Evaluating the Effect of Semantic Enrichment on Entity Embeddings of IoT Knowledge Graphs

A. NONYMIZED[1]

This paper has been anonymized for the purpose of anonymous reviewing

Abstract. IoT devices in Smart homes generate large amounts of data by sending messages to share measurements at frequent intervals. In order to make this IoT measurement data interoperable, ontologies such as SAREF are used to represent this data as knowledge graphs (KGs). These resulting KGs contain measurements from IoT devices together with the context about those devices, such as type of measurements, or in which room it is located. Through the use of embedding methods we learn embedding representations for entities in KGs that can be used to generate prediction models. In this study, we investigate how the structure of the IoT knowledge graph affects the effectiveness of embedding methods for such tasks. Specifically, we look at the effect on embedding quality of enriching IoT KGs by making implicit information such as temporal sequences and measurement value similarity explicit. We perform experiments on two IoT KGs represented using the SAREF ontology. One is in a basic setting, and the other has this information explicitly added. We use RDF2vec to create embeddings, train a classifier, and compare classifier accuracy.
Analysis of our results show that classifiers trained with embeddings from enriched KGs outperform classifiers trained with embeddings from original KGs. This indicates that the explicit information in the enriched graph improves the quality of the embeddings of entities.

1 Introduction

With IoT devices becoming more prevalent in every day homes, common frameworks in which IoT devices can interact become more important. [1] Some of these frameworks use ontologies to create a common knowledge representation which can represent all available information that IoT devices can share. The resulting knowledge graph (KG) contains not only the information from the IoT devices, but also, due to the organisation of the ontology, the context of the information.

All this information combined in one KG provides opportunities to learn over the shared knowledge by using embedding methods that create vector representations of the entities in a KG, which can be used to train ML models used as forecasters or classifiers. A classifier could, for example, be used to predict whether the outside temperature is expected to be warm or cold, which in turn can be used to turn a heater on or off to save energy.

However, the ontologies that are used in these frameworks were instead developed in order to optimise embedding methods to learn from them, they were developed in a manner that: "facilitates the matching of existing assets in the smart applications domain". [3]. Explicitly adding implicitly available information could potentially improve the quality of entity representations that are created with the embedding methods. In this research we investigated the effect that semantic enrichment of a KG has on the quality of entity embeddings learned from it. This was tested by taking existing IoT KGs and creating new IoT KGs by semantically enriching them, through the addition of new properties and entities for each measurement. Classifiers are trained with the respective entity embeddings and the accuracy of these classifiers was used to determine the quality of the entity embeddings.

Before the experiment is described in more detail, the next section will give an overview of relevant research and use it to define some concepts we used in this work.

2 Background

In this section we will give a short overview of existing work that is relevant to our research. The first part provides an explanation of what we define as IoT graphs, while the second part provides background to embedding methods and specific methods that will be used in our experiments.

2.1 IoT graphs

We define IoT graphs as KGs specifically created to represent measurement data from IoT devices. SAREF [3] was created to enable interoperability between IoT devices, it was designed to be able to represent any information coming from IoT devices, serving as a common "language" to share information with any other device. In Figure 1, we show an example of a SAREF graph, four of the six properties of a `saref:Measurement` entity[1] connect to an entity that connects to every measurement made by that device, connecting to the same `saref:Device` `saref:FeatureOfInterest`, `saref:Property` and `UnitOfMeasurement` entities, with the other two properties relating to literals. So when walking through this graph, every measurement is reachable within two steps.

As seen in Figure 1, there are a few entities, such as `saref:Device` or `saref:Property`, that are connected to the much larger amount of measurement entities. This disproportionate imbalance between many measurement entities, and a few other entities is what we consider to define a IoT KG. As stated in [9], the New York entity in DBpedia is specifically highly connected because from it half of all other entities can be reached within two steps. This high connectivity

[1] In order to be consistent in our terminology we keep referring to these as entities, but for RDF purposes these are resources.

is also characteristic for IoT graphs, but instead of it being a specific characteristic of one entity, this holds for every entity in the graph. Section 3.4 describes how the datasets we used follow this structure.

