IoT

MDPI

Letter

Natural Language Understanding for Multi-Level Distributed Intelligent Virtual Sensors

Radu-Casian Mihailescu [1,2,*], Georgios Kyriakou [3] and Angelos Papangelis [4]

[1] Department of Computer Science, Malmö University, 20506 Malmö, Sweden
[2] Internet of Things and People Research Center, Malmö University, 20506 Malmö, Sweden
[3] Mirado Consulting, 11144 Stockholm, Sweden; georgios.kyriakou@mirado.com
[4] Axis Communications, 22369 Lund, Sweden; angelos.papangelis@axis.com
* Correspondence: radu.c.mihailescu@mau.se

Received: 29 October 2020; Accepted: 30 November 2020; Published: 6 December 2020

Abstract: In this paper we address the problem of automatic sensor composition for servicing human-interpretable high-level tasks. To this end, we introduce multi-level distributed intelligent virtual sensors (multi-level DIVS) as an overlay framework for a given mesh of physical and/or virtual sensors already deployed in the environment. The goal for multi-level DIVS is two-fold: (i) to provide a convenient way for the user to specify high-level sensing tasks; (ii) to construct the computational graph that provides the correct output given a specific sensing task. For (i) we resort to a conversational user interface, which is an intuitive and user-friendly manner in which the user can express the sensing problem, i.e., natural language queries, while for (ii) we propose a deep learning approach that establishes the correspondence between the natural language queries and their virtual sensor representation. Finally, we evaluate and demonstrate the feasibility of our approach in the context of a smart city setup.

Keywords: internet of things; virtual sensing; deep learning; natural language understanding

1. Introduction

Large-scale deployments of Internet of Things (IoT) sensing technologies and Big Data have opened up new avenues regarding the types of data that can now be collected and information that can be inferred [1]. Inherently, this raises the problem of effectively extracting valuable insights, as the volume of data continues to increase at an exponential pace.

In particular, smart cities are one of the core concepts that have emerged in an attempt to leverage massive streams of real-time data, obtained by installing and integrating various sensing technologies and connecting them through the internet. The term smart city refers to the idea conceptualised at the beginning of the 21st century, as the synergy between the technological and societal aspects of a city [2]. Essentially, it brings together a myriad of services and applications with the role of leveraging technology to augment quality of life, by means of addressing topics such as smart transport and mobility, smart management and governance, smart economy and smart grids, to name a few [3,4]. For a comprehensive and systematic review on smart cities we refer the reader to [5].

Especially in the context of complex settings, such as smart cities, it is often the case that the intended environment's state is not directly perceivable through individually dedicated physical sensors, e.g., estimating traffic levels in certain zones of the city. Thus, a number of various sensor types and values may need to be fused in order to determine the current state for a particular sensing task. A virtual sensor represents a logical entity that is performing a data transformation, which takes as input multiple sensor values and outputs the current state of the environment. In [6], the authors tackled the problem of integrating heterogeneous and distributed sensors in a dynamic manner,

thereby introducing the concept of dynamic intelligent virtual sensors (DIVS) in order to support the creation of services designed to address complex problems based on fusing various types of data. Moreover, they presented an application of DIVS in the context of activity monitoring using sensors in a smart building environment. In [7], the authors proposed a refined version of DIVS, which incorporates an active learning component, that enables the system to take input from both the user, in the form of feedback, as well as from the physical world through sensing. This input is further incorporated into the model via incremental learning. The mechanism proposed is analysed based on a publicly available room occupancy dataset. In the following, we extend the work in [6], introducing multi-level DIVS and demonstrating their applicability in the broader context of smart city environments.

The goal for multi-level DIVS is two-fold: (i) to provide a convenient way for the user to specify high-level sensing tasks; (ii) to construct the computational graph (we regard a computational graph as a graph where the nodes correspond to mathematical operations) that provides the correct output given a specific sensing task and a given mesh of (virtual) sensors. To address (i), we need to take a user-centric approach, whereby we enable the user to interact with the system in an intuitive and natural manner. To this end, we propose a goal-oriented natural language-based interface, that provides a user-friendly mechanism for defining complex sensing tasks for users, even for those lacking technical expertise. In this way, the user can pose queries to the system in spoken language, representing a sensing goal or task. Furthermore, in order to solve (ii), the system needs to include a natural language understanding component, which interprets the user's task and translates it into a sequence of computations, which ultimately provide the correct output for a given query, all while obscuring from the user the technicalities of the underlying sensing infrastructure being deployed in the environment. In short, the mechanism for multi-level DIVS can be understood as a configuration meta-layer, where the human user presents the system with a high-level task specified in natural language, which prompts the system to propose a new multi-level DIVS, constructed based on the pre-existing sensors and virtual sensors in the system. We use the term multi-level to denote the hierarchical nature of the computational graph, where a DIVS may use the output from other DIVS as input, as well as various different physical sensors. In Figure 1 we depict such an example where the output from $DIVS_2$ is further accessed by a particular service. Importantly, the user can inspect the DIVS computational graph in order to determine the validity of the solution, if such a solution exists.

