Sampling IoT Knowledge Graph to Enhance Computational Efficiency

BSc Thesis *(Afstudeerscriptie)*

written by

Raihan Karim Ishmam
(born Septemer 12th, 2003 in Bangladesh)

under the supervision of **Dr. Victor de Boer** and **Mr. Roderick van der Weerdt**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

Bachelor in Artificial Intelligence
Intelligent Systems Track

at the *Vrije Universiteit Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
July 21st, 2023	Dr. Ronald Siebes (second reader)

Abstract

In recent years, IoT knowledge graphs have emerged as a focal point in the scientific community, playing a pivotal role in addressing the challenges of data integration. Not only has it attracted widespread attention for its convenient semantic interoperability, but has also unlocked immense potential for data analytics to serve in the IoT domain. One issue is that, because of the heterogeneous and longitudinal format of data collected by devices, IOT KGs may be very large. This can lead to challenges in loading and querying the knowledge graphs, resulting in inefficiency in the process. Sampling techniques offer a practical way to tackle the size issue of IoT knowledge graphs, thereby enabling efficient analysis. This study investigates the sampling of IoT KGs, reducing them to a manageable size while maintaining good computational properties to answer relevant user queries. We investigate this in the form of a case study, where we work with data from multiple heterogeneous devices in a Dutch office building, collected in the context of the Interconnect project. Our algorithm uses semantic filters based on relational properties to sample desired parts of the KG. The relational properties are hosted by the SAREF ontology, making the algorithm extendable to other SAREF-based KGs. We evaluated the method by querying the sampled knowledge graph for the end-user-provided usage scenarios. The results show that the sampled knowledge graph retains completeness with regard to our user-end objectives, despite being reduced to a smaller size.

Keywords— IoT, Knowledge Graphs, Interconnect Project, SAREF

Contents

Chapter 1

Introduction

Linked data have become an integral part of the data science field, enabling the integration and interlinking of diverse data sources to a semantic web of shared data [9]. As part of this paradigm, knowledge graphs have emerged as a powerful representation and organization mechanism for connected data [15]. Knowledge graphs capture the relationships and semantic connections between entities, forming a network of interlinked entities. The capabilities of knowledge graphs extend to facilitate advanced data exploration and analysis, as they can link data from multiple domains. In the domain of the Internet of Things (IoT), knowledge graphs play a crucial role in harnessing the potential of interconnected devices and sensors through the data they collect [5].

The field of Internet of Things (IoT) has witnessed a surge in interest and research focus in recent years, with IoT knowledge graphs emerging as a crucial tool within the scientific community [21]. IoT knowledge graphs enable a holistic and interconnected view of IoT data, fostering enhanced understanding and decision-making. Knowledge graphs play a pivotal role in addressing the complex challenges associated with data integration in the IoT domain. By facilitating semantic interoperability and unlocking the vast potential of data analytics, IoT knowledge graphs have garnered significant attention.

1.1 Problem Statement

However, a notable challenge in the realm of IoT knowledge graphs is the large size of these graphs, which arises from the heterogeneous and longitudinal nature of the collected data. As IoT devices generate diverse data formats over time owing to their wide range of devices, the resulting knowledge graphs can become remarkably large [15]. Furthermore, the granularity combined with the longitudinal data creates a size concern as the devices track the data history via time points [21], adding to the graph size.

This size issue of knowledge graphs poses significant challenges to their effective utilization [1]. Large knowledge graphs, particularly in the context of IoT, can present difficulties in querying and loading. As the graph increases in size, the complexity of executing queries and retrieving relevant information increases. The sheer volume of data and intricate relationships between entities can lead to longer query response times through resource-intensive processing [2]. Moreover, loading the entire knowledge graph into the memory for analysis becomes impractical as the size escalates. This limitation hampers the efficiency of knowledge graph utilization and inhibits the exploration of valuable insights [10].

1.2 Consumer-grade Devices

The problem narrows down further to working with these knowledge graphs on consumer-grade devices, which makes the issue even bigger due to the limited hardware power. Enabling querying capabilities on consumer-grade devices, rather than relying solely on high-performance computers, is an active area of research as researchers aim to overcome these limitations and empower users to perform queries on their everyday devices. The interest is also from the user end to have control over the IoT data of their surrounding environment, such as their homes [12]. By empowering everyday users to perform queries locally without the need for specialized infrastructure or computational resources, the accessibility and adoption of IoT knowledge graphs can be significantly enhanced. Therefore, it is crucial to develop techniques that address these challenges and enable efficient querying and loading of large IoT knowledge graphs.

1.3 Research Prospect

Sampling techniques have emerged as a practical solution for addressing the size issue of knowledge graphs. Through a selective choice of representative subsets of the graph, sampling enables a more efficient analysis, ensuring a reduced graph size while preserving essential insights within the semantics [8]. In this study, we propose an algorithm to sample IoT knowledge graphs, seeking to reduce their size to a manageable scale with respect to consumer-grade devices, without compromising the computational properties or completeness of the knowledge graph in relation to user queries. So our research question states:

How can we sample IoT Knowledge Graphs to a feasible volume, while maintaining essential computational properties and completeness to effectively answer user-end queries?

To answer this question, we carried out a case study on IoT system data

from a Dutch office building, which falls under the Interconnect project [16], which will be introduced in *Section 2.2*. Our user provided us with questions along with data. We converted the user questions into scenarios that were used for two different tasks: developing filters and later for evaluation. First, we extracted the relevant entities of the knowledge graphs present in the scenarios. These entities assisted us in the selection of relevant information. Relational property filters [3] were implemented in the algorithm to search for the entities. The relations were based on the SAREF [14] ontology, which extends the usability of our algorithm to other knowledge graphs built on SAREF and will be further discussed in *Section 3.1.1*. The algorithm successfully sampled a reasonably reduced knowledge graph. The resulting knowledge graph is evaluated using the entities extracted earlier to check the utility of the graph. We also generated a mapping algorithm to help with the navigation of RDF files within the knowledge graph to search for files for specific devices, as the file names are not self-expressive. Finally, the limitations are discussed, and research that could be conducted in the future to build on the work presented in this study is presented.

