shown in Listing 7-2, a question classification can find a classification subset of questions with a machine learning method. As previously described, this thesis considered Support Vector Machine and Logistic Regression. The semantic question answering does not use a multi-layer perceptron, which is one of the deep learning methods, thereby taking a long time to train data. Logistic Cross-Validated Regression gives the better result under a 1559 lined labeled dataset. The Linear SVC has been used to train Li&Roth taxonomy, but unexpectedly the accuracy, precision and F1 Score values gave a lower performance. On the other hand, the Linear SVC created a result over %90 for the dataset of Wh-Question Taxonomy.

Listing 7-3 shows us the answer rate of the semantic question answering. In Chapter 7.2, we have explained the parameters of Precision, Recall, F1 Score, and Accuracy of the Model. All of the parameters are above than %60 percent. We can

Listing 7-6 takes into consideration to answer to Factoid Question, Keyword Question, Indicative Question, Boolean Question.

//Start to write the conclusion part

//Another result is about imbalanced data sets. Compare the sets of accuracy

8 Conclusion

8.1 Summary

OPC UA Web-based application is an essential task that meets the general requirements of the OPC UA Protocol. Due to the enormous size of data produced by OPC UA protocol or production machines, human operators need a guidance system that alleviates the complexities by using natural languages queries.

The problem of design and implementation can be achieved with our proposal.

//List of your key findings

//The conclusion is an opportunity to succinctly answer the "So What?" question by placing the study within the context of how your research advances past research about the topic.

//We showed the architecture would be useful for smart factories. Without SPARQL, the efficiency of human operators increases. The Difficulty of stream data queries is that a query delays sending a query until it gets an answer. This can be acceptable steps before query formulation causes delays.

// OPC UA Servers can be used with desktop applications instead of OPC UA Web-based software, however. However, major drawbacks are installation, admin-panel security, ineligible to use simultaneously.

// We proved that the information model and the address space of OPC UA Protocol are convenient to deploy with a semantic question answering. At the beginning of research, our initial question was like how can we send a natural query to an OPC UA Server.

//The limitation is data size and quality. We need more subject-predicate-object triples. More importantly, the data source of any smart factories should categorize and customize the predicates of triples according to the requirements of any smart factories. If so, we can implement algorithms that are more precisely such as deep learning or reinforcement learning methods to improve the quality of answers. Moreover, the system can be designed to show answers to reasoning questions. For instance, when a production machine created an error, a question answering system can answer questions regarding errors.

//Synchronous calls are not suitable for OPC UA Web-based communication. Creating threads for every session created by an external user might lead to a single point of failure because of the synchronization of threads. We offered non-blocking I/O queues to queue every request by implementing quasi-asynchronous calls. Even if a web-based application created with asynchronous calls, it balances the load of requests to stave off a single point of failure or failure state. On the other hand, a discovery server can complete the architecture in terms of Service Oriented Architecture. Our approach handles the communication between client-server architecture hard-coded endpoints. A discovery service would be more convenient in remote communication connected with multiple OPC UA Servers. Finding endpoints and sharing certificates become signification in the existence of inter-smart factories.

//The hardest part was --- integration of streamed data key-value mapper, find a way querying into OPC UA Servers, reducing the average request time with a properly selected algorithm,

//The semantic question answering is not useful for mission-critical systems. The solution of this problem a template based query generation can be enforced.

//Instead of creating a new application for instant semantic linked data, OPC UA Servers can shape the data, however, it is cumbersome and time-wasting objective. At a limited phase, one can utilize for the linked data concept, but the generalization of this application is not possible because of the structures of every OPC UA Servers.

1) Conclusions: concise statements about your main findings, related to your aims/objectives/hypothesis

1) What should (and should not) be in the conclusion?
2) How long should it be?
3) What am I trying to say in my conclusion?

What should not be in the conclusion?

1) Discussion: This should be in the Discussion section. If your thesis combines the two, use sub-headings to distinguish between them

2) Any points that have not been mentioned in the Discussion section; your conclusions should be based only on points already raised.

3) References: It is quite unusual to include references in this section, as it is mainly a review of what has already been said.

4) Unnecessary information: your conclusion should be concise.

How long should my conclusion be?

The length of your conclusion will depend on a number of variables,

Check with your supervisor and with highly regarded past theses.