The contribution of this paper is two-fold. First, we provide an automatic procedure to generate a questions/answers dataset, large enough to generalize the association between natural language queries (NLQs) and our proposed virtual sensor functional representation. Second, we apply a suitable machine learning approach that is able to learn the correspondence between NLQs and virtual sensor representations in a generalized manner.

The rest of the paper is organized as follows. Section 2 provides an overview of related work. Section 3 presents the concept of multi-level DIVS. In Section 4 we introduce the SmartCity-NLQ dataset, alongside the automatic generating procedure used in constructing this dataset. In Section 5 we provide experimental results and evaluate the feasibility of our approach. Finally, Section 7 concludes the paper.

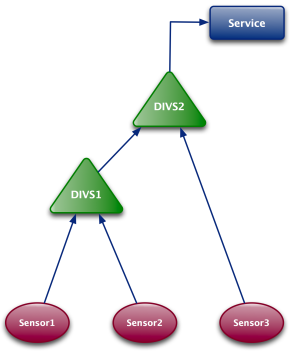


Figure 1. Example of multi-level DIVS.

2. Related Work

Human–machine technologies and user-centred design are playing and increasingly important role in the context of smart environments, where services are offered to end-users by different nodes in the system, in a ubiquitous and collaborative manner [8]. Goal-driven approaches and agent-based systems are typically employed to facilitate the interface with the end-user in a way that hides the complexities of configuring the smart environment to fulfil the user goal. This goes beyond the traditional wireless sensor network (WSN) setup [9], where the emphasis is exclusively on organizing the data gathering and routing protocols, given a network of sensors deployed in the environment.

In [10], the authors introduced the distributed goal-oriented computing architecture (DGOC) for smart-home environments, where users need to express their goals based on a predefined formal model. Then, a case-based reasoning approach is used for planning a certain course of action. Both the services offered by the system, as well as the user goals need to be semantically annotated a priori. Similarly, the work in [11] addresses the problem of goal-oriented end-user development for web of things devices. The authors use semantic web ontologies for knowledge representation and represent the task conveyed by the user as a planning problem. Alternatively, the problem of multiple cooperating services is also surfacing in the context of smart factories, where the reconfiguration of manufacturing environments is gaining increased importance. For example, dynamic orchestration of semantic web services is used in [12] to solve the reconfiguration task at runtime. A human-in-the-loop solution for automatic production line assembly is presented in [13], with the aim of seamlessly including and assisting human operators. In [14], the authors propose a visual programming tool that allows the user to specify the desired state of a smart environment graphically. Semantic API descriptions are used to perform automatic service composition, in order to satisfy the user goal. The solution employs first-order logic to compose services. Agent-based approaches [15], as well as role-based models [16], have also been proposed for smart environments, where a dynamic set of devices, with their functionalities and services, cooperates temporarily to achieve a user goal.

At the same time, in the private sector, intelligent personal assistants for smart homes, using voice commands, such as Alexa, Siri, Cortana, and Google Home are becoming more popular in the market. Voice-based control is particularly appealing, due to the fact that it enables users to anthropomorphize technology and interact with it in a natural way, similar to that of communicating with another person. Moreover, it allows the user to do so without the need of interrupting current activities, such as for instance would be required in the case of interacting with a visual tool. Automatic speech recognition (ASR) [17] and natural language processing (NLP) techniques [18] are employed to enable the user interaction based on voice control. Deep learning (DL) approaches have consistently replaced state-of-the-art language models over a wide range of applications including machine translation, speech recognition, sentiment analysis, text summarization, question answering or conversational agents (e.g., [19–21]). However, most IoT solutions (such as IFTTT (https://ifttt.com/) —"If this, then that"—which is web-based service that creates chains of simple conditional statements) are limited to support strict commands, which can interact with only one device at a time.

In contrast to the abovementioned works, in this paper we design a system to tackle the problem of complex sensing tasks in a ubiquitous smart city setup, based on expressing the high-level goal of the user by means of natural language queries. Importantly, we aim to avoid strict language patterns, or being confined to a small set of fixed predefined commands. Instead, we propose a flexible and versatile approach design to cope with numerous grammatical forms for specifying a wide variation of goals.