Chapter 2

Background Knowledge and Literature

Sampling, a fundamental technique in data analysis, exists in various forms and plays a critical role in addressing the challenges posed by large knowledge graphs. In this section, we delve into a literature study of three different sampling techniques, especially for knowledge graphs. Following the literature study, we provide the necessary background information about the project regarding the framework and the hosting end user.

2.1 Literature Study

The most generic sampling method is random sampling. However, there has been ongoing research on developing techniques that offer more nuanced and targeted ways of selecting data subsets from knowledge graphs, considering specific characteristics and objectives. Three such techniques are systematic sampling, self-supervised sampling, and differentiable sampling. Examining these sampling methods will help us gain insight into their effectiveness and potential for addressing the challenges posed by large knowledge graphs.

2.1.1 Systematic Sampling

Systematic sampling [8] is a novel variant of the random sampling approach that sorts relations before sampling. Given a relation R of N tuples, if n is the number of tuples to be sampled from R, a tuple is selected at random from the first k = $[\frac{N}{n}]$ tuples of R and every *kth* tuple thereafter. The study also investigated the influence of sorted data on the quality of sample relations and the estimation of query result sizes. Empirical experiments conducted by the authors validated that systematic sampling yields higher accuracy than traditional sampling methods [8]. However, the limitation of this method is that it does not consider the semantic characteristics of the graph, which may lead to

an inadequate representation of the graph, potentially missing important connections and patterns. Thus, this approach is not usable in this study because relationships and contextual information are crucial in IoT knowledge graphs.

2.1.2 Differentiable Sampling

Another study introduces the DSKReG (Differentiable Sampling on Knowledge Graph for Recommendation with Relational GNN) model [20], which addresses the key challenge of case-irrelevant data encountered in knowledge graphs contributing to their massive sizes within the context of recommendation systems. The challenges at a deeper level involve node-degree skewness and noisy interactions. To address these issues, the authors propose a differentiable sampling strategy that enhances the effectiveness and accuracy of the recommendations. The evaluation of the DSKReG model through experimental assessments demonstrated its superior performance compared to existing knowledge graph-based recommender systems [20]. This research significantly contributes to the field of knowledge-aware recommendation systems, but the exclusive focus of the technique on recommender systems makes it unreplicable in the domain of IoT knowledge graphs.

2.1.3 Cluster Sampling

There are also sampling techniques with sub-variants, such as cluster sampling [6]. Cluster sampling offers a cost-effective alternative by sampling entity clusters instead of individual triples, thereby reducing identification costs. Variants include weighted cluster sampling, two-stage cluster sampling, and stratified sampling. Weighted cluster sampling enhances estimation accuracy by assigning higher probabilities to larger clusters. Two-stage cluster sampling further reduces costs by estimating cluster accuracies based on triple samples, thereby contributing to improved efficiency. Lastly, stratified sampling enables incremental evaluation of evolving KGs by reusing evaluation results, leading to increased efficiency in evaluation procedures. Experimental results from the study demonstrate the effectiveness of these techniques, showcasing up to 80% cost reduction without compromising the evaluation quality [6]. Although cluster sampling has some heuristics to control the sampling, the clusters are typically heterogeneous. This still keeps the randomness in the equation, thus ruling out the option to be used in our case as clusters in IoT knowledge graphs may not exhibit clear boundaries or homogeneous characteristics.

2.2 Interconnect and VideoLab

The Interconnect Project[1] is an innovative initiative approved by the European Commission, spread across 11 countries with 50 active partners. The international initiative aims to bring efficient energy management within the reach of

[1] https://interconnectproject.eu/

end-users by enhancing the semantic interoperability of IoT data, developing efficient data integration methodologies, and addressing the challenges posed by the heterogeneity of IoT devices and data sources [16]. By leveraging knowledge graphs and linked data principles, this project aims to enable a holistic and interconnected view of IoT data, fostering enhanced understanding and insights for real-world applications. The creation of an interconnected ecosystem of smart devices, sensors, and data sources will enable seamless data exchange and collaboration. The interdisciplinary approach of the project brings together experts from various fields, including computer science, engineering, and data science, to collaboratively tackle the complexities of IoT data and pave the way for a smarter and more interconnected future. One of the 50 partners of this project is VideoLab [18]. The VideoLab[2], which once served as the birthplace of Philips' pioneering DVD and Blu-ray player, now stands as an 8-storey building dedicated to fostering entrepreneurial ventures. Operating under the Office-S label, the VideoLab is now a thriving hub, facilitating entrepreneurship in its broadest and most dynamic sense.

[2] https://strijp-s.nl/ondernemers/videolab/

Chapter 3

Methodology

VideoLab has an in-house smart building equipped with an IoT system in its office. In-house smart buildings utilize various Internet of Things (IoT) devices, sensors, and automation systems to monitor and control different aspects of the building's operations to ensure user comfort [19]. Although the devices and sensors in the VideoLab office are interconnected via IoT systems, automation systems are not available in the network. However, VideoLab is eager to enable some automations within its IoT system. To implement this, they need answers to certain questions, ensuring the necessity of the automations within the system. They hypothesized scenarios in which they would want to have automated systems to take action through their devices. These scenarios were provided to us as a set of parameters for the relevant measurements, along with IoT system data from their devices and sensors. As an example to illustrate the parameters, in a scenario where the temperature is 19 °C, the set of parameters will include a value of 19, labeled as temperature (*see Section 3.2*). The question imposed over the scenarios was *'Does this scenario exist? If so, how frequently?'*.

