ML-based Translation Methods for Protocols and Data Formats

Tamas Tothfalusi
Industrial IoT Division
AITIA International Inc.
Budapest, Hungary
tothfalusi@aitia.ai

Eszter Varga, Zoltan Csiszar and Pal Varga
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
pvarga@tmit.bme.hu

***Abstract*—In order to exchange information between systems, the information must get encoded into a predefined data format, and it must be transferred in a protocol that the communicating parties have agreed upon. This works well if all parties follow the same protocol standard and use the same data description schemes. If systems use different data formats or protocols, then some sort of translation is required. Protocol and data format translation has been attempted previously through rule-based approaches, ontologies, and also by using machine learning (ML) techniques. Due to the current advances related to AI/ML methods, tools, and infrastructure, the accuracy and feasibility of "translation" with ML-approaches improved significantly. This paper introduces a generic approach and methodology for translating data formats and protocols with ML-based methods and presents our initial results through JSON-XML and JSON-SenML translation.**

***Index Terms*—protocol translation; machine learning; neural machine translation; natural language processing; LLM**

I. Introduction

Successful information exchange requires communicating systems to follow the same data formats and the same protocol dialogue setup, as well as error handling.

Heterogeneous, distributed, cyber-physical system-of-systems (CPSoS), however, often lack complete protocol or data structure matching – in which case the information exchange either fails or one of the parties must align. One way of alignment is based on the "robustness principle" (be conservative in what you send, and be liberal in what you accept), where either the receiver part needs to be intelligent enough to adjust its understanding to the protocol or the data format – or there should be a translator in between the sender and the receiver(s) that align the data and the protocol dialogue as expected by the receiver. We refer to this alignment in understanding as "translation".

Translating textual information between natural languages has improved very significantly in the last decade. This is primarily due to the improvement of Artificial Intelligence (AI) and Machine Learning (ML) methods, tools, and, most importantly, its engineering infrastructure.

This paper focuses on the possibilities of ML-based translation for machine-to-machine (M2M) type communication patterns – namely, the translation of data structures and protocols.

The contributions of this paper are the following: i) it provides a timely overview on M2M communication translation approaches and results; ii) it introduces a generic methodology to ML-based translation of data formats and M2M protocols; iii) it presents initial but successful translation results for JSON-XML and JSON-SenML example datasets.

This paper is structured in the following way. Section II provides an overview of current M2M translation approaches. Section III introduces our suggested methodology for ML-based M2M data format and protocol translation, supported by examples with language translation tools. Section IV presents our current results with this methodology and toolset applied to JSON-XML and JSON-SenML translations. Section VI presents our initial results on a ChatGPT-based translation test. Section VI summarizes the findings and suggests a path for future works in the domain.

II. Related Work

Interoperability among systems is best achieved by using a common implementation of the same standardized protocol. This is not always feasible, unfortunately.

Interoperability within heterogeneous CPSoS can be achieved by real-time translation [1]. Besides mapping data formats, however, multiple challenges appear in protocol error propagation handling [2], and tackling the security- and privacy-related issues [3] in autonomous CPSoS with dynamic handling of interoperability issues. The Eclipse Arrowhead Framework addresses such interoperability issues [4] of CPSoS, and is dedicated to providing protocol translators [5] as part of the Industry 5.0 [6] initiative.

The XML- [7] and JSON-based [8] data representations are commonly used data formats to exchange information. Effective translation between these representations has an immediate practical impact. Several solutions have been developed to translate between these formats in recent years [9], [10]. We use these as a baseline.

Translating between XML, or JSON – XML schemas is not always straightforward, but it is important to validate the XML/JSON translation results. There are no RFCs or standards to map the data values. This is where ontology could help by adding semantic annotations and supporting metadata translation. F. Moutinho et al. [11] presented a method to solve the interoperability problems between systems which use different XML schemas. The proposed solution is integrated with the Arrowhead frameworks, and it automatically instantiates XML translators based on semantic annotations [12]. G. Amaro [13] et al. further elaborated on this method and extended the work to support JSON- and XML-based systems.

While investigating the recent research works on AI- and ML-based methods regarding IoT platforms, we found that the models are principally used for data analysis [14], real-time prediction [15], and furthermore, for security reasons [16].