3. The Multi-Level DIVS Model

Essentially, the goal of multi-level DIVS, for a given sensing task expressed in natural language, is to perform compositional reasoning based on the set of pre-existing physical and/or virtual sensors and basic operations. For instance, suppose the user formulates the following query: "Give me the number of cars that are parked illegally at Malmö University". In order for the model to output the correct answer, a sequence of computational transformations need to be applied on the video feed obtained from cameras surveying the university's parking lot. In our model, the sequence of operations to be performed for a specific input question is represented through a computational graph. In Figure 2b we depict the graph corresponding to the abovementioned query. Each node in the graph contains one of the predefined functions in the system. Functions can either implement basic operations (e.g., arithmetic operators) or correspond to the specific role played by an existing virtual sensor (e.g., car recognition on image data). Without loss of generality, and for the sake of simplicity, in the following we are using functions of arity maximum 2. This visual representation is designed to provide a straightforward interpretable description of the multi-level DIVS for the end-user. In order to derive the sequence of operations to be applied on the incoming sensor data, we further provide a functional representation, which boils down to a preorder traversal of the computational graph, resulting in a prefix notation. Below we provide as a simple example the functional representation corresponding to the graph depicted in Figure 2a:

```
filter_Location[Malmö University] filter_sensor[temperature] fork
filter_Location[Malmö Arena] filter_sensor[temperature]  union
compare_lower
```

More formally, the multi-level DIVS model takes as input a query $\mathbf{w} =< w_1, \ldots, w_N >$, which is a sequence of words w_i, and predicts a functional representation $\mathbf{s} =< s_1, \ldots, s_M >$, which specifies a sequence of functions $s_i \in \mathcal{F}$, where $\mathcal{F}$ denotes the predefined set of basic functions in the system. The input and output sequences can be of variable, possibly different, lengths. The model uses a sequence-to-sequence architecture, producing first an encoded intermediate-level feature map $\mathbf{h} =< h_1, \ldots, h_T >$. Then, the model estimates the conditional probability of generating s_i at time step i, given the input query $\mathbf{w}$ and the previously generated sequence $s_{<i}$ as:

$$p(s_i|s_{<i}, \mathbf{w}) = g(s_{i-1}, q_i, h_T), \tag{1}$$

where the function g is given by a softmax over all functions in $\mathcal{F}$, $s_{<i} = (s_1, ..., s_{i-1})$, and q_i is the hidden state of decoder at time step i:

$$q_i = f(q_{i-1}, s_{i-1}, h_T) \tag{2}$$

In short, the model predicts the sequence of computational steps required in order to implement the desired sensing task expressed in the query. We implement the sequence-to-sequence model using Gated Recurrent Units (GRUs), introduced by Kyunghyun Cho et al. in [22], meaning that the choice for the recurrent component f in Equation (2) is a GRU. The reason we have choosen GRUs instead of long-short term memory networks (LSTMs) or other recurrent neural network (RNN) variants is due to its computational efficiency when it comes to smaller sized datasets [22].

In addition, we append an attention mechanism which has the role identifying which words from the query are most relevant for predicting the next function in our functional representation output, by assigning high attention weights to those words. In other words, this is achieved by computing a so-called attentive context vector c_j by weighting the encoded representations $\mathcal{H} =< h_1, \ldots, h_T >$ with their corresponding attention weights:

$$c_j = \sum_i \alpha_{ji} h_i \tag{3}$$

Finally, the attentive context vector c_j and the hidden q_{i-1} state of the decoder from the last time step are used to generate the output s_j, by replacing h_T with c_j in Equations (1) and (2).