In combination with the question, these scenarios served as user-end queries for this case study. The aim of this study is to sample the knowledge graph of their IoT system to a smaller size while maintaining the completeness of the data to be able to answer these user questions. Thus, the success of the study will be evaluated by querying the sampled knowledge graph to check if the parameters in the given scenarios are present in the knowledge graph, despite the reduction. *Figure 3.1* shows an overview of the timeline of the methodology. However, prior to diving into the methodology, we deemed it essential to furnish relevant information concerning the utilized data and frameworks.

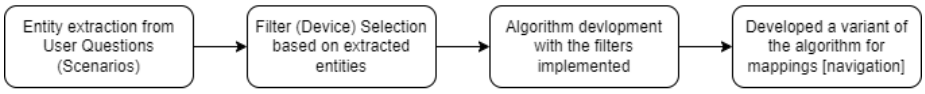

Figure 3.1: Methodology Diagram

3.1 Data and Framework

3.1.1 SAREF Ontology

The Smart Applications REFerence ontology (SAREF)[1] is a standardized and widely adopted framework designed to enable semantic interoperability between IoT devices from different manufacturers that use different protocols and standards [4]. It provides a common vocabulary and structure to represent the characteristics and functionalities of smart appliances and devices. SAREF offers a set of classes, properties, and relationships that facilitates the description and integration of various IoT data sources. *Figure 3.2* shows an overview of the main classes and relationships in SAREF [14]. SAREF contains a total of 81 classes, 35 object properties, and 5 data properties. In our research, we utilized the SAREF ontology as a key component in the implementation of the criteria constraints for the filters within our sampling algorithm. To an extent, this standardizes our algorithm by extending its usability over other IoT knowledge graphs that are built on the SAREF ontology. SAREF has also been used for mapping of the knowledge graph, that we will be using, as a part of another study which will be mentioned in *Section 3.1.1.*

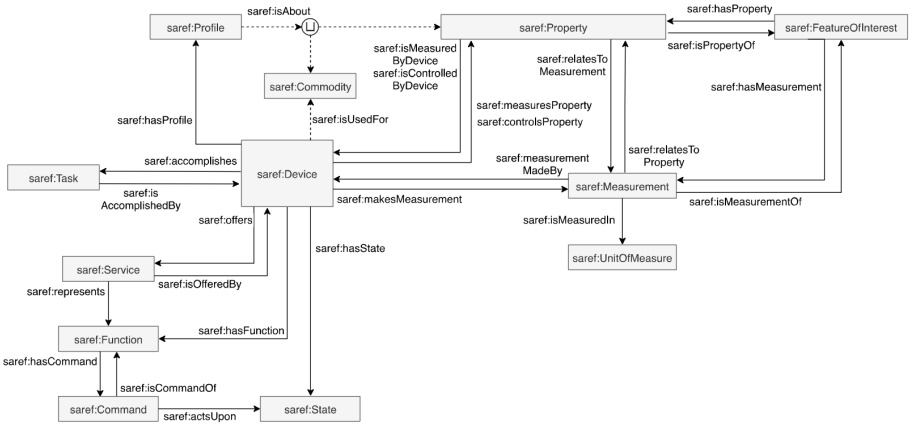

Figure 3.2: Overview of SAREF ontology

Saref ontology overview

3.1.2 GraphDB and SPARQL

GraphDB[2] is a semantic graph database that plays a crucial role in managing and querying large-scale knowledge graphs, including those in the IoT domain

[1] https://saref.etsi.org/

[2] https://www.ontotext.com/products/graphdb/

[7]. SPARQL (SPARQL Protocol and RDF Query Language)[3] is the standard query language and protocol for Linked Open Data on the web or for RDF triplestores [11]. Using the SPARQL query language, researchers and developers can interact with knowledge graphs to efficiently extract valuable insights and knowledge. SPARQL is specifically designed for querying and manipulating data represented in RDF (Resource Description Framework), making it an essential component for exploring IoT knowledge graphs. In the context of this study, we use both GraphDB and SPARQL for the evaluation of our algorithm in *Chapter 4*.

3.1.3 Data Mapping

The IoT system data received from VideoLab were raw heterogeneous data from sensors and devices. However, the data were mapped to RDF triples as part of another study by the authors of this paper, where they performed a similar mapping for another smart-home system [17]. The SAREF ontology was used as the framework for mapping the system data, resulting in our source knowledge graphs. If we refer back to *Figure 3.2*, the *device* block we see is the base class, eventually linking to all other classes via one or a chain of relations. For instance, the *device* class is linked to the measurement being made stored in the $measurement$ class via the $saref : makesMeasurement$ relation. Because the *device* class is the base class, the RDF data is spread between RDF files, where each file represents one device and holds the corresponding data. Additionally, SAREF has extensions of its ontology tailored to specific domains. *SAREF4BLDG*[4] is an extension for the building domains which has been used in our knowledge graphs, as our data concerns a smart building.

3.2 Entity Extraction from Scenarios

The scenarios provided by our user were in terms of various device and sensor measurements that are relevant for the automations to take place. *Table 3.1* shows the scenarios defined by a set of parameters forming the table columns, holding specific values expected from the measurements. The table also provides a description for each of the scenarios to better convey how the set of parameters form the scenarios as a whole.