Neural Machine Translation (NMT) [17] [18] is a state-of-the-art, Neural Network-based technique to translate between languages. The training phase operates with dictionaries, which are formed into word pairs after a preprocessing phase.

Pasindu [19] et al. presented techniques to improve the performance of Neural Machine Translation methods in language translation. The authors applied the OpenNMT [18] framework for the evaluation work. The results show that translating between grammatically similar languages could be improved by using a middle language.

III. A MODERN APPROACH TO DATA AND PROTOCOL TRANSLATION

A. *Interoperability issues and Machine Learning*

Industry 5.0 integrates intelligent digital technologies, such as dynamically interconnected CPSoS and AI-optimized IoT solutions, to speed up the manufacturing processes.

Since each vendor and production integrator may use a different combination of the currently popular data formats, compatibility between the CPS endpoints is not guaranteed. A new problem appeared in this domain: interoperability issues. To solve the communication needs between these closed domains, we can investigate Machine Learning (ML) solutions as a possible approach. ML is a subdomain of Artificial Intelligence (AI), aiming to provide models for automatic operation from the previously learned knowledge base without programming. In the following sections we present our approach and initial results on ML-based data format translation.

B. *Methodology*

The general idea of the ML-based data format translation comes from the success of Natural Language processing. We can treat M2M communication data formats as structured and somewhat rigid language sentence formats. We suggest applying the same tools for M2M data format translation as for NLP; although for the M2M case, we feed the models and tools with a great volume of data schema example pairs of the two (from-to) data formats. M2M translation could be compared to a human ”meeting protocol” translation. This is more like a format exchange from a meeting session of one human culture to a meeting format of another human culture. In protocol translation, we need to consider fundamental differences in information exchange, which is a very complex matter. On the other hand, protocol message translation could be much simpler, so the current paper stretches just as far as protocol messages. Our suggestion is to handle protocol messages as multi-embedded heterogeneous data formats. In this case, the challenge gets simplified to the following tasks:

1) identification of protocol encodings, message types, and versions,
2) identification of message embedding boundaries, and
3) data format translation.

The current paper focuses on the challenges of the latter.

C. *Tools and Architecture*

XML and JSON are well-known and widely used protocols to represent data models, piping and instrument diagrams, sensor measurement values, configurations, and other information.

Listing 1: XML data format

```
<bookstore>
  <book category=”fiction”>
    <title lang=”en”>Harry Potter</title>
    <author>J.K. Rowling</author>
    <year>1997</year>
    <price>20.00</price>
  </book>
</bookstore>
```

They have distinct structures for representing data, which is often predefined in a schema. JSON employs a straightforward key-value pair structure, making it easy to parse and understand, while XML utilizes nested hierarchical tags, offering flexibility in expressing complex data relationships. Listing 1 and Listing 2 represent the same content in XML and also in JSON format.

Listing 2: JSON data format

```
”bookstore”: {
  ”book”: [
    {
      ”category”: ”fiction”,
      ”title”: { ”@lang”: ”en”, ”#text”: ”Harry Potter” },
      ”author”: ”J.K. Rowling”,
      ”year”: 1997,
      ”price”: 20.00
    }
  ]
}
```

If we investigate these two data representation formats, we can notice two key points:

1) Both have text-based, human-readable formats.
2) Both have recognizable and predefined, schema-related semantics.

Based on these key features, the translation needs regarding XML-to-JSON, JSON-to-XML, XML-to-XML, and JSON-to-JSON correspond to a text-to-text, or more specifically, a language translation task.

D. *AI/ML tools for language translation*

In recent years, the usage of AI/ML methods to translate between languages or dialects became more and more popular. If we examine the currently available techniques, three categories could be highlighted: i) Neural Machine Translation (NMT), ii) Natural Language Processing (NLP), and iii) Large Language Model (LLM)-based translation.