When examining the Web of Things ontology a similar structure can be observed, with the `wot:property` entity acting as the `saref:Measurement` entity. [12] Similarly the Semantic Sensor Network ontology uses `ssn:observation`. [2] Earlier research by Moreira et al. [6] has shown that, with minimal mappings, most properties of IoT data can be represented by both SSN and SAREF. In our experiments we focus on KGs that were modelled with SAREF, but based on similarities between all discussed ontologies we expect our results to also be representative for KGs modelled with the other ontologies.

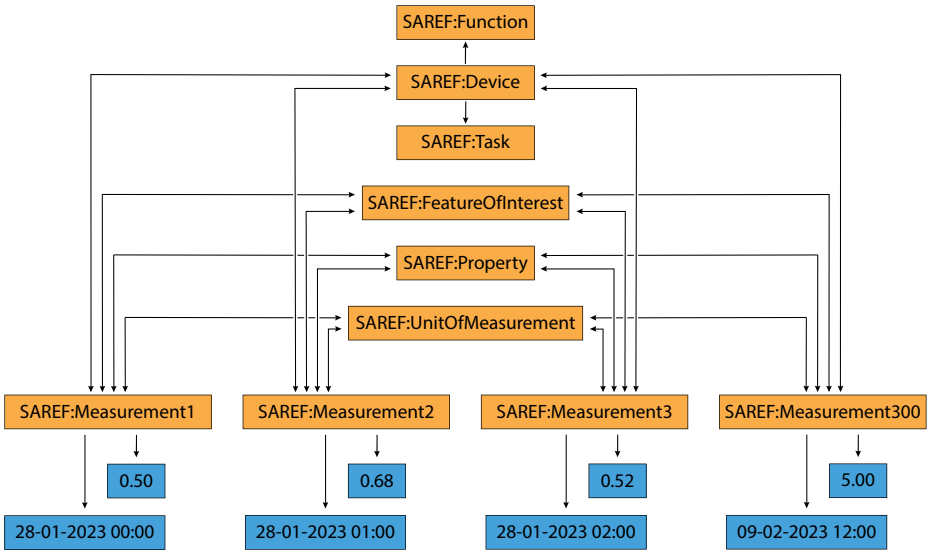

Fig. 1: Subset of four measurements from a SAREF IoT graph, orange boxes are entities, blue boxes are literals. Arrow heads represent the direction of the properties.

2.2 Embedding algorithms

The purpose of embedding models is to create a numerical representation for specific entities. Word2vec learns embeddings for words based on with which other words they co-occur in sentences. [5] RDF2vec uses random or directed walks to mimic sentences, working under the assumption that good representations for nodes can be learned based on with which other nodes they co-occur in random walks. [10] Research has shown that this assumption holds for many graphs, such as DBpedia or Wikidata.

In [9] the authors describe multiple variants of RDF2vec methods, which are all evaluated on different KGs. These graphs were generated based on specific characteristics that a KG can have, such as cardinality restrictions or relations to particular individuals. Similar to their work we research the effect of KG characteristics on the quality of RDF2vec embeddings. However our characteristics are based on graphs that are used in practice, instead of very specific logical definitions.

Adding new information to a graph based on implicit information can have an adverse effect, as shown by [4]. In their paper, experiments are performed to test the effect of additional implicit information, but the quality of the embeddings actually declines. They hypothesise that the initial absence of the implicit information was in it self a signal. Our semantic enrichment approach similarly utilises implicit knowledge, but in the case of IoT KGs the implicit information is used to create new relations, because the current relations are insufficiently connecting relevant entities.