The columns of the table are what we are interested in as they display the mode of measurement that is being made. As we are trying to make a more generic algorithm to be able to sample for a wider range of questions, and not specifically, this set of five questions, we extract the information from the table that relates to the entities in the table instead of hard-coding it to specific

[3] https://www.w3.org/TR/rdf-sparql-query/
[4] https://saref.etsi.org/saref4bldg/v1.1.2/

]

Srno.	Time	Window	Thermostat	Description
1	Office hours	Open for 15 mins	T = 15° C	During Office hours if the window is open for 15 mins or more, the temperature of the thermostat on the radiator is reduced to 15 deg C.
2	Office hours	Closed for 15 mins	T = 22° C	During Office hours if the window is closed for 15 mins or more, the temperature of the thermostat on the radiator is set to 22 deg C.
4	Office hours	If Offline	T = 19° C	During Office hours if the window is Offline, and irrespective of the movement sensor, the temperature of the thermostat on the radiator is set to 19 deg C.
5	18:00 to 07:00		T = 15°C	Outside office hours from 18:00 hrs to 07:00 hrs, the temperature is set to 15 deg C
6	Sundays		All Thermostats go from minimum to maximum to minimum.	On request from Kuijpers, to maintain the radiators, it helps to move the pin inside the thermostat frequently. This automization increases the temperature to maximum and brings it back to minimum once a week on Sunday.

Table 3.1: User-provided automation scenarios

values. After analyzing the scenarios, these were the four entities we extracted, which we deemed necessary to answer the questions:

- **Time**: The scenarios are applicable for certain time ranges, which will be restricted by the time data in the knowledge graph. The descriptions also mention monitoring the duration over which a certain activity is exhibited before the automation can be enabled. The timestamps of these activities from other devices are also required to set this constraint.
- **Temperature**: All the suggested automation is regarding the monitoring and adjustment of the temperature of the thermostat and the radiators, making the temperature data an important factor to assess the utility of the automation.
- **Windows Status**: The windows being open or closed is also a relevant factor to be monitored for some of the scenarios, which makes the data regarding the status of the windows essential for the answers.

- **Location**: We also decided the location where the measurements are being recorded is also relevant information for answering the questions as our system environment is not a single room but a building. Thus, we need the location information to ensure that the room where the window data are from is also the room where the corresponding temperature data are recorded.

3.3 Filter Selection - Devices

As we sought to base the algorithm on a generic level of semantics, we explored the knowledge graph. We sampled a small knowledge graph and loaded it into graph DB, where we queried it to explore the relations and entities. Although the sampling was random, we manually selected some parts of the sample to ensure that we obtained a variety of devices. In an attempt to look for a general class linking our extracted entities as well as consulting back to the SAREF overview in *Figure3.2*, the *device* class was found to be linked to all the required information that we sought. The IoT system of VideoLab has a wide range of devices; thus, the selection of devices with relevant measurements will reduce our knowledge graph size. Therefore, we chose to use the device type as our filter for the sampling algorithm. An overview of the devices and their related properties is shown in *Figure 3.3*, which we used for the selection of our devices.

Each column in the table in *Figure 3.3* represents a distinct device. The first row shows the number of devices we have for each of the types, summing up to a total of 1,462 devices. The rows represent the properties being monitored and altered by the devices. The highlighted cells with a cross indicate that the device holds that certain property of the row, whereas the other blank cells indicate that they do not have the data of that property. After carefully reviewing the table, we selected the following devices for their data to be included in our sampled knowledge graph:

- **Zigbee Thermostat**: Although quite some devices have the temperature data all the temperature constraints in the scenarios are linked to either the thermostat or radiator, probably due to the availability of the heating and cooling ability property. This makes the interest in the temperature data exclusive to these two devices. One of the two types of thermostat devices is the Zigbee Thermostat, which has the relevant temperature data that our user is interested in.
- **Z-Wave Radiator Thermostat**: This is the second of the two types of thermostats that are there, so it also holds the data that the user is interested in due to the same reason mentioned above.

		SmartSense Motion Sensor	SmartSense Multi Sensor	Zigbee Thermostat	Airwits CO2 Sensor	SmartPower Outlet	Calumino People Counter	SmartSense Moisture Sensor	Z-Wave Door/ Window Sensor	Z-Wave Radiator Thermostat	Awair Omni Air Quality Detector
		393	377	358	227	30	14	14	6	1	
measurements	Temperature	x	x	x	x			x		x	x
	Humidity				x						
	CO2				x						
	Power Usage					x					
	People Count						x				
Movement SELF	Accelerometer X		x								
	Accelerometer Y		x								
	Accelerometer Z		x								
	Accelerometer status		x								
Metadata	Seconds [[Running Time]]						x				
	MAC Address						x				
	Firmware Version						x				
	Location - Building	x	x	x		x		x	x	x	x
	Location - Room	x	x	x		x		x	x	x	x
	Device Status	x	x	x		x		x	x	x	x
	Battery Voltage Level	x	x	x				x	x	x	
Status	Contact Status		x						x		
	Tamper Status								x		
	Motion Detected	x									
	Water Detection Status							x			
	Device Power Source			x							
Heating	Heating Set Point			x						x	
	Heating Lower Range			x						x	
	Heating Upper Range			x						x	
	Thermostat Temperature			x						x	
	Thermostat Mode			x						x	
	Fan Mode			x						x	
Cooling	Cooling Upper Range			x							
	Cooling Lower Range			x							
	Cooling Set Point			x							

Figure 3.3: Devices and their properties

- **Z-Wave Door/ Window Sensor**: The only other device type that we need in our sample as it holds the data of the status windows which is one of the entities we extracted.

As all three selected categories of devices contain location data, no specific device related to the location entity is required. The selected devices together make a total of 365 devices, approximately 25% of the total number of devices. Therefore, we aim to make a significant size reduction to our knowledge graph.

3.4 Sampling Algorithm

Now that we have the filters, we introduce our strategy to implement them in the form of an algorithm. As mentioned earlier in *Section 3.1.3*, the knowledge graph is distributed among a set of RDF files, each holding the data for a distinct device. Our approach attempts to identify target relations in the files and sample them upon success.