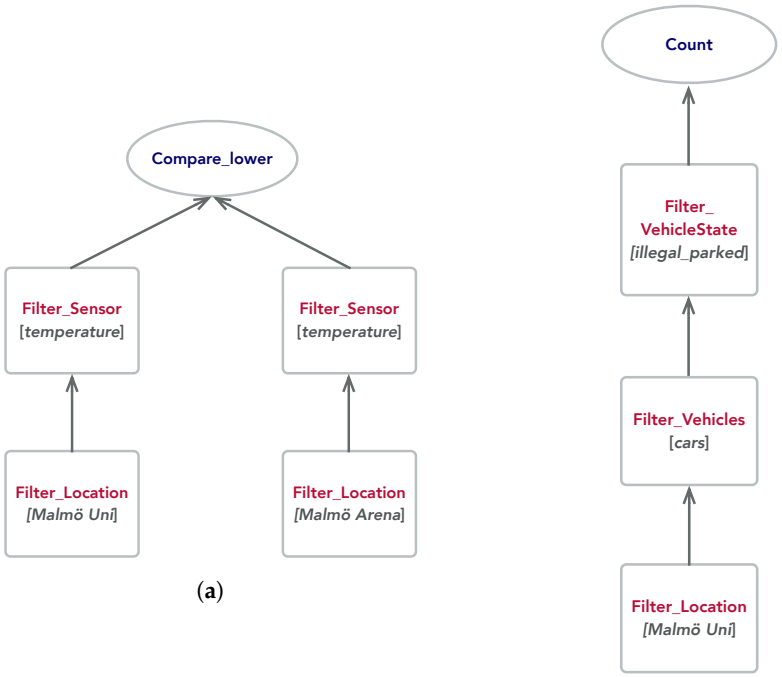


Figure 2. Computational graphs corresponding to the following natural language queries (**a**) Is it colder at Malmö University compared to Malmö Arena? (**b**) Give me the number of cars that are parked illegally at Malmö University.

4. The SmartCity-NLQ Dataset

In the following, we introduce the SmartCity-NLQ dataset (https://github.com/Angel0r/SmartCity-NLQ-Dataset), which has been designed for the purpose of evaluating and learning to generalize the association between natural language queries (NLQs) and our proposed virtual sensor functional representation, with application in the smart city domain. In order to construct the SmartCity-NLQ dataset, we apply a similar template-based question generator approach to that of Johnson et al. [23]. However, in contrast to their dataset construction approach, we are not concerned here with manipulating image data and instead operate over the multi-level DIVS computational graph.

4.1. Dataset Structure

Our dataset supports four different categories of questions, namely:

- Comparison: ”Is this larger than that”?
- Existence: ”Are there any...?”
- Counting: ”How many...?”
- Attribute: ”What is the length...?”

Furthermore, note that it is mandatory that the question specifies the physical location being sensed, which enables the system to select the appropriate set of sensors for answering it. Specifically, the semantics used in generating the queries are based on the following set of constructs:

4.1.1. Operators

- Comparison operator. Returns the boolean results using the arithmetic comparison operation of two integers.
- Set operator. Returns a set by performing the union or the intersection operations on the two input sets.

4.1.2. Functions

- Filter function. These functions encapsulate the functionality provided by the sensors or the virtual sensors available in the system. In addition, the filter_location function has the role of returning the set of sensors designated to that specific location. For example, the sequence filter_location[Folkets Park], filter_sensor[temperature] would first return the set of sensors in Folkets Park and subsequently, would return the value of the temperature sensor.
- Count function. Returns the number of items in a set.
- Exist function. Returns a boolean depending if the set is empty or not.
- Attribute function. Returns the specified attribute of the input item.

Hence, the functional representation for multi-level DIVS consists of chains of functions in combination with the aforementioned arithmetic and logical operators. The lowest level function is typically a filter_location function that takes as parameter the location name from the query and returns the set of available sensors. Once the specific sensor is selected based on the filter_sensor function, the incoming sensor data (e.g., temperature value, video stream from camera, CO_2 level, etc.) is provided implicitly as input to the next function in the sequence. The functions and operators perform computations on sets of items, or integer values representing item attributes.

4.2. Dataset Generation

Building upon the work of [23], we construct templates, represented as JSON objects, which have the role of automatically generating a large number of natural language questions for each of the abovementioned categories. Each template object consists of several text templates, representing multiple ways of expressing a certain functional representation in natural language, parameters, nodes and functional maps. In addition, metadata, designating a group of words which fall under a specific categorical description, is used to instantiate parameters within the text templates, while nodes and functional maps are used for generating the actual functional representation. The procedure entails replacing the placeholders in the text templates with actual words. The number of permutations generated is linked to the number of placeholders in the sentence and the number of words available to replace them in the metadata. The process is repeated for all questions, for all templates. Therefore, with just a few parameters per template and a small number of question categories we can generate a large set of unique questions. Synonyms are further used to increase the language diversity. Furthermore, it is worth pointing out that based on this procedure, the dataset can be easily further expanded. In Appendix A of the paper we provide an example that explains in more detail the implementation of the template-based approach. All of the templates follow the same structure. In total, the dataset consists of 48,620 questions, with a corresponding total number of 84 functional representations, realized with an average number of 21 text templates and an average of 2.85 parameters per question.