Neural Machine Translation (NMT) is an AI-driven translation technique. It operates and trains neural networks on

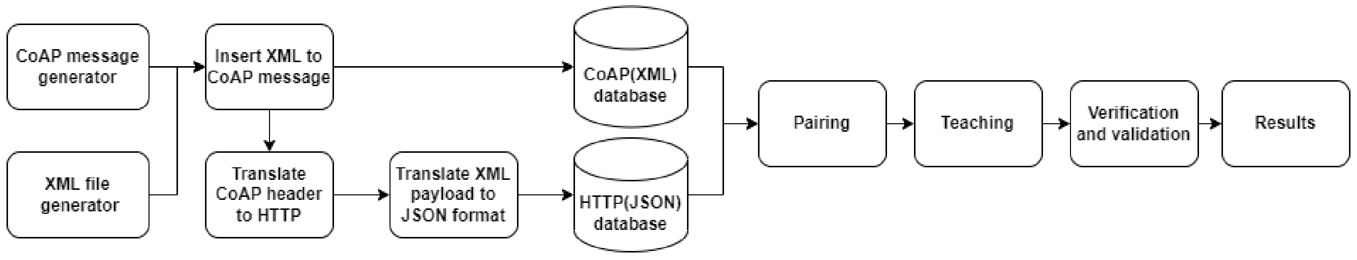

CoAP message generator
XML file generator
Insert XML to CoAP message
Translate CoAP header to HTTP
Translate XML payload to JSON format
CoAP(XML) database
HTTP(JSON) database
Pairing
Teaching
Verification and validation
Results

Fig. 1: Workflow stages

Tool name	Note
PyTorch	An open-source library for DL.
Scikit-learn	An ML library for data analysis.
SpaCy	NLP library with pre-trained models.
Google VertexAI	Pre-trained AI models.
ChatGPT	Automatic content generation.
Wit	Extraction of information from sentences.
Huggingface	Pre-trained AI models.
NLTK	Open-source NLP library.
LangChain	A framework to work with language models.
OpenNMT	Open-source library for NMT-related models.

TABLE I: Machine Learning tools and libraries

large amounts of parallel (text-based) data to learn the relationships and patterns between expressions and words in different languages. NMT and deep learning have shown significant improvements in the accuracy of machine translation.

Natural Language Processing (NLP) constitutes a sector of AI dedicated to understanding and analyzing human language. It has a variety of applications, including search engine optimization, automating customer service interactions, and facilitating textual translation tasks.

Large Language Models (LLM) are the latest approach in the AI-driven solution workspace. It got famous based on its groundbreaking results in generative AI, through tools such as ChatGPT [20]. These tools can already be used as native translators for certain data formats that are included in the language model training.

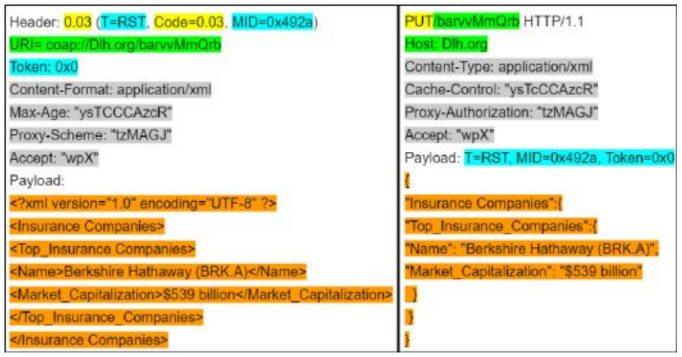
Table I summarizes the state-of-the-art ML tools, regarding fast language processing and translation.

IV. Test bed and model creation

A. The architecture of the test environment

Figure 1 represents the stages of the work process. First, we created a CoAP message generator and an XML generator, which were used to generate random content. Both generators were implemented in a way that the level of embedding and the vocabulary from which the words are drawn randomly can be configured. We designed it this way with the intent of later being able to mimic real-life messages where only a limited number of words are used usually. After writing the generator scripts we embedded the XML content in the CoAP message as payload, and generated 50,000 messages.