The algorithm begins with an extra step of locating the knowledge graph. File extensions in the directory of the source knowledge graph are checked to make a selection of all the RDF files through the *'.ttl'* extension, thus locating the knowledge graph. This step is followed by extracting the data from the

knowledge graphs by reading the content of the selected files. Once the file data are read, the algorithm searches for the desired filters in terms of semantic relations in the data. These relations are implemented in the form of knowledge triples, where we add the last two contents of the triple as a condition to be found in the data.

These two contents, the predicate, also called the relation, and the object, which is an entity, is where the fine-tuning capability of the algorithm is held. These two parameters can be used to tune the algorithm to the information we are looking for in the data. For instance, if we are looking for a certain device or set of devices A and we know that A has relations B or related object C exclusive to it, we can add B or C in the parameter to sample a knowledge graph with A exclusively. For our case study, we tuned it to look for specific entities, namely models of the devices as that is our selected filter. Finally, if the parameters are found in the data, we sample them; otherwise, we skip them. *Algorithm 1* shows the pseudocode for the algorithm. Looped if-else statements are added to provide deeper insight into the algorithm; usually, the code has a set of desired parameters that are looped over[5]

```
for each file in directory do
    if file has extension .ttl then
        read file data
        if data has relation 'saref :
          hasModel"Z − WaveDoor/WindowSensor"8sd : string' then
            add file to sample
        else
            if data has relation
              'saref : hasModel"ZigbeeThermostat"8sd : string' then
                add file to sample
            else
                if data has relation
                  'saref : hasModel"Z − WaveRadiator' then
                    add file to sample
                end
            end
        end
    end
end
```

Algorithm 1: Sampling Algorithm Pseudocode

[5]The code can be found in our Github repository `https://github.com/raihan-karim-ishmam/Sampling-IoT-Knowledge-Graph-to-Enhance-Computational-Efficiency`

3.5 Algorithm Variant for Mapping

This section of the methodology is an extension of our aim of sampling the IoT knowledge graph. Our design rationale for this variant of the algorithm is based on the challenges faced with the cryptic names of the RDF files that constitute our knowledge graph. The file names are the device IRIs, which make it difficult to navigate through them to locate any specific set of devices. Although we can query the files once we have sampled the knowledge graph to a reasonable size, this problem persists if we want to locate our sample knowledge graph in another knowledge graph without resampling.

This variant of the algorithm has the same pseudocode as that demonstrated by *Algorithm 1*, with minor modifications [6]. The parameters are kept the same as we look for our custom set files, but they can be modified to locate other devices with corresponding distinct properties. In addition, we have a counter to keep a record of the number of devices as well as the number of distinct devices for a certain property. The distinct property in our case was that of the device model. Another modification is in the last step, instead of sampling, we output the file name along with the counters and the filtered property, either in the console or as a document shown in the figure below. *Figure 3.4* shows a section of the output document for the mapping of our sampled knowledge graph.

Index	File Name	Counter
1	urn_Device_SmartThings_000d09d0-fe11-47a8-b886-6a50cae01027.ttl	Zigbee Thermostat1
2	urn_Device_SmartThings_0148e5c8-6cd3-40a9-9e63-7117a25fb3b7.ttl	Zigbee Thermostat2
3	urn_Device_SmartThings_016991c1-26b5-4e1e-92a1-141d514f410b.ttl	Zigbee Thermostat3
4	urn_Device_SmartThings_022aa775-bb1e-497c-926f-446af7e0c795.ttl	Zigbee Thermostat4
5	urn_Device_SmartThings_02b0d079-4f7d-4773-a3dc-c2dd76753ecf.ttl	Zigbee Thermostat5
6	urn_Device_SmartThings_02ec340f-b394-4a7e-892a-3fa75540d7a5.ttl	Zigbee Thermostat6
7	urn_Device_SmartThings_037d8c73-1828-4ab5-9378-26d72b260bcc.ttl	Zigbee Thermostat7
8	urn_Device_SmartThings_04927462-3f3f-465a-ab66-577f3ae97ad5.ttl	Zigbee Thermostat8
9	urn_Device_SmartThings_05deeda9-429f-41b6-ae1b-f9c1562c9416.ttl	Zigbee Thermostat9
10	urn_Device_SmartThings_05f85fd0-7bd7-4479-8196-84be463f7117.ttl	Zigbee Thermostat10
11	urn_Device_SmartThings_0633707a-c7ed-470d-954a-13bdabfadf96.ttl	Zigbee Thermostat11
12	urn_Device_SmartThings_07f708f9-ed9b-49e9-ac75-cced6dae09fe.ttl	Zigbee Thermostat12
13	urn_Device_SmartThings_08e7f0a4-0a68-4ed0-9491-1fbbace3a5f1.ttl	Zigbee Thermostat13
14	urn_Device_SmartThings_08eea511-36e9-4ba2-b389-156c03993770.ttl	Zigbee Thermostat14
15	urn_Device_SmartThings_0958b282-14f2-44a3-afd4-44cec43dfd98.ttl	Zigbee Thermostat15
16	urn_Device_SmartThings_0a1498b7-650a-48d5-9c69-37731e181bbb.ttl	Zigbee Thermostat16
17	urn_Device_SmartThings_0b4670d7-8572-479a-858c-d914686ee17b.ttl	Zigbee Thermostat17
18	urn_Device_SmartThings_0c0bde8d-5f2f-4478-8065-16a308f34b15.ttl	Zigbee Thermostat18
19	urn_Device_SmartThings_0c34c0bb-bc30-4b0b-8468-c33d1cfc31af.ttl	Z-Wave Door/Window Sensor1

Figure 3.4: Mapping of sampled KG

[6]The code can be found in our Github repository `https://github.com/raihan-karim-ishmam/Sampling-IoT-Knowledge-Graph-to-Enhance-Computational-Efficiency`

Chapter 4

Evaluation

As expected, the sampling algorithm successfully sampled 365 files. Despite variations among the sizes of each file, the knowledge graph was reduced to 25% of the size of the source knowledge graph. Therefore, we observed a close relationship between the devices and the size proportions between the source and the sampled knowledge graph. In this section, we evaluate the utility of the resulting knowledge graph. To assess whether the knowledge graph maintained the computational properties required to answer the user questions, we ran SPARQL queries to determine whether the knowledge graph had the extracted entities *(see Section 3.2)* required to answer the questions. The resulting knowledge graph was loaded into GraphDB, where the SPARQL queries were run on it.