5. Experimental Results

We evaluate the performance of our proposed multi-level DIVS model on the SmartCity-NLQ dataset. The test set consists of ten thousand examples randomly sampled, while the remainder of the

dataset is used for training. During training we implement teacher forcing, which means using the real target outputs as each next input in the sequence, instead of using the model's prediction as the next input. Given the functional representation introduced in Section 3, for evaluation, we investigate the word error rate (WER) metric, which computes the difference between the model's generated sequence answer and the template-generated answer, i.e., the ground truth:

$$WER = \frac{S + D + I}{N} \tag{4}$$

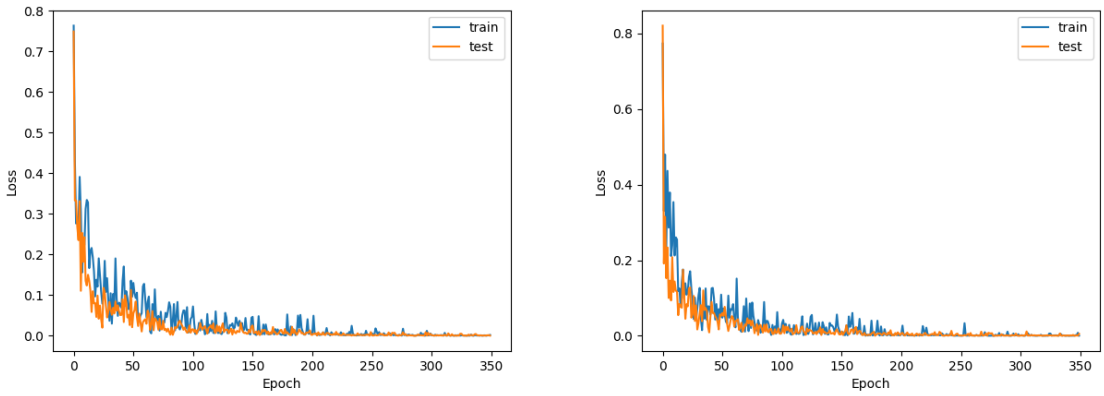
where S is the number of substitutions, D is the number of deletions, I is the number of insertions, C is the number of correct words, and N is the number of words in the reference ($N = S + D + C$). Essentially, WER computes the number of word insertions, substitutions and deletions necessary to convert the hypothesis (or answer) into the reference (or ground truth), representing the Levenshtein distance [24], which is then divided by the number of words of the reference sequence. Note that in Equation (4) substitutions, insertions and deletions are weighted equally. Thus, lower values of the WER metric correspond to fewer deviations from the ground truth.

In Table 1 we report results obtained when constructing multi-level DIVS based on natural language queries. The proposed model attains a high accuracy above 99%, as well as a low WER score, which is indicative of very good performance. At the same time, notably, extending the model to include the attention mechanism, previously introduced in Section 3, has been shown to improve performance both in terms of accuracy and WER.

Essentially, the attention mechanism allows the model to focus on different parts of the input sequence at every stage of the output sequence, such that context can be preserved from beginning to end. In Figure 3 we plot the model loss, defined as cross-entropy loss, in the number of epochs (an epoch is when the entire training set is passed forward and backward through the neural network exactly once) for the following set of optimised hyperparameters: number of hidden units 256, batch size 100 and teacher forcing 0.5. In both instances, with and without the attention mechanism, the models converge quickly after about 150 epochs, however, noticeably, the slope of the loss curve is steeper for the attention mechanism, denoting a speed-up during training.

Table 1. Model performance.

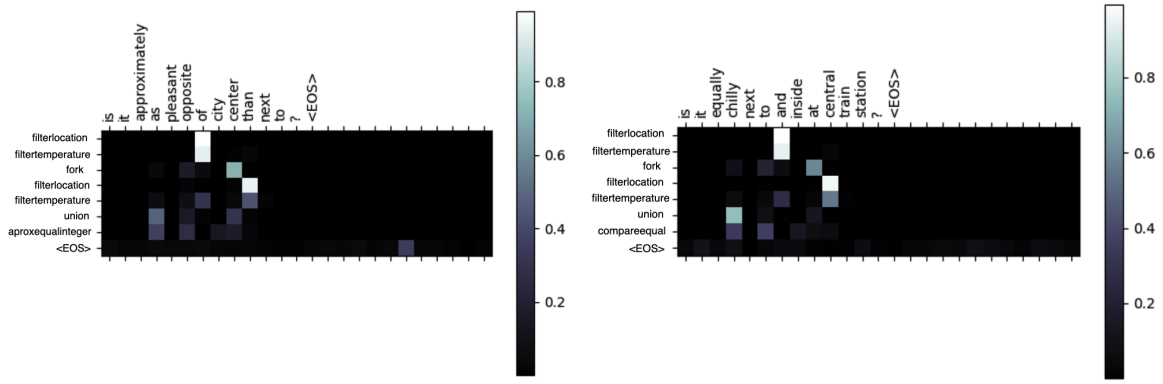
Model	WER	Wrong answers	Accuracy
Multi-level DIVS	5.25	16	99.66%
Multi-level DIVS with Attention	3	10	99.85%