What am I trying to say in my conclusion?

What are you trying to say?

What I did learn?

What am I proudest of?

What was the hardest part?

How did I solve the difficulty?

Alternatively, in other words:

1) To what extent you achieved your aims/objectives OR not; if not, why not?

2) How important and significant your results are, as well as any limitations of your research (e.g. small sample size, other variables)

3) Where the research should go from here: what are some interesting further areas to be explored based on what you discovered or proven?

//All research questions in the section named Scope and Methods. Answer all of them

8.2 Main Contributions

This thesis contributes to both the OPC UA Web-based Software and Question Answering research area in the following ways:

- We introduce a new research topic about a combination of OPC UA web-based software with Authentication Control and evaluate the suitability of a generated source data of web-based software to a Semantic Question Answering. Moreover, this study improves previous studies that have mentioned in Chapter 2.1.1.
- We conveyed research relevant to Natural Language Understanding Technology and a communication protocol to serve as an industrial automation assistance system aspect of controlling and monitoring of production systems. Hence, we put together different research areas that had never been combined before.
- We are simplifying the problem of OPC UA RESTful service integration into session based OPC UA protocol, generalizing the problem of restricted domain question answering aspect of smart factories and serializing the OPC UA Information Model and streamed data presented by smart factories with the comparison of algorithms that the serialization implemented. Indirectly, we provide a semantic model to solve constraint the meaning of exposed data in the information model.
- We are formalizing the analysis in various ways. Firstly, a theoretical background about OPC UA Protocol and Services were explicitly stated in order to examine functional and performance results of the web application. Secondly, we have given a comprehensive analysis over semantic serialization of OPC UA Information Model and streamed data of a smart factory so as to interpret viability and suitability as a data source for a restricted question answering system. Thirdly, we presented the semantic question answering with implementation and design details that can be beneficial for other researchers.
- We study generating semantically linked data from an OPC UA Server, an embedded controller such as S7-Controller, real-time and semantic data based-on Fraunhofer IWU (Enilink) and assess their performance with Precision, Recall,

F1 Score, and Accuracy parameters against a Semantic Question Answering System. In addition, we propose a way to use streamed data key-value mapping and show the differences in the use of other streamed linked data processing.

- We demonstrate the results through experiments and empirical results and compare them with former solutions
- We employ state-of-art research about Semantic Question Answering with natural language processing tools and assess generated data by OPC UA Servers against the Semantic Question Answering. Furthermore, we propose a novel architecture to implement OPC UA Web-based Software and assess the architecture. Along with the performance improvements, our practical work shows guidance for future research in terms of abstraction, reusability, discoverability, and composability.

8.3 Assessment of Research Questions and Requirements

In this chapter, we will answer the questions that we have defined in Chapter 1.3. OPC UA Web-Based Software has targeted multiple roles in accordance with data acquisition from OPC UA Servers, creation of the linked data, and a question answering system. The main goal was to create a combined a Human Machine Interaction tool that could be used in a smart factory. Moreover, we point out general problems so that other smart factories can develop their own solutions. First, a technical user or a web user does not need to know detailed system architecture. However, the semantic question answering needs specific keywords that technical personals only knows. This issue is not a bad evidence for a question answering system because a restricted-question answering system can face such issues that we have examined in Chapter 2.2. A drawback of the web-based application can suffer performance problem if the system is implemented large-scale network because of architectural requirements. For instance, the division between smart factories demand different volume of performance while using the web-based application. Our finding is to identify most loaded parts in the web-based application and take countermeasure with a load balancer. According the experimental development phase shows us that a monolithic and a loose-coupled architecture can meet the requirements at a basic level. In order to implement a system that is more complex with gateway integrated

into microservices can meet requirement of remote smart factories. Another drawback is a latency problem that caused by consecutive queries. We have experimented load balancing between web user and front-end application and among back end application with non-blocking input/output queue. As for the question of "Which technologies should I use?" , a software developer has variety of opportunities that described in Chapter 6. While backend frameworks need performance of serialization of JSON and XML data and low latency, frontend framework expect to handle dynamic user interface and data binding and asynchronous features. Moreover, pros and cons of technologies have been listed in Chapter 6 in a detailed way. So we can answer the question "Why these technologies have been used?". Serialization of the OPC UA Address Space has been examined in Chapter 4.1.4 and 4.1.5. As previously mentioned, we should entirely take into consideration continuous data and static data. KVIN Service handle continuous data by mapping key-value pair with LevelDB. After testing with the question answering, natural queries can give result with SPARQL query. On the other hand, we reached the goal of how we can send a query to OPC UA Servers by means of using linked data. The Main reasons that we constructed the semantic question answering are allowing implementing an information extraction system, reducing errors because of SPARQL queries, and exemplifying a restricted-question answering aspect of smart factories. (Evaluation of Question Answering)