4.1 Temperature agaisnt Time

```
PREFIX ic: <https://interconnectproject.eu/example/>
PREFIX saref: <https://saref.etsi.org/core/>

SELECT DISTINCT (?value AS ?Temparature)
                (?dateTime AS ?Time)
WHERE {

    ic:device_urn:Device:SmartThings:9a6268d1-e0cd-484d
    -95cd-42737fa4bce0
                saref:makesMeasurement ?measmeasurement.
    ?measurement saref:relatesToProperty ?property.
    ?property a saref:Temperature.
    ?measurement saref:hasvalue ?value;
                saref:hasTimestamp ?dateTime.
```

Listing 4.1: SPARQL query for temperature and time

A thermostat device was randomly chosen to obtain an insight into how the temperature recorded by the thermostat varied over time. *Listing 4.1* shows the query used to extract the temperature and time data of the device. *Figure 4.1* depicts a visualization of the temperature against time from the query results plotted using *Matplotlib*[1] [22]. By observing the patterns and fluctuations depicted in the visualization, we can gain insights into the impact of certain times or activities during those times on the temperature. *For a monthly overview of the temperature data, refer to the appendix.*

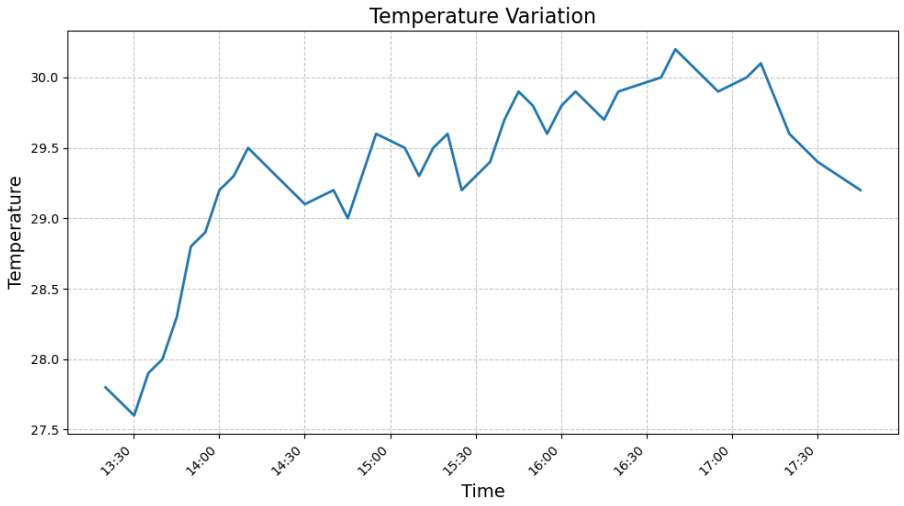

Figure 4.1: Temperature Variation Hourly

4.2 Windows Status against Time

```
PREFIX ic: <https://interconnectproject.eu/example/>
PREFIX saref: <https://saref.etsi.org/core/>

SELECT DISTINCT (IF(?value = 1, 'Open', 'Close') AS ?Status)
                (SUBSTR(STR(?dateTime), 1, 10) AS ?Date)
                (SUBSTR(STR(?dateTime), 12, 8) AS ?Time)

WHERE {
```

[1] https://matplotlib.org/

```
  ic:device_urn:Device:SmartThings:0c34c0bb-bc30-4b0b-8468-
      c33d1cfc31af
         saref:makesMeasurement ?measmeasurement.
  ?measurement saref:relatesToProperty ?property.
  ?property a ic:Contact.
  ?measurement saref:hasvalue ?value;
         saref:hasTimestamp ?dateTime.
}b
```

Listing 4.2: SPARQL query for window status and time

The same approach as in the previous evaluation case was replicated, but this time a random windows sensor was picked. It is necessary to emphasize that the mappings mentioned in *Section 3.5* were used for both device selections in this and the previous use case. We decided not to plot the data because the values were binary, *open* or *close*. *Listing 4.2* shows the query used to extract the window status and time. The time and month were output separately to improve comprehensibility. *Table 4.1* presents the query results. This use case demonstrates the status of windows, and the corresponding times could be used, for instance, to compare against the temperature for a specific time range.

Status	Date	Time
Open	3/4/2022	18:30:03
Close	3/4/2022	23:00:05
Open	3/5/2022	9:20:01
Open	3/5/2022	9:25:02
Open	3/5/2022	12:10:02
Open	3/5/2022	12:45:02
Close	3/5/2022	13:50:04
Open	3/5/2022	13:55:02
Open	3/5/2022	14:15:03
Open	3/5/2022	14:30:03
Close	3/5/2022	14:55:03
Open	3/5/2022	15:00:02
Open	3/5/2022	15:20:03

Table 4.1: Windows status and corresponding times

4.3 Sparql Query 3

```
PREFIX ic: <https://interconnectproject.eu/example/>
PREFIX saref: <https://saref.etsi.org/core/>
PREFIX s4bldg: <https://saref.etsi.org/saref4bldg/v1.1.2/>
PREFIX s4ener: <https://saref.etsi.org/saref4ener/v1.1.2/>
SELECT DISTINCT ?Model
                (substr(str(?Serial), 29) as ?Serial_Number)
                (substr(str(?Room), 66) as ?Room_IDs)
WHERE {
  ?dev saref:makesMeasurement ?Measurement;
       s4ener:serialNumber ?Serial;
       saref:hasModel ?Model;
       s4bldg:isContainedIn ?Room.
}
```

Listing 4.3: SPARQL query for device location

In addition to the results derived from the previous queries, the locations where the measurements were made need to be considered to determine their effect on each other. The knowledge graph was queried for the respective device locations corresponding to the device measurements, as shown in *Listing 4.3*. The query extracts the device model with their serial number to illustrate the distinction between the devices of the same model and the room IDs exposing their location. Device IDs as those in *Listing 1* and *2*, can also be included if needed, but we have excluded them in our query for visual clarity. Figure 3 outlines the query results for device location information.