(**a**) (**b**)

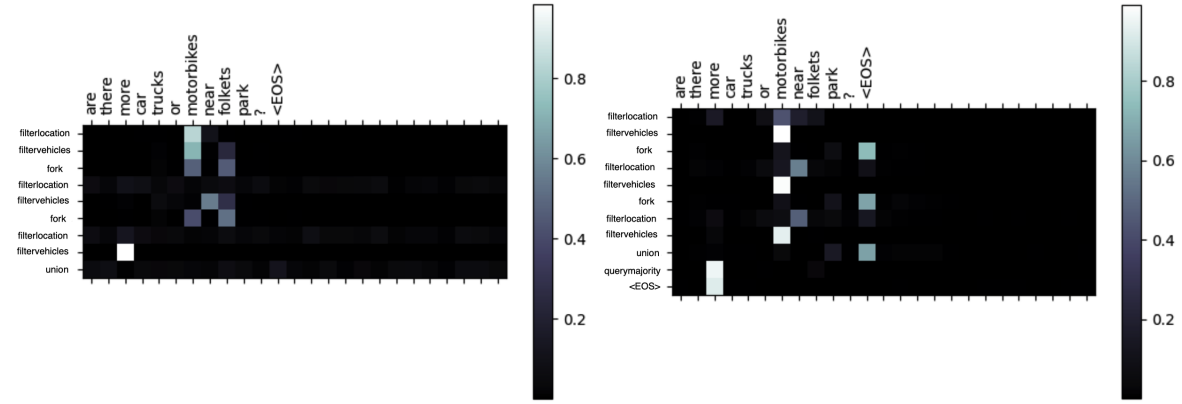
Figure 3. Model loss (**a**) without Attention mechanism; (**b**) with Attention mechanism.

Additionally, in Figure 4 we provide a heatmap representation showing the attention output as a matrix for several question and answer examples. The columns illustrate the sequence of words in the question, while the rows illustrate the sequence of functions, corresponding to the functional representation of the answer. We depict the heatmap using a black and white scale. A brighter coloration of the square in the heatmap signifies that the model paid more attention to that specific word. That is to say, which words from the query are the most relevant to each function from the generated sequence. In this way, we have a better understanding of where the network is focused most when producing the output. It is worth noting that, for the two examples in Figure 4, with similar functional representations, we can observe similarities in the distribution of attention over the query sequences, which are also of similar form. Given the convergence of the training loss for the attention network plotted in Figure 3b, we depict in Figure 5 the evolution of the attention and network output for different numbers of epochs. Noticeably, in Figure 5a the function sequence is missing the comparison operator, while the heatmap representation appears difficult to interpret. In contrast, after 350 epochs, in Figure 5b the model provides the correct functional representation for the given query. Moreover, the heatmap displays a higher level of interpretability, with several clear correspondences between functions and natural words, such as is the case for the comparison operator and the filter_vehicle function.



(**a**) (**b**)

Figure 4. Heatmap representation example for two questions depicted in (**a**) and (**b**) belonging to the comparison class.



(**a**) 40 Epochs (**b**) 350 Epochs

Figure 5. Heatmap representation illustrating the attention output for the same question after training for different number of epochs.

6. Discussion

For a more in-depth analysis, we provide in Table 2 an evaluation based on breaking down the results into the four categories of questions comprised in the SmartCity-NLQ dataset. Additionally, we account for the number of words that compose each questions and make the distinction between long and short questions. Namely, we classify short questions as those with a word count lower or equal to 12 and long questions otherwise, the span of words varying between 6 and 24. Note that the dataset consists of a balanced mix of long and short questions, representing 56.4% and 43.6%, respectively. Interestingly, the model performs very well in the case of the comparison and attribute classes of question, for both with and without the attention mechanism. However, as it pertains to the existence and counting classes, the model returns a number of errors, specifically in the case of short questions. We can observe as well, that the attention mechanism is responsible for an approximate 5% increase in accuracy for these particular instances.

Table 2. Model performance for the different classes of questions.