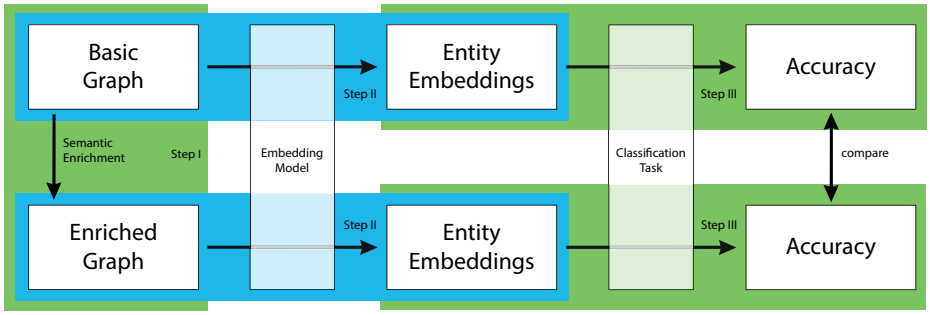

Fig. 2: Experimental pipeline.

3 Method

In order to expose the effect that semantically enriching IoT graphs has on entity embeddings, the following experiments were created. The experimental pipeline is depicted in Figure 2. Each experiment consists of two graphs: an IoT graph (*basic graph*), and a copy of that first IoT graph that has been semantically enriched (*enriched graph*). For the second step an embedding method is used to create embeddings for specific entities in each graph, and step three uses these entity embeddings to train a classifier. Comparing the accuracy of the classifier trained with embeddings from the basic graph, with the accuracy of the classifier trained with embeddings from the enriched graph will show the effect of the semantic enrichment.

Each step is described in more detail in the following subsections. All the code used in these experiments is available in our github repository.[2]

To investigate whether the amount of devices affects the embeddings, the experiments were performed with three different quantities of devices. We used either all the measurements of: one device, all the devices from one home or all the devices from all the homes in the dataset.

3.1 Step I: Semantic Enrichment

In order to perform the semantic enrichment of the graph we created three new entities based on implicit knowledge in the graph.

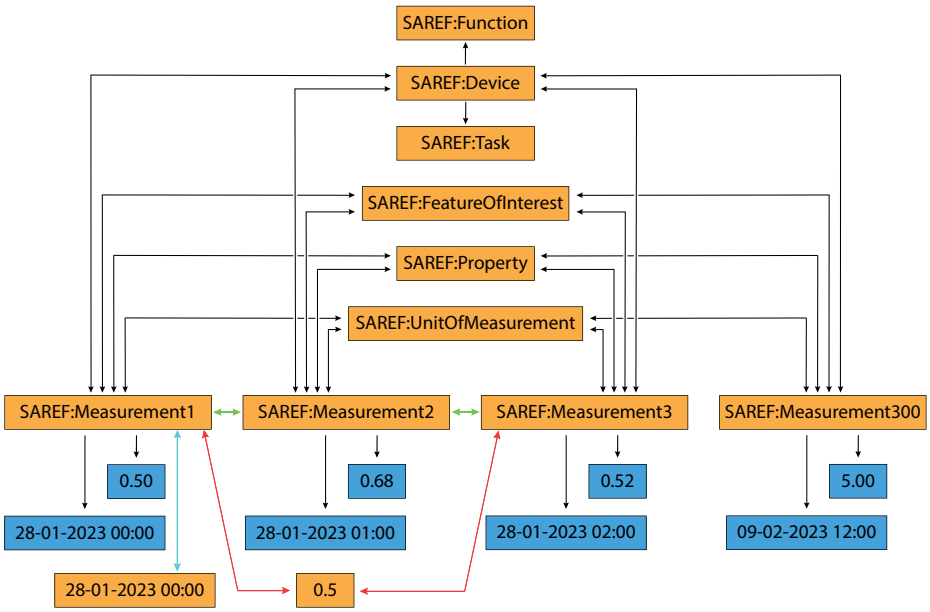

Fig. 3: Updated visualization of Figure 1 with the semantic enrichment.

Rounded value (red line in Figure 3) Each measurement entity has a measurement value, which is used to create a new entity based on the rounded value of the measurement. In this research the value was rounded to one decimal.

```
ex:measurement1 ex:hasRoundedValue ex:0.5
ex:0.5 ex:hasRoundedValue ex:measurement1
```

Sequence links (green line in Figure 3) For each measurement entity a property is added towards the "next" and "previous" measurement entities, based on the chronological order of the measurements.