Chapter 7.1 detailed parameters of the evaluation

8.4 Future Works

2) Future Research: Where to go from here (can include where NOT to go, if your research demonstrated that a particular approach or avenue was not useful)

The devices that connected to OPC UA are growing day by day and requirements are expanded according to new requirements of smart factories. Experts need more tools that can take natural input by giving a scientific output. In order to provide this, linked data sources should be more capable of representing internal devices, actuators, and sensors. Such HMI devices connected to a remote domain of a smart factory need a more complex discovery service like Global Discovery Service. A Global Discovery Service can handle certificate management between local and remote domain through a certificate distribution. So each device can authenticate through a global discovery service and

OPC UA Clients can easily find endpoints of OPC UA Servers. OPC UA Servers can change data inside and enlarge with a hierarchical structure of a smart factory. To serialize such changes requires a continuous serialization process; however, a continuous SPARQL language can detect the triples on updated roots of OPC UA Servers. Many different serializations of OPC UA static data has been left for further development due to architectural difficulties. KVIN Service could be turned into a continuous SPARQL endpoint instead of key-value storage.

//Named Entity Recognition can customize for company needs.

//Aggregated Server for general requirements for Smart Factory and for monitoring nodes.

//Abbreviation Checker can be handled

//Linked Data Efficiency will be improved. Especially, we can add more properties and corresponding explanations through the linkedfactory web page.

//A microservice architecture can be planned

//Reason induction system can be added if more data would be provided

//A machine learning system based on deep learning can be implemented if we get more data that generated by Fraunhofer IWU.

//Spell Checker has been implemented

//In a particular range of time, a user send a real time query against KVIN Service.

//Non Blocking IO Queue (e.g. Rabbitmq can be added to balance load)

//Sparql endpoint of KVIN should comply with the param http://domain.com:10080/sparql/endpoint&{SPARQL Query}

//Multiple federated query support with “SERVICE” statement can be extended with the new version. SPARQL GRAPH statement can be extended in accordance with Update

Bibliography

[1] F. IWU, 'eniLink', 2015. [Online]. Available: http://platform.enilink.net/. [Accessed: 23-Nov-2018].

[2] K. Thoben, S. Wiesner, and T. Wuest, '" Industrie 4 . 0 " and Smart Manufacturing – A Review of Research Issues and Application Examples', vol. 11, no. 1, 2017.

[3] A. R. Diekerma and E. D. Liddy, 'Evaluation of restricted domain Question-Answering systems', *Cent. Nat. Lang. Process.*, pp. 12–16, 2004.

[4] Microsoft, 'Microsoft and the OPC Foundation Demonstrate Industry-Standard Interoperability at ISA Expo/98', 1998. [Online]. Available: https://news.microsoft.com/1998/10/19/microsoft-and-the-opc-foundation-demonstrate-industry-standard-interoperability-at-isa-expo98/. [Accessed: 12-Mar-2018].

[5] Unified Architecture, 'OPC Technologies', *OPC UA Foundation*. .

[6] S. Cavalieri, M. G. Salafia, and M. S. Scroppo, 'Integrating OPC UA with web technologies to enhance interoperability', *Comput. Stand. Interfaces*, no. August 2017, 2018.

[7] S. Cavalieri, D. Di Stefano, M. G. Salafia, and M. S. Scroppo, 'A web-based platform for OPC UA integration in IIoT environment', *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, pp. 1–6, 2017.

[8] T. Paronen, 'A web-based monitoring system for the Industrial Internet', 2015.

[9] X. Su, H. Zhang, J. Riekki, A. Keränen, J. K. Nurminen, and L. Du, 'Connecting IoT sensors to knowledge-based systems by transforming SenML to RDF', *Procedia Comput. Sci.*, vol. 32, pp. 215–222, 2014.