We translated then the CoAP messages to HTTP using our own translator, the operation of which can be observed in Figure 2. With yellow, the code of the message was translated to the corresponding method name in HTTP. For this, we looked up the CoAP and HTTP references and tried to find the best matching method names and codes. The parts highlighted with blue are the message type, the MID, and the token. These parameters exist only in CoAP therefore we put these in the first line of the HTTP payload. The options and corresponding headers are highlighted with grey. Here we tried to find the best matching header for each option but in some cases, there were no directly matching headers. For this reason, we made the header-option pairing configurable from file to fine-tune for the special case where our default choice might not work. The payload and body are highlighted with orange. Within the CoAP message, the XML payload was translated to JSON with the use of xmltodict [21] python library. To perform message pairs as training-, verification- and validation data sets, we translated every message in the CoAP database.



```
Header: 0.03 (T=RST, Code=0.03, MID=0x492a)
URI= coap://Dlh.org/barvvMmQrb
Token: 0x0
Content-Format: application/xml
Max-Age: "ysTCCCAzcR"
Proxy-Scheme: "tzMAGJ"
Accept: "wpX"
Payload:
<?xml version="1.0" encoding="UTF-8" ?>
<Insurance Companies>
<Top_Insurance Companies>
<Name>Berkshire Hathaway (BRK.A)</Name>
<Market_Capitalization>$539 billion</Market_Capitalization>
</Top_Insurance_Companies>
</Insurance Companies>
```

```
PUT/barvvMmQrb HTTP/1.1
Host: Dlh.org
Content-Type: application/xml
Cache-Control: "ysTcCCAzcR"
Proxy-Authorization: "tzMAGJ"
Accept: "wpX"
Payload: T=RST, MID=0x492a, Token=0x0
{
"Insurance Companies":{
"Top_Insurance_Companies":{
"Name": "Berkshire Hathaway (BRK.A)",
"Market_Capitalization": "$539 billion"
 }
}
}
```

Fig. 2: Mapping CoAP to HTTP

B. Model creation with the OpenNMT framework

After the investigation of the currently available, open-source ML frameworks for language translation, we selected the OpenNMT [18] solution to create models for the translation tasks. Building an OpenNMT model requires four input files: two for training and two for validation. Two files contain messages in the source language and two contain their equivalents on the target language. The paths of these files should be given in a YAML data serialization file with the number of input training records, the frequency of validation, and saving models. The major steps of a model-building procedure are the following:

1) Creating vocabulary from the training data,

2) Training the model,
3) Validation of the model.

The first step of teaching a model is to build a vocabulary from the training data sets, then the training and validation periods alternate multiple times. Finally, the model might be evaluated using an additional testing data set.

V. Translation between data formats

A. OpenNMT-based XML and JSON translation

As the first use case, we tested XML-JSON translation using the NMT method. In the first round models were taught to translate XML messages with one layer of embeddedness containing one key-value pair. All tag names and value pairs were generated from a discrete set of strings and numbers. This narrows down the possible messages that might be translated, but as industrial protocol standards also have strict rules of how they represent data, it does not make false simplifications about the method's usability.

Table II represents the results of the first use case. Train, Valid and Test columns represent the size of the data sets respectively. The Val:Train shows that after how many training iterations comes a validation step. Each training iteration consists of 50 records. Train Acc. and Valid Acc. columns abbreviate the training and validation accuracy scores.

Train	Valid	Test	Val:Train	Train Acc.	Valid Acc.
1000	500	200	1:10	64.63%	82.55%
2000	500	200	1:10	71.19%	95.01%
5000	1000	200	1:10	88.82%	99.99%
10000	1000	200	1:10	88.37%	100.00%

TABLE II: XML-JSON containing a single key-value pair

The results showed that the model is able to learn the structure in a relatively small training set, but required a 100-1000 times bigger set to minimize the error rate to below 0.01%. The error rate of OpenNMT training is calculated from the correctly translated tokens and the total number of tokens. Models in the second round were also taught to translate XML messages with one layer of embeddedness but containing multiple key-value pairs. Table III summarizes the results of the second use case.

Train	Valid	Test	Val:Train	Train Acc.	Valid Acc.
2000	2000	200	1:10	50.40%	53.09%
4000	2000	200	1:10	58.38%	54.55%
6000	2000	200	1:10	74.30%	68.08%
8000	2000	200	1:10	88.29%	74.97%
10000	2000	200	1:10	92.10%	77.22%
12000	2000	200	1:10	96.00%	79.07%
14000	2000	200	1:10	98.90%	80.65%
16000	2000	200	1:10	99.00%	81.15%
18000	2000	200	1:10	99.30%	81.77%
20000	2000	200	1:10	99.50%	82.41%

TABLE III: XML-JSON containing multiple key-value pairs

As a second use case, we trained and tested JSON-SenML translation models. SenML [22] is a predefined format for representing sensor measurement values. It could be a common language for providers who apply different schemas and provide different JSON payload formats. Table IV represents the accuracy of the trained models.