Type of Query	**Comparison**		**Existence**		**Counting**		**Attribute**	
Query size	Short	Long	Short	Long	Short	Long	Short	Long
WER (no Att.)	0	0	4.25	0	1	0	0	0
Accuracy (no Att.)	100%	100%	92.4%	100%	92.4%	100%	100%	100%
WER (with Att.)	0	0	2	0	1	0	0	0
Accuracy (with Att.)	100%	100%	97.3%	100%	97.3%	100%	100%	100%

One of the key directions for future work includes applying transfer learning techniques for natural language processing in order to determine the extent to which our approach can work with limited label data, as well as to increase the language diversity, by enabling the system multiple ways of expressing various sensing tasks through natural language. Specifically for this latter task, one possibility we plan to explore in future work is to further evaluate our system based on real human utterances collected through crowd-sourcing.

7. Conclusions

In this paper we have introduced multi-level distributed intelligent virtual sensors as an extension of the work in [6], focusing on the one hand, on a convenient way for the user to specify high-level sensing tasks, and on the other hand, on a machine learning approach to transform natural language queries into computational graphs that can implement the aforementioned sensing tasks. We have shown that our model can translate efficiently novel generated questions to functional representation answers. Furthermore, we have introduced in this work the SmartCity-NLQ dataset, which consists of question and answer tuples, based on a mesh of sensor inputs, which describe an environment of a Smart City. Notably, the dataset is easily expandable given the template generation procedure presented in this paper.

Author Contributions: Conceptualization, R.-C.M., G.K. and A.P.; Methodology, R.-C.M., G.K. and A.P.; Software, G.K., A.P and R.-C.M.; Validation, G.K., A.P and R.-C.M.; Formal analysis, R.-C.M.; Investigation, R.-C.M., G.K. and A.P.; Data curation, G.K. and A.P.; Writing—original draft preparation, R.-C.M.; writing—review and editing, R.-C.M., G.K. and A.P.; Supervision, R.-C.M.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Stiftelsen för Kunskaps- och Kompetensutveckling grant number 20140035 and by Crafoord project grant number 20200953.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In Figure A1 we provide an example for the structuring of a template object.

The questions section in Figure A1 specifies three different ways for expressing the same query and embeds three placeholders within each formulation, which are related to the parameters section of the template object. The metadata information is used to instantiate the placeholders based on the parameters they denote. Specifically, let us consider that for Location we are going to use the following phrases Malmö university, Malmö stadium and Folkets park; while for Vehicles we are going to use motorcycles and cars; and finally for VehicleState we want parked or driving as possible vehicle states. In addition, the nodes and functions map are used to construct the functional representation. Figure A2 gives the output which lists some of the generated questions. Below we also include the functional representation corresponding to the first question from the list (also depicted in Figure 2b). Note that for all the remaining questions only the parameters of the functional representation are going to change, while the functions sequence remains unchanged.

```
{
  "params": [
    { "type": "Location", "name": "<L>" },
    { "type": "Vehicles", "name": "<V>" },
    { "type": "VehicleState", "name": "<VS>" }
  ],
  "questions": [
    {
      "text":
      "How many <V> are illegally <VS> at <L>?"
    },
    {
      "text":
      "Are there <V> are illegally <VS> at <L>?If yes, how many?"
    },
    {
      "text":
      "Give me the number of <V> that are <VS> illegally at <L>?"
    }
  ],
  "nodes": [
    { "inputs": [], "type": "scene" },
    {
      "side_inputs": ["<L>", "<V>", "<VS>"],
      "inputs": [0]
    },
    { "inputs": [0, 1], "type": "count" }
  ],
  "functions_map": [
    { "function": "filterVehicles", "name": "<V>" },
    { "function": "filterLocation", "name": "<L>" },
    { "function": "filterVehicleState", "name": "<VS>" }
  ]
}
```

Figure A1. Example of template object.

```
"Give me the number of cars that are parked
illegally at malmö university?"
"Give me the number of motorcycles that are parked
illegally at malmö stadium?"
"Give me the number of trucks that are parked
illegally at folkets park?"
"Give me the number of trucks that are driving
illegally at malmö stadium?"
...
```

Figure A2. Set of questions.

```
filter_Location[Malmö University] filter_Vehicles[cars]
filter_VehicleState[parked illegaly] count
```