[2] https://anonymous.4open.science/r/semantic-enrichment-of-IoT-graphs-A46E

```
ex:measurement1 ex:nextMeasurement ex:measurement2
ex:measurement2 ex:previousMeasurement ex:measurement1
```

Timestamp (blue line in Figure 3) The literal timestamp value is used to create a new entity in the graph, which has a relation towards every measurement taken at that moment, and an inverse property from each of those measurements back to the timestamp entity.

```
ex:measurement1 ex:measuredAtTime ex:timestamp1
ex:timestamp1 ex:measuredDuring ex:measurement1
```

3.2 Step II: Embedding creation

For the embedding step of our experiment we use the pyRDF2vec implementation of RDF2vec. [11] This is an implementation of RDF2vec light, which only creates embeddings for specific entities in KGs, specifically timestamp entities, not for all nodes. Timestamps were selected to be embedded because they correspond with the classification task, which is described is 3.3. This also provided the freedom to add and remove devices and measurements without having to adjust the pipeline, keeping it as similar as possible between experiments. All experiments used the reverse function of pyRDF2vec.

An initial experiment compares different settings for the following hyperparameters: number of walks per entity, length of the walks and number of epochs the embeddings are trained. A subset of the dataset described in Section 3.4 was used that only included 2000 time points, to decrease the time it takes to train the embedding model. Based on the results of this first experiment settings were selected to run the remaining experiments.

Table 1: Range of the hyperparameter settings that were explored.

hyperparameter	settings
number of walks	6 10 25
walk length	2 4 6
number of epochs	1-40

3.3 Step III: Evaluation Task

The evaluation of the embeddings quality is a classification task, classifying each timestamp as either hot or cold. The timestamps were labelled by dividing the dataset in two by sorting the timestamps from hot to cold, and labelling the first half as *warm*, and the second half as *cold*.

The classification was performed with a Multilayer perceptron (MLP) implementation written using PyTorch [8], consisting of two hidden layers with 512 ReLU activation nodes. These were used as the default hyperparameters. The input to the MLP were the embeddings of the timestamp, and the output was the hot or cold label.

3.4 Datasets

At the moment of writing we were not able to find open IoT graphs, which is likely due to the privacy concerns of making peoples IoT device data publicly, therefore we created these ourselves, using an energy consumption dataset on device level containing real device measurement data. We created the graphs following the mapping procedure detailed in [13]. Templates were created for each device in the dataset, mapping each measurement to the graph together with related properties.

In order to create the entity files for the embedding step the timestamps of a dataset were collected and used to connect with the outside temperature at that location at the specific time. From those temperatures the hot and cold labels were created. The outside temperature values were retrieved through worldweatheronline.com.

OPSD Household data The OPSD Household 1hour dataset [7] consists of energy consumption measurements from devices in different types of buildings, of which we use only the six residential homes, since this is closest to the type of measurements we want to investigate. Table 2 shows the distributions of the different device types over each separate residence.[3]

Table 2: Distribution of device types over residences in OPSD Household dataset.

	device types										
residence id	*grid import*	*grid export*	*solar panel*	*dishwasher*	*electric vehicle*	*refridgerator*	*freezer*	*heat pump*	*washing machine*	*circulation pump*	
residence 1	■		■	■			■	■	■		6
residence 2	■			■			■		■	■	5
residence 3	■	■	■	■		■	■		■	■	8
residence 4	■	■	■	■	■	■	■	■	■		9
residence 5	■			■		■			■		4
residence 6	■	■	■	■			■		■	■	7
	6	3	4	6	1	3	5	2	6	3	39

The dataset contains measurements taken over a five years duration, but not every device recorded measurements for the entire period. In order to have a complete dataset we chose to extract a subset of ten months where all devices had recorded measurements, by removing only two devices (the freezer from

[3] The IoT KGs can be found at https://anonymous.4open.science/r/SAREFized-OPSD-household-graph-84A4

residence 2, and the grid export from residence 6), if we would have included these the measurements would only be available for two months.