[10] X. Wang, X. Zhang, and M. Li, 'A survey on semantic sensor web: Sensor ontology, mapping and query', *Int. J. u- e- Serv. Sci. Technol.*, vol. 8, no. 10, pp. 325–342, 2015.

[11] K. R. Llanes, M. A. Casanova, and N. M. Lemus, 'From Sensor Data Streams to Linked Streaming Data : a survey of main approaches', vol. 7, no. 2, pp. 130–140, 2016.

[12] H. Hasemann and A. Kröller, 'The Wiselib TupleStore: A Modular RDF Database for the Internet of Things', *J. Phys. Soc. Japan*, Apr. 2018.

[13] H. Müller, L. Cabral, A. Morshed, and Y. Shu, 'From RESTful to SPARQL: A case study on generating semantic sensor data', *CEUR Workshop Proc.*, vol. 1063, pp. 51–66, 2013.

[14] E. Laukkanen, 'Java source code generation from OPC UA information models', p. 72, 2013.

[15] B. Katti, C. Plociennik, and M. Schweitzer, 'SemOPC-UA: Introducing Semantics to OPC-UA Application Specific Methods', *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1230–1236, 2018.

[16] Pure Python OPC-UA Client and Server, 'Free OPC-UA Library'. [Online]. Available: https://github.com/FreeOpcUa/python-opcua. [Accessed: 22-Nov-2018].

[17] M. Schleipen, O. Sauer, and J. Wang, 'Semantic integration by means of a graphical OPC Unified Architecture (OPC-UA) information model designer for Manufacturing Execution System', Karlsruhe.

[18] V. Zhong, C. Xiong, and R. Socher, 'Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning', pp. 1–12, 2017.

[19] X. Xu, C. Liu, and D. Song, 'SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning', pp. 1–13, 2017.

[20] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, 'The Stanford CoreNLP Natural Language Processing Toolkit', *Proc. 52nd Annu. Meet. Assoc. Comput. Linguist. Syst. Demonstr.*, pp. 55–60, 2014.

[21] F. F. Luz and M. Finger, 'Semantic Parsing Natural Language into SPARQL : Improving Target Language Representation with Neural Attention', vol. 1803.04329.

[22] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, 'Template-based question answering over RDF data', *Proc. 21st Int. Conf. World Wide Web - WWW '12*, p. 639, 2012.

[23] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, 'SPARQL as a Foreign Language', no. August, 2014.

[24] S. Palaniappan, U. K. Sridevi, and J. Subburaj, 'Ontology based Question Answering system using JSON-LD for Closed Domain', vol. 119, no. 12, pp. 1969–1980, 2018.

[25] C. Giannone, V. Bellomaria, R. Basili, and I. Department of Enterprise Engineering, University of Rome Tor Vergata, Roma, 'A HMM-based Approach to Question Answering against Linked Data', *Comput. Sci.*

[26] S. C. Tirpude and A. S. Alvi, 'Closed Domain Keyword based Question Answering System for Legal Documents of IPC Sections & Indian Laws', no. 2, pp. 5299–5311, 2015.

[27] D. Mollá and J. L. Vicedo, 'Question answering in restricted domains: An overview', *Comput. Linguist.*, vol. 33, no. 1, pp. 41–61, 2007.

[28] S. Ferré, 'SQUALL: A controlled natural language for querying and updating RDF graphs', *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7427 LNAI, pp. 11–25, 2012.

[29] V. Garcia and E. Motta, 'AquaLog: An ontology-driven question answering system to interface the semantic web', *Proc. 2006 Conf. Hum. Lang. Technol. Conf. NAACL*, vol. Companion, no. June, pp. 269–272, 2006.

[30] E. Kaufmann, A. Bernstein, and L. Fischer, 'NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies', *4th Eur. Semant. Web Conf. ESWC 2007*, pp. 1–2, 2007.

[31] D. Damljanovic, M. Agatonovic, and H. Cunningham, 'Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction', *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6088 LNCS, no. PART 1, pp. 106–120, 2010.

[32] M.-A. Nolin *et al.*, 'Bio2RDF Network Of Linked Data', *Web*, no. January, pp. 1–8, 2008.