	Model	Serial_Number	Room_IDs
1	"Z-Wave Door/Window Sensor"	"935-583c-4b3e-8b3c-c32d2f252721"	"3e349f0c-9ed0-48b2-957f-4212b0c16c68"
2	"Z-Wave Door/Window Sensor"	"0bb-bc30-4b0b-8468-c33d1cfc31af"	"384101b1-466e-45a3-889d-4b2d938ed4aa"
3	"Z-Wave Radiator Thermostat"	"57f-d857-4a89-964c-7be035df0751"	"5b9ed2e2-6ec1-48ce-9dac-98bccd8f4870"
4	"Zigbee Thermostat"	"079-4f7d-4773-a3dc-c2dd76753ecf"	"1fab20af-8cce-4893-bec9-ad0a54e68045"
5	"Zigbee Thermostat"	"40f-b394-4a7e-892a-3fa75540d7a5"	"b0f601e1-d85f-4524-907e-ada8eb489433"
6	"Zigbee Thermostat"	"087-7927-4ba1-831c-d19273fe459c"	"b0f601e1-d85f-4524-907e-ada8eb489433"
7	"Zigbee Thermostat"	"2cc-b087-44d0-b7c7-2915c33b81db"	"ed73d598-b9ed-4448-9ece-e553e4b23e32"
8	"Zigbee Thermostat"	"61b-9e76-4094-aa66-49a484860e01"	"4e9497bb-5174-43cb-82b5-5fc9067a220f"
9	"Zigbee Thermostat"	"6d9-2ae2-41d9-a644-ab697b0fb71c"	"9bdc8aaa-ef05-4bf6-9b97-8e0190046a12"
10	"Zigbee Thermostat"	"e1c-efae-4a51-a980-76ba65d15d29"	"885ed11e-9713-41bc-b8f3-983345e122b0"
11	"Zigbee Thermostat"	"ddd-baa2-4a6a-b797-9eaf22fd1d71"	"9745c60d-0417-47f9-8c89-01e6d2ec869f"

Figure 4.2: Devices and their locations

Chapter 5

Discussion and Conclusion

In conclusion, this study aims to address the primary issue of large-sized IoT Knowledge Graphs to mitigate the inefficiency in loading and querying. A case study was conducted to investigate the sampling of an IoT knowledge graph to achieve a manageable volume while preserving the crucial computational properties and completeness to provide answers to user-end queries.

This study presents an algorithm designed to sample IoT knowledge graphs based on the SAREF ontology using semantic relation-based filters centered on the base class of the ontology, the "Device" class. The semantic filters administered in the algorithm have the potential to fine-tune the sampling to the desired properties. The algorithm successfully reduced the knowledge graph size by 75%, while retaining the computational utility for answering relevant user queries. Finally, we evaluated the resulting sampled knowledge graph by querying the graph with decisive entities extracted from user-provided automation scenarios. The results of the queries showcased the ability to maintain completeness and serve the intended objective by outputting the desired data. The reliance of the algorithm on the SAREF ontology provides a strong foundation for its effectiveness in dealing with other SAREF-based IoT knowledge graphs.

However, this research also acknowledges some limitations. The applicability of the algorithm is currently limited to IoT knowledge graphs constructed using SAREF ontology. In addition, the algorithm assumes the knowledge graph's representation through distinct RDF files for each device with their respective data, as SAREF infers each device as a knowledge base [13]. This exclusivity is in regard to the algorithm being developed based on the SAREF framework. For the algorithm to be effective, it requires the presence of properties (relations or related entities) that distinctly differentiate the devices or data of interest from other devices, or those lacking the desired data.

5.1 Future Work

Our study has laid the baseline for future research to attempt to answer the user questions from VideoLab, using our sampled knowledge graph. The prospect could to extended to address more scenarios involving properties not touched upon in our paper. In future research, it would be worthwhile to explore ways to extend the applicability of the algorithm to other standard ontologies and to adapt it to handle diverse data representations. Furthermore, efforts to enhance the flexibility of the algorithm and overcome limitations in differentiating devices or data sources could lead to more comprehensive sampling techniques. By addressing these aspects, the potential of the algorithm can be broadened to encompass a wider array of IoT domains and scenarios, thereby contributing to the advancement of efficient IoT knowledge graph analysis and utilization.

This study makes no definitive claim regarding the full usability of consumer-grade devices for sampled Knowledge Graphs. Nevertheless, by undertaking this research on a device of this grade, we have taken a small yet significant step towards advancing the field of knowledge graph sampling and enabling research to be conducted effectively on consumer-grade platforms. This contribution brings us closer to harnessing the potential of everyday devices and opens new possibilities for expanding the utilization of Knowledge Graphs in practical applications.

Acknowledgement

I wish to extend my sincere appreciation to my supervisor, Victor de Boer, for his constant guidance and support throughout this thesis project. Victor's invaluable insights, continuous guidance, and constructive feedback greatly shaped the structure and direction of this paper. I am also deeply grateful to Roderick van der Weerdt for his pivotal role in this project. Roderick's commitment to facilitating access to essential resources substantially contributed to the realization of this research.