Train	Valid	Test	Val:Train	Train Acc.	Valid Acc.
2000	1000	100	1:10	81.95%	87.57%
4000	1000	100	1:10	84.75%	87.56%
6000	1000	100	1:10	85.68%	87.49 %

TABLE IV: JSON-SenML: Provider Message 1

B. Inspection results of OpenNMT

Investigating the XML-JSON translation results, the experience is that the longer messages grow, the more records are required to fine-tune a model. With messages containing only one key-value pair 10000 record was enough to minimize the error rate to below 0.01%, while with messages containing one to four key-value pairs the training data set containing 20000 records only scored 82.41% on validation accuracy.

Regarding the second use case, the provider schemes contained several numeric values and had a significantly wider value set than in the first use case. It turned out that OpenNMT is not suitable for translating texts that include several different numeric values. Listing 3 and Listing 4 present an incorrectly translated test result. This might be due to the following reasons. Firstly, OpenNMT creates a vocabulary from the training set, so it favors using numbers that had already been used in the training set. With significant intervals of numbers, it is highly unfavorable. Secondly, as shown in Table IV, the validation accuracy did not get better with more training steps, which can be due to pattern overlearning.

Listing 3: Expected SenML message

```
[
    {"u" : "C"},
    {"n" : "sensor1temp", "v" : 68},
    {"n" : "sensor2temp", "v" : 11.25},
    {"n" : "relay1state", "vb" : true}
]
```

Listing 4: Predicted SenML message

```
[
    {"u" : "C"},
    {"n" : "sensor1temp", "v" : 93},
    {"n" : "sensor2temp", "v" : 81.53},
    {"n" : "relay1state", "vb" : true}
]
```

To eliminate the above problems, teaching from larger data sets could be an effective way. If the results are not satisfactory, then using an ML-based learning model to translate the structure and then algorithmically inserting the numeric values might be the solution.

VI. ChatGPT-based XML to JSON translation

As a comparison to the initial results, we also tested the viability of ChatGPT [20] in protocol translation by translating XML to JSON. We applied the following prompt:

"Translate the following XML formatted texts to JSON. Only give the translation as an answer, nothing else."

We gave 100 XML formatted texts to ChatGPT for translation, then compared the results with the translations coming from the translator we used before. The XMLs were randomly generated using completely random strings and a maximum depth of 3. We found that disregarding whitespaces *89 out of the 100 translations were identical* to the translations made by our translator. In other words, 89% of the messages were properly translated. With approximately 850 key-value pairs out of which 11 were inaccurate, we can conclude that the accuracy for key-value pair translation is around 98.7%.

The 11 incorrect translations can be classified into 3 groups based on the type of mistake. In the first group, there were 4 incorrect translations, that were caused by leaving out escape characters as the strings in XML were put between parenthesis in JSON. In the second group, there were two translations where the name of the XML tag ended with a hyphen. The third group consists of 5 translations, where simply one character was left out from the translation.

Few shot prompting could be a possible solution to improve the previous results. Mistakes from the first and second groups could be written into the prompt as well as something important to look out for, to try to avoid them. Furthermore, all of the mistakes in these groups can be corrected by postprocessing as they are easily detectable.

VII. Summary

In this paper, we investigated an ML-based method to solve the M2M interoperability problems by text-based protocol translation. Since the widespread industrial IoT payload formats are XML and JSON, we focused on these during our work. We recognized that ML-based language processing and language translation models have improved a lot in recent years. This identification supports the introduced model to map the key-value pairs between different formats. As a proof-of-concept, we showed an OpenNMT-based approach. To measure the accuracy, we generated several data sets and defined two use cases. If we examine the results of each scenario, the accuracy of our model was 82% in the XML-JSON related case, and 87% in the JSON-SenML related case. We also tested ChatGPT for XML-JSON translation, and measured 98.7% accuracy regarding key-value pair translation, and 89% regarding the whole XML content.