[33] V. Lopez and E. Motta, 'PowerAqua: Fishing the Semantic Web', vol. 7295, no. May 2014, 2012.

[34] A. Marginean, 'GFMed:Question answering over biomedical linked data with Grammatical Framework', *Semant. Web*, vol. 8, no. 4, pp. 565–580, 2017.

[35] C. Unger and P. Cimiano, 'Pythia: Compositional meaning construction for ontology-based question answering on the semantic web', *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6716 LNCS, pp. 153–160, 2011.

[36] R. Margaret and D. Daniel, 'Definition of Smart Factory'. [Online]. Available:

https://searcherp.techtarget.com/definition/smart-factory. [Accessed: 05-Dec-2018].

[37] C. Team, 'What is the smart factory and its impact on manufacturing?', *13 June 2018*. [Online]. Available: https://ottomotors.com/blog/what-is-the-smart-factory-manufacturing. [Accessed: 05-Dec-2018].

[38] T. Stock and G. Seliger, 'Opportunities of Sustainable Manufacturing in Industry 4.0', *Procedia CIRP*, vol. 40, no. Icc, pp. 536–541, 2016.

[39] 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', *Comput. Ind.*, vol. 83, pp. 121–139, 2016.

[40] D. Gorecky, M. Schmitt, M. Loskyll, and D. Zühlke, 'Human-machine-interaction in the industry 4.0 era', *Proc. - 2014 12th IEEE Int. Conf. Ind. Informatics, INDIN 2014*, pp. 289–294, 2014.

[41] C. Gonzalez, 'What are Human Machine Interfaces and Why Are They Becoming More Important?' [Online]. Available: https://www.machinedesign.com/iot/what-are-human-machine-interfaces-and-why-are-they-becoming-more-important. [Accessed: 04-Dec-2018].

[42] J. A. Holgado-terriza, 'Mobile Human Machine Interface based in OPC UA for the control of industrial processes . Mobile Human Machine Interface based in OPC UA for the', no. August, 2016.

[43] O. Niggemann, G. Biswas, J. S. Kinnebrew, H. Khorasgani, S. Volgmann, and A. Bunte, 'Data-driven monitoring of cyber-physical systems leveraging on big data and the internet-of-things for diagnosis and contro', *CEUR Workshop Proc.*, vol. 1507, no. August, pp. 185–192, 2015.

[44] VisualWorks, 'Cincom Smalltalk ObjectStudio OLE User's Guide', Cincinnati, Ohio, P40–3805–03 ©, 2010.

[45] M. Bell, 'OLE for Process Control: Overview', in *OPC Foundation, Oct*, 1998, pp. 1–12.

[46] Unified Automation GmbH, 'Introduction to Classic OPC'. [Online]. Available: http://documentation.unified-automation.com/uasdkhp/1.0.0/html/_l2_classic_opc.html. [Accessed: 05-Dec-2018].

[47] O. F. Burke, Thomas, President and B. S. I. Eric J, Byres, Chifef Technology Officer, 'Securing Your OPC Classic Control System'.

[48] Arnon Rotem-Gal-Oz, *SOA Patterns*, First Edit. New York: Manning, 2012.

[49] B. Farnham and R. Barillère, 'Migration From OPC-DA to OPC-UA', *Proc. ICALEPCS2011*, 2011.

[50] O. P. C. Unified, A. Specification, A. Space, and M. Release, 'OPC Unified Architecture Part 3: Address Space Model', *Specification*, vol. Part 3, no. Release 1.04, 2017.

[51] Unified Automation GmbH, 'Address Space Concepts'. [Online]. Available: http://documentation.unified-automation.com/uasdkhp/1.0.0/html/_l2_ua_address_space_concepts.html. [Accessed: 07-Dec-2018].

[52] Mariusz Postol PhD. Eng. (Project Manager), 'OPC UA Information Model Deployment', Wolczanska, Poland, 2016.

[53] O. P. C. Unified, A. Specification, and S. Release, 'OPC Unified Architecture Specification Part 4: Services', *Specification*, vol. Part 4, no. Release 1.04, 2017.

[54] O. P. C. Unified, A. Specification, and I. M. Release, 'OPC Unified Architecture Specification Part 5: Information Model', *Specification*, vol. Part 5, no. Release 1.04, 2017.