Bibliography

[1] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

[2] Nurendra Choudhary, Nikhil Rao, Sumeet Katariya, Karthik Subbian, and Chandan K. Reddy. Self-supervised hyperboloid representations from logical queries over knowledge graphs. In *Proceedings of the Web Conference 2021*, WWW '21, page 1373–1384, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Sutanay Choudhury, Khushbu Agarwal, Sumit Purohit, Baichuan Zhang, Meg Pirrung, Will Smith, and Mathew Thomas. Nous: Construction and querying of dynamic knowledge graphs. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1563–1565, 2017.

[4] European Commission, Content Directorate-General for Communications Networks, Technology, W Strabbing, P Stapersma, B Roelofsen, L Daniele, and A Aalberts. *Study on ensuring interoperability for enabling Demand Side Flexibility*. Publications Office, 2019.

[5] Alex Donkers, Dujuan Yang, and Nico Baken. Linked data for smart homes: Comparing rdf and labeled property graphs. 06 2020.

[6] Junyang Gao, Xian Li, Yifan Ethan Xu, Bunyamin Sisman, Xin Luna Dong, and Jun Yang. Efficient knowledge graph accuracy evaluation. *Proc. VLDB Endow.*, 12(11):1679–1691, jul 2019.

[7] GUTING R. H. Graphdb : Modeling and querying graphs in databases. *Proc. of the 20th VLDB Conf., 1994*, 1994.

[8] B. Harangsri, J. Shepherd, and A. Ngu. Selectivity estimation for joins using systematic sampling. In *Database and Expert Systems Applications. 8th International Conference, DEXA '97. Proceedings*, pages 384–389, 1997.

[9] Pascal Hitzler and Krzysztof Janowicz. Linked data, big data, and the 4th paradigm. *Semant. Web*, 4(3):233–235, jul 2013.

[10] Chih-Yuan Huang and Yu-Jui Chang. An adaptively multi-attribute index framework for big iot data. *Computers Geosciences*, 155:104841, 2021.

[11] Mohamed Kharrat, Anis Jedidi, and Faiez Gargouri. Sparql query generation based on rdf graph. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, IC3K 2016, page 450–455, Setubal, PRT, 2016. SCITEPRESS - Science and Technology Publications, Lda.

[12] JeongGil Ko, Byung-Bog Lee, Kyesun Lee, Sang Gi Hong, Naesoo Kim, and Jeongyeup Paek. Sensor virtualization module: Virtualizing iot devices on mobile smartphones for effective sensor data management. *International Journal of Distributed Sensor Networks*, 11(10):730762, 2015.

[13] Roberto Reda, Antonella Carbonaro, Victor de Boer, Ronald Siebes, Roderick van der Weerdt, Barry Nouwt, and Laura Daniele. Supporting smart home scenarios using owl and swrl rules. *Sensors*, 22(11), 2022.

[14] SAREF. Smart applications reference ontology, and extensions.

[15] Rohith Teja. Why you should pay more attention to knowledge graphs?, Aug 2021.

[16] Eliana Vales. Interconnect blog - interoperability as a requirement, Jul 2020.

[17] Roderick van der Weerdt, Victor de Boer, Laura Daniele, and Barry Nouwt. Validating saref in a smart home environment. In Emmanouel Garoufallou and María-Antonia Ovalle-Perandones, editors, *Metadata and Semantic Research*, pages 35–46, Cham, 2021. Springer International Publishing.

[18] VideoLab.

[19] D. S. Vijayan, A. Leema Rose, S. Arvindan, J. Revathy, and C. Amuthadevi. Automation systems in smart buildings: A review. *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[20] Yu Wang, Zhiwei Liu, Ziwei Fan, Lichao Sun, and Philip S. Yu. Dskreg: Differentiable sampling on knowledge graph for recommendation with relational gnn. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 3513–3517, New York, NY, USA, 2021. Association for Computing Machinery.

[21] Rongxin Xu, Qiujun Lan, Shiva Raj Pokhrel, and Gang Li. A knowledge graph based survey on distributed ledger technology for iot verticals. *ACM Comput. Surv.*, jul 2023. Just Accepted.

[22] Aldrin Yim, Claire Chung, and Allen Yu. Matplotlib for python developers: Effective techniques for data visualization with python.

Chapter 6

Appendix

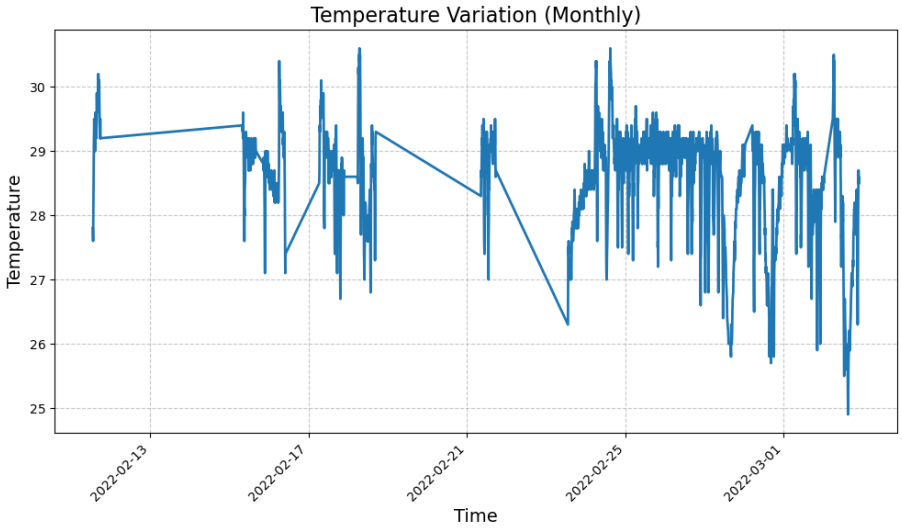

Figure 6.1: Temperature Monthly Overview