The final manipulation to the data was to transform the energy consumption measurement from its original value of accumulated consumption since start, to accumulated consumption over the last hour. This manipulation was performed to ensure that the measurement values in the graph would be recurring, which would not be the case for accumulated measurement values, because those would only increase.

The final graph represents 8133 timestamp entities linking to measurements from 37 devices from ten device types, spread out over six residences. Three different versions of this graph were created in order to be able to distinguish between the effects of adding more devices from within the same home, and adding devices from other homes. The following short hand is used to refer to different compositions of the graph:

res1dev1: this graph uses only measurements of one device. this is the heat-pump from residence 4. This graph contains 89477 triples.

res1devA: this graph uses all measurements from all devices in one home, in this case all devices from residence 4.This graph contains 715,765 triples.

resAdevA: this graph uses all measurements from all devices in all available home that are available. It contains 3,220,912 triples.

Semantically Enriched data After the semantic enrichment described in Section 3.1 was performed on the OPSD dataset the graphs grew:
res1dev1: This graph contains 122,007 triples.
res1devA: This graph contains 976,005 triples.
resAdevA: This graph contains 4,391,992 triples.

4 Results

In this section the results of the experiments are reported. The hyperparameter selection uses the subset with the 2000 hottest and coldest time points of the OPSD dataset, while the full OPSD dataset is used in Section 4.2.

4.1 Hyperparameter Selection

The intention of the hyperparameter selection experiment was to find a combination of short training time and ideally accuracies that were representable for the quality of the embeddings learned from each graph, not to optimize for the highest accuracy possible.

Table 3 displays accuracies and training times achieved with the resAdevA-2000 graphs, for multiple hyperparameter settings. Each combination was trained for 40 epochs.

After removing the options that were ended prematurely due to an expected training time of at least 50 hours, marked here with blacked-out cells, the grid was reduced into two directions to explore, increasing either the number of walks,

Table 3: Classification accuracy on the test set and training time of the embedding models for multiple hyperparameters, on the OPSD-2000 graph.

		SAREF basic						SAREF enriched					
		walk length											
		2		4		6		2		4		6	
		acc	sec	acc	sec	acc	sec	acc	sec	acc	sec	acc	sec
number of walks	6	50.92%	222	51.5%	7,488	51.67%	14,458	61%	288	70.25%	25,323	77.83%	53,563
	10	53.17%	233					75.83%	315				
	25	52.58%	288					92.83%	391				

or the walk length. By increasing the walk length the difference in accuracy between using the basic or enriched graph grows, but the training time also grows extensively. Alternatively increasing the number of walks had a similar effect on the accuracy, but the time to train only grew slightly between different settings.

We attribute this difference in training time to the increase of entities that were needed to be retrieved during training of the models with an increased walk length. As discussed earlier in Section 2.1, IoT graphs require only a few steps through the graph to reach all entities in the graph. By increasing the walk length more entities are included that are connected to many entities, increasing the possible unique walks greatly. To handle this more compute time is required.

Based on our requirements of short training time and distinct accuracies the best option is a walklength of 2 and 25 walks per entity, since the time increase is limited, and the accuracy increase allows for more range between the embedding quality of basic and enriched graphs.