[55] OPC Foundation, 'OPC Unified Architecture Specification Part 14: PubSub Release', 2018.

[56] P. D. Leslie F. Sikos, 'Mastering Structured Data on the Semantic Web'.

[57] B. DuCharme and Beijing, *Learning SPARQL Querying and Updating with SPARQL 1.1 Bob*, Second Edi. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'REILLY, 2013.

[58] TU Dresden, 'Plt-TUD'. [Online]. Available: https://github.com/plt-tud/opc_ua_xml_export_client. [Accessed: 22-Nov-2018].

[59] C. D. Manning, 'Foundations of Statistical Natural Language Processing - Christopher D. Manning', pp. 1–704, 2005.

[60] D. Jurafsky and J. H. Martin, 'Speech and Language Processing', *Speech Lang. Process. An Introd. to Nat. Lang. Process. Comput. Linguist. Speech Recognit.*, vol. 21,

pp. 0–934, 2009.

[61] A. Taylor, M. Marcus, and B. Santorini, 'The Penn Treebank: An Overview'.

[62] E. Loper and S. Bird, 'NLTK: The Natural Language Toolkit', 2002.

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

[64] neo4j, 'The Jaccard Similarity Algorithm'. [Online]. Available: https://neo4j.com/docs/graph-algorithms/current/algorithms/similarity-jaccard/. [Accessed: 16-Jan-2019].

[65] P. Christen, 'A Comparison of Personal Name Matching: Techniques and Practical Issues', *Sixth IEEE Int. Conf. Data Min. - Work.*, no. September, pp. 290–294, 2006.

[66] Wordnet Similarity for Java, 'WS4J Demo'. [Online]. Available: http://ws4jdemo.appspot.com/?mode=s&s1=Is+the+system+health+is+good%3F&s2=What+is+the+status+of+system%3F. [Accessed: 17-Jan-2019].

[67] T. Wei and H. Chang, 'Measuring Word Semantic Relatedness Using WordNet-Based Approach', *J. Comput.*, vol. 10, no. 4, pp. 252–259, 2015.

[68] OPC Foundation, 'Build OPC UA .NET applications using .NET StandardLibrary', 2016. [Online]. Available: http://opcfoundation.github.io/UA-.NETStandard/. [Accessed: 10-Dec-2018].

[69] B. T. C. S. A. Fraunhofer IOSB, ACLPT, RWTH, Technische Universität Dresden, fortiss, kalycito, basysKom, HMS, Hilscher Competence in Communication, 'open62541 implementation', 2018. [Online]. Available: https://open62541.org/. [Accessed: 10-Dec-2018].

[70] A. Lock, *ASP.NET Core In Action*. New York: Manning Publications Co., 2018.

[71] TechEmpower, 'TechEmpower Framework Benchmarks'. [Online]. Available: https://github.com/TechEmpower/FrameworkBenchmarks. [Accessed: 23-Jan-2019].

[72] J. H. Skeie, *Ember.js in Action*. New York: Manning Publications Co., 2014.

[73] M. Tielens Thomas, *React in Action*. Manning Publications Co., 2018.

[74] S. Hochhaus and M. Shoebel, *Meteor In Action*. New York: Manning Publications Co., 2016.

[75] O. P. C. Unified, A. Specification, and S. M. Release, 'OPC Unified Architecture Specification Part 2:Security Model', vol. Part 2, no. Release 1.04, 2018.

[76] Universita degli Studi di Catania, 'OPC UA Web Platform', 2017. [Online]. Available: https://github.com/OPCUAUniCT/OPCUAWebPlatformUniCT. [Accessed: 14-Jan-2019].

[77] W3C, 'Xpath Tutorial'. [Online]. Available: https://www.w3schools.com/xml/xpath_intro.asp. [Accessed: 14-Jan-2019].