We identified that the presented OpenNMT model has some limitations regarding the numeric values. Future work will further elaborate the translation of the numeric values between JSON and SenML, and the training mechanism of the language translation models to perform more accurate text-based protocol mapping tasks. Our further research regarding the use of ChatGPT in this field aims to refine the prompts to give and to test other kinds of translations using the ChatGPT API. In case of not being able to avoid predictable mistakes, postprocessing by a script could further improve the accuracy.

Acknowledgment

This research work has been funded by the European Commission, through the European H2020 research and innovation programme, KDT Joint Undertaking, and National Funding Authorities from 10 involved countries – including Hungary – under the research project Arrowhead fPVN with Grant Agreement no. 101111977.

References

[1] H. Derhamy, J. Eliasson, and J. Delsing, "Iot interoperability—on-demand and low latency transparent multiprotocol translator," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1754–1763, 2017.
[2] H. Derhamy, J. Eliasson, J. Delsing, P. P. Pereira, and P. Varga, "Translation error handling for multi-protocol soa systems," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, 2015, pp. 1–8.
[3] S. Maksuti, M. Zsilak, M. Tauber, and J. Delsing, "Security and autonomic management in system of systems," *Infocommunications Journal*, vol. 13, no. 3, pp. 66–75, 2021.
[4] P. Varga, F. Blomstedt, L. L. Ferreira, J. Eliasson, M. Johansson, J. Delsing, and I. Martínez de Soria, "Making system of systems interoperable – the core components of the arrowhead framework," *Journal of Network and Computer Applications*, vol. 81, pp. 85–95.
[5] C. Paniagua, "Service interface translation. an interoperability approach," *Applied Sciences*, vol. 11, no. 24, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/24/11643
[6] A. Renda, S. Schwaag Serger, D. Tataj, A. Morlet, D. Isaksson, F. Martins, M. Mir Roca, C. Hidalgo, A. Huang, S. Dixson-Declève, P. Balland, F. Bria, C. Charveriat, K. Dunlop, and E. Giovannini, "Industry 5.0, a transformative vision for Europe: governing systemic transformations towards a sustainable industry," *EC Directorate-General for Research and Innovation*, 2021.
[7] "W3C XML Specification," https://www.w3.org/TR/REC-xml/.
[8] "Internet draft of JSON scheme," https://json-schema.org/.
[9] "Converter from Code beautify," https://codebeautify.org/xmltojson.
[10] "convertJSON: XML to JSON and vice-versa converter," https://www.convertjson.com/json-to-xml.htm.
[11] F. Moutinho, L. Paiva, P. Maló, and L. Gomes, "Semantic annotation of data in schemas to support data translations," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016.
[12] F. Moutinho, L. Paiva, J. Köpke, and P. Maló, "Extended semantic annotations for generating translators in the arrowhead framework," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2760–2769, 2018.
[13] G. Amaro, F. Moutinho, R. Campos-Rebelo, J. Köpke, and P. Maló, "Json schemas with semantic annotations supporting data translation," *Applied Sciences*, vol. 11, no. 24, 2021.
[14] S. P. Khedkar and R. AroulCanessane, "Machine learning model for classification of iot network traffic," in *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2020, pp. 166–170.
[15] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.
[16] K. R. Dalal, "Analysing the role of supervised and unsupervised machine learning in iot," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2020, pp. 75–79.
[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ArXiv*, vol. 1409, 09 2014.
[18] "OpenNMT, An open source neural machine translation system." https://opennmt.net/.
[19] P. Tennage, A. Herath, M. Thilakarathne, P. Sandaruwan, and S. Ranathunga, "Transliteration and byte pair encoding to improve tamil to sinhala neural machine translation," in *2018 Moratuwa Engineering Research Conference (MERCon)*, 2018, pp. 390–395.
[20] OpenAI, "Chatgpt," https://chat.openai.com, 08 2023.
[21] "Python module that makes working with XML feel like you are working with JSON," https://pypi.org/project/xmltodict/.
[22] C. Jennings, Z. Shelby, J. Arkko, A. Keranen, and C. Bormann, *RFC 8428 – Sensor Measurement Lists (SenML)*, Std. IETF, 2018.