References

1. Mahdavinejad, M.S.; Rezvan, M.; Barekatain, M.; Adibi, P.; Barnaghi, P.; Sheth, A.P. Machine learning for internet of things data analysis: A survey. *Digit. Commun. Netw.* **2018**, *4*, 161– 175. [CrossRef]
2. Mardacany, E. Smart cities characteristics: Importance of buit environments components. In Proceedings of the IET Conference on Future Intelligent Cities, London, UK, 4–5 December 2014; pp. 1–6.
3. Camero, A.; Alba, E. Smart City and information technology: A review. *Cities* **2019**, *93*, 84–94. [CrossRef]
4. Hoon Kim, T.; Ramos, C.; Mohammed, S. Smart City and IoT. *Future Gener. Comput. Syst.* **2017**, *76*, 159–162.
5. Yigitcanlar, T.; Kamruzzaman, M.; Buys, L.; Ioppolo, G.; Sabatini-Marques, J.; da Costa, E.M.; Yun, J.J. Understanding 'smart cities': Intertwining development drivers with desired outcomes in a multidimensional framework. *Cities* **2018**, *81*, 145–160. [CrossRef]
6. Mihailescu, R.C.; Persson, J.; Davidsson, P.; Eklund, U. Towards Collaborative Sensing using Dynamic Intelligent Virtual Sensors. In *Intelligent Distributed Computing*; Badica, C., El Fallah Seghrouchni, A., Beynier, A., Camacho, D., Herpson, C., Hindriks, K., Novais, P., Eds; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 217–226.
7. Tegen, A.; Davidsson, P.; Mihailescu, R.; Persson, J.A. Collaborative Sensing with Interactive Learning using Dynamic Intelligent Virtual Sensors. *Sensors* **2019**, *19*, 477. [CrossRef] [PubMed]
8. Reddy, Y.V. Pervasive Computing: Implications, Opportunities and Challenges for the Society. In Proceedings of the 2006 First International Symposium on Pervasive Computing and Applications, Pisa, Italy, 13–17 March 2006; p. 5.
9. Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422. [CrossRef]
10. Palanca, J.; Val, E.d.; Garcia-Fornes, A.; Billhardt, H.; Corchado, J.M.; Julián, V. Designing a goal-oriented smart-home environment. *Inf. Syst. Front.* **2018**, *20*, 125–142. [CrossRef]
11. Lastra, J.L.M.; Delamer, M. Semantic web services in factory automation: Fundamental insights and research roadmap. *IEEE Trans. Ind. Inform.* **2006**, *2*, 1–11. [CrossRef]
12. Feldmann, S.; Loskyll, M.; Rösch, S.; Schlick, J.; Zühlke, D.; Vogel-Heuser, B. Increasing agility in engineering and runtime of automated manufacturing systems. In Proceedings of the 2013 IEEE International Conference on Industrial Technology (ICIT), Cape Town, South Africa, 25–28 February 2013; pp. 1303–1308. [CrossRef]
13. Ferrer, B.R.; Mohammed, W.M.; Lobov, A.; Galera, A.M.; Lastra, J.L.M. Including human tasks as semantic resources in manufacturing ontology models. In Proceedings of the IECON 2017—43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, China, 29 October–1 November 2017; pp. 3466–3473. [CrossRef]
14. Mayer, S.; Verborgh, R.; Kovatsch, M.; Mattern, F. Smart Configuration of Smart Environments. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1247–1255. [CrossRef]

15. Alkhabbas, F.; Ayyad, M.; Mihailescu, R.; Davidsson, P. A Commitment-Based Approach to Realize Emergent Configurations in the Internet of Things. In Proceedings of the IEEE International Conference on Software Architecture Workshops, ICSA Workshops, Gothenburg, Sweden, 5–7 April 2017; pp. 88–91.
16. Mihailescu, R.; Spalazzese, R.; Heyer, C.; Davidsson, P. A Role-Based Approach for Orchestrating Emergent Configurations in the Internet of Things. In Proceedings of the 2nd International Workshop on Internet of Agents (AAMAS'17), Sao Paulo, Brazil, 8–12 May 2017; pp. 18–36.
17. Wang, D.; Wang, X.; Lv, S. An Overview of End-to-End Automatic Speech Recognition. *Symmetry* **2019**, *11*, 1018. [CrossRef]
18. Jain, H.; Mathur, K. Natural Language Processing Through Different Classes of Machine Learning. In Proceedings of the Fourth International Conference on Computer Science & Information Technology, Sydney, Australia, 21–22 February 2014; Volume 4, pp. 307–315.
19. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. *arXiv* **2020**, arXiv:2005.14165.
20. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692
21. Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; Catanzaro, B. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv* **2019**, arXiv:1909.08053.
22. Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, 25–29 October 2014; A meeting of SIGDAT, a Special Interest Group of the ACL; pp. 1724–1734.
23. Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C.L.; Girshick, R. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1988–1997.
24. Miller, F.P.; Vandome, A.F.; McBrewster, J. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau Levenshtein Distance, Spell Checker, Hamming Distance*; Alpha Press: Orlando, FL, USA, 2009. Available online: https://dl.acm.org/doi/book/10.5555/1822502 (accessed on 4 November 2020).