Appendix A

A.1 Question, Precision, Recall

Question	ID	Precision	Recall
What do linkedfactory, heatmeter, and e3fabrik incorporate exactly?	1	0	0
Provide me a combined result for IWU and e3sim	2	1.0	1.0
I want to know which one carries fofab?	3	1.0	1.0
There is a member named fofab. Please give me all of its members	4	1.0	1.0
I am a customer for this company. Could you tell me please what the value of sensor1 of machine1 is	5	0	0
Could you tell me please what is the current value of sensor2 in machine2?	6	1.0	1.0
What POWERMETER holds?	7	1.0	1.0
What does FOFAB incorporate?	8	1.0	1.0
What does machine5 HOLD?	9	1.0	1.0
What does gmx comprise?	10	1.0	1.0

What comprises karobau?	11	0	0
System health for sensor2 in machine6	12	1.0	1.0
Tell me the health of system for sensor2 in machine1	13	0.0	0.0
Could you browse generated data?	14	1.0	1.0
Give me all of the members of gmxspanen4	15	0.0	0.0
What holds coolingwater?	16	1.0	1.0
What is the hierarchical structure of fofab?	17	1.0	1.0
What contains IWU?	18	A	B
Could you give me the members in which contained by versuchsfeld?	19	1.0	1.0
Could you give me the members in which linkedfactory has?	20	A	B
What is the value of sensor1 in machine6?	21	1.0	1.0
What is minimum that we can calculate for sensor1 of machine1?	22	1.0	1.0
What is the value of maximum can be calculated by the sensor1 of machine1?	23	1.0	1.0
Could you tell me what the average for sensor3 in machine1 is?	24	1.0	1.0
I need to learn an average value for sensor5 in machine2	25	0.0	0.0

What is the average of sensor3 in machine3?	26	A	B
Could you get me the references of nodes?	27	A	B
Could you browse generated data?	28	A	B
Is the E3-Sim member of linkedfactory?	29	0.0	0.0
Could you take me all members of generated data?	30	A	B
Give me all registered node id	31	A	B
I need to learn parent node id in generated data	32	A	B
Could you give me parent node id in the file of generated data?	33	A	B
Give me all data blocks	34	A	B
Data blocks in generated OPC file	35	A	B
Give me the name of stations in generated data	36	A	B
All stations which are in generated data or new data	37	A	B
Please combined result of datablock, station	38	A	B
Who is Fofab?	39	0.0	0.0
Why can all nodes can be browsed?	40	0.0	0.0

Table 0-1: Precision and Recall of Answers

A.2 Coffeescript Sample

```
SDNEntity = require('./SDNEntity.js')
_ = require('underscore')
uuid = require('uuid')
config = require('../config.js')
request = require('request')

class SDNController extends SDNEntity
  controllers = []

  constructor: (@uReg) ->
    @controllerID = 2000
```

Listing 0-1: A sample from Coffeescript

A.3 JavaScript Counterpart of the CoffeeScript Sample

```
 (function() {
  var SDNController, SDNEntity, config, request, uuid, _,
    __bind = function(fn, me){ return function(){ return
fn.apply(me, arguments); }; },
    __hasProp = {}.hasOwnProperty,
    __extends = function(child, parent) { for (var key in
parent) { if (__hasProp.call(parent, key)) child[key] =
parent[key]; } function ctor() { this.constructor = child; }
ctor.prototype = parent.prototype; child.prototype = new
ctor(); child.__super__ = parent.prototype; return child; };

  SDNEntity = require('./SDNEntity.js');

  _ = require('underscore');

  uuid = require('uuid');

  config = require('../config.js');

  request = require('request');

  SDNController = (function(_super) {
    var controllers;

    __extends(SDNController, _super);

    controllers = [];
```