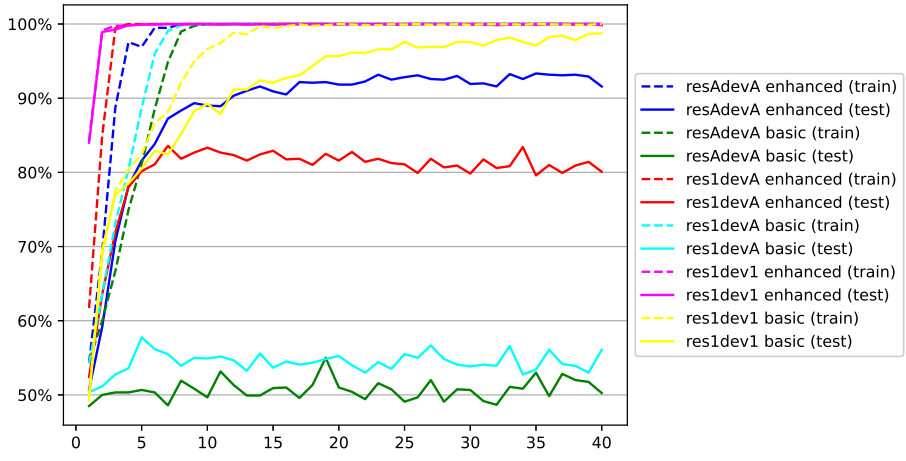

Fig. 4: Effect of the number of epochs on the accuracy of classifiers trained with various graphs.

In order to determine the effects of the number of epochs the embeddings model is trained, we trained six embedding models, one for each version of the basic and the enriched graphs. The training and test accuracy were calculated after each epoch, for a total of 40 epochs. The results are vizualised in Figure 4. Training accuracies are portrayed with dashed lines, test accuracies with solid lines.

All training accuracies reach 100% after 10 epochs, except res1dev1-basic, which takes 20 epochs. Two out of three test accuracies for basic graphs never increase above ±50% and are therefore not taken into account. The enriched graphs test accuracies remain consistent after 20 epochs.

Based on these findings we trained the model for the next experiment for 20 epochs, this is enough to reach the plateau of resAdevA enriched and res1devA enriched, but will also avoid overfitting by not training too long.

4.2 Basic vs Enriched

The results of the experiment have been visualized in Figure 5. For both datasets, and for any amount of devices being used, the classifier trained on the enriched graph always outperforms the classifier trained on the basic graph. The smallest difference occurred with res1dev1 of the OPSD data, and even then the accuracy was still significantly different (p=*0.00766*).

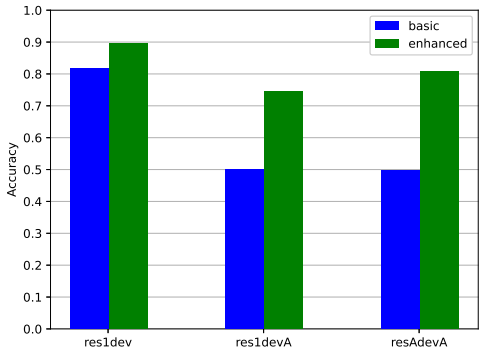

Fig. 5: Average accuracies of the classifiers, trained on different amounts of devices, for the full OPSD data.

Accuracies of classifiers trained on basic graphs were, with the exception of res1dev1, always close to 50%. This indicates that these embeddings did not contain any information that the classifier was able to learn from, since only two possible classes exist and therefore always picking the same class yields an accuracy of 50%. The exception being the classifier trained on the res1dev1 graph. A

possible explanation is that because this is a singular device, all measurements values that were available as literals were enough for the embedding model to learn a signal, because that one device by itself happens to be a very good indicator for what was being classified, the energy consumption of a heatpump to predict the outside temperature.

Accuracies of classifiers trained on enriched graphs were highest when learning from res1dev1. As with the classifiers trained on basic graphs, this could be explained by the fact that this device is a good indicator for the classes. When more devices from the same residence were added (res1devA) the accuracy of classifiers reduced, which might be explained by all the additional devices, and therefore measurements, that were added, are not providing more, or the right, information for the embeddings, causing the classifiers accuracy to decrease. For example, a dishwasher would not be a good predictor for the outside temperature.

Classifiers trained with embeddings from resAdevA had a significantly higher accuracy (p=*0.00099*) then classifiers trained with embeddings from res1devA. The difference in relation to res1devA and res1dev1 being that not only other devices were added, but other devices from different homes were added, which include new devices that are good indicators for the classification labels.

5 Discussion

During this research some unexpected observations were made, which are presented in this section. They serve as a way to explain our process, but also to suggest some possibilities for future research.

5.1 Data Validity

For now only energy consumption measurements were used, primarily because this was the only public data we could acquire at the time. More heterogeneous measurements could have a different effect of the embedding methods. Additional measurements could for example include CO2 levels or room occupancy.

In Section 2.1 we explained why we expect our results to generalize to IoT KGs created with different ontologies, based on the similarities in their design. Additional research could explore the differences, to determine if semantic enrichment has a bigger or smaller effect.

5.2 Enrichment Choices

The rounded values were now always rounded to one decimal, using more decimals would result in more rounded value entities, each having less connections with measurements, since less measurement values would be rounded to the same rounded value. But rounding to too many decimals runs the risk of becoming too specific, with a “rounded” value for every measurement value. How much rounding is applied could be an interesting parameter for future research.

Even though all the measurements were about energy consumption, they were made by many different devices. But the rounded values (and literal values) made not distinction between, for example, a measurement of 0.6 by a freezer, or a measurement of 0.6 of a dryer. Making separate entities out of these, such as `0.6_freezer` and `0.6_dryer` would allow the embedding method to learn the difference.

5.3 Evaluation Task

For now we used one classifier, without any optimization. For the purposes of this experiment that sufficed to compare the accuracies of the classifiers trained on different KGs, but other classifiers could be used to examine if they are differently effected with different KGs.

The classifiers are overfitting on the training set, as can be observed in Figure 4. This could be an indication that we need more data, or that the embeddings are creating too specific representations for the entities. RDF2vec not only embeds similar entities close together, it also tries to embed dissimilar entities further away, which could cause all the entities to be spread out evenly, leaving no generalisation to learn for the classification model.

In our experiments we predicted the outside temperature based on a house, but this outside temperature is the same for all the houses in each dataset, because we only have one outside temperature on city level. Therefore the result that including more houses leads to a higher accuracy can also mean a more representative house was added, that better represents what is being classified. Future research should explore the benefits of sharing information between different homes in order to improve embeddings, but also explore how this effects privacy concerns.

6 Conclusion

In this research we set out to answer the question: *What is the effect of semantically enriching a KG on the quality of entity embeddings learned from it.* When we consider the accuracy of the classifier an indicator of the quality of the entity embeddings, the results of our experiment make it clear that semantic enrichment had a positive effect on the entity embedding quality. The semantically enriched KGs outperformed their corresponding basic KGs in every instance, regardless of graph size.

Because the information added by the semantic enrichment is implicitly already available in the KG, changing the ontology design to add the enrichments would not provide any new information to the ontology. Instead we see the semantic enrichment step as an additional part of preprocessing the graph, to enhance the quality of the embeddings.

Acknowledgements. This work is part of the InterConnect project (interconnectproject.eu/) which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 857237.

References

1. Akasiadis, C., Pitsilis, V., Spyropoulos, C.D.: A multi-protocol iot platform based on open-source frameworks. Sensors **19**(19), 4217 (2019)
2. Compton, M., Barnaghi, P., Bermudez, L., GarcíA-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., et al.: The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Journal of Web Semantics **17**, 25–32 (2012)
3. Daniele, L., den Hartog, F., Roes, J.: Created in Close Interaction with the Industry: the Smart Appliances REFerence (SAREF) Ontology. In: International Workshop Formal Ontologies Meet Industries. pp. 100–112. Springer (2015)
4. Iana, A., Paulheim, H.: More is not always better: The negative impact of a-box materialization on rdf2vec knowledge graph embeddings. In: CEUR Workshop Proceedings. vol. 2699, pp. Paper–5. RWTH (2020)
5. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013). https://doi.org/10.48550/ARXIV.1301.3781, https://arxiv.org/abs/1301.3781
6. Moreira, J., Daniele, L., Pires, L.F., van Sinderen, M., Wasielewska, K., Szmeja, P., Pawlowski, W., Ganzha, M., Paprzycki, M.: Towards iot platforms' integration semantic translations between w3c ssn and etsi saref. In: SEMANTICS workshops (2017)
7. Open Power System Data: Data Package Household Data. Version 2020-04-15 https://data.open-power-system-data.org/household_data/2020-04-15/. (Primary data from various sources, for a complete list see URL). (2020)
8. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
9. Portisch, J., Paulheim, H.: The RDF2vec Family of Knowledge Graph