Listing 0-2: Counterpart of sample CoffeeScript in Figure 1.1

A.4 KVIN Service Sample Query

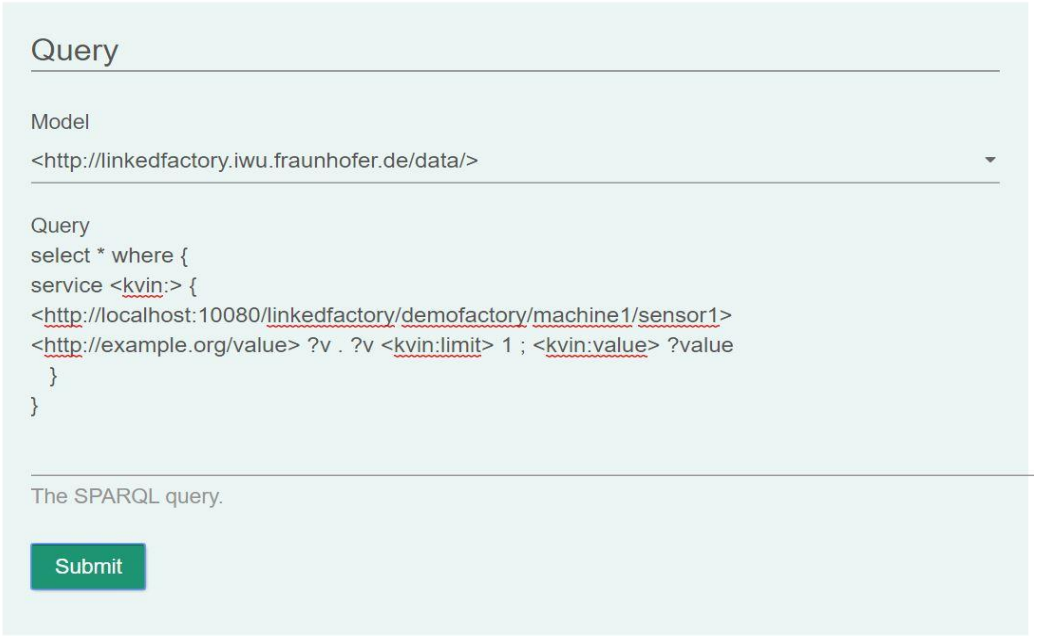

Figure 0-1: Enilink Sample SPARQL Query

A.5 KVIN Service Result of a Key-Value Pair

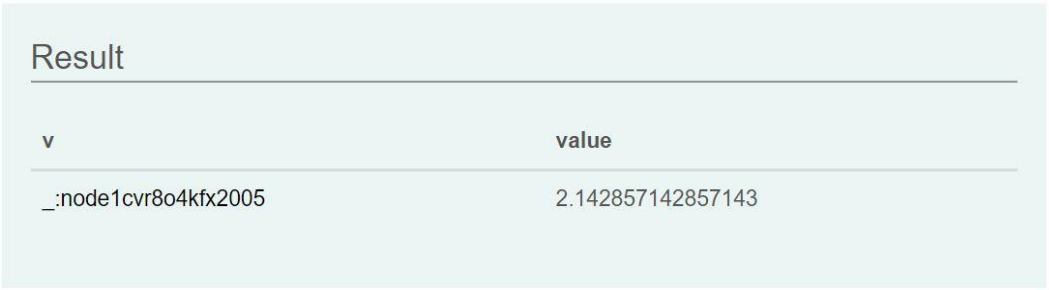

Figure 0-2: A result from a continuous data

Glossary

Softmax Layer: It is a regression-based result to assign a multi-classification machine learning problem.

Machine Learning: It is the science of getting computers to act without being explicitly programmed.

Reinforcement Learning: It is ta type of Machine Learning Algorithms which allows software agents and machines to automatically determine the ideal behavior within a specific context, to maximize its performance.

Long Short Term Memory: LSTM is a unit of recurrent neural network which composed of a cell, an input gate, an output gate and a forget gate.

Bi-directional Long Short Term Memory: A bidirectional LSTM layer learns bidirectional long-term dependencies between time steps of time series or sequence data.

Word Vector Representation: It is a word vector in a row of real valued numbers

Recurrent Neural Network: It is a subclass of artificial neural network where connections between nodes from a directed graph or directed acyclic graph along a sequence.

Stanford CoreNLP Tokenization: It provides a tool that tokenizes a text snippet or blob of text

Stanford CoreNLP Part of Speech Tagger (POS Tagger): It provides a tool of which labels tokens with their part of speech tag

Neural Machine Translation: It is an end-to-end learning approach for automated translation, with the potential to overcome many of the weaknesses of conventional phrase-based translation systems.

Epoch: This term explains that is single pass through whole training dataset.

Index

No index entries found.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Chemnitz, den 26. February 2019

_____

[Comments] Orcun Oruc

TODO: Es wird empfohlen die offizielle Selbständigkeitserklärung des ZPAs zu verwenden: http://www.tu-chemnitz.de/verwaltung/studentenamt/zpa/formulare/Allgemein/allgemein/selbststaendigkeitserklaerung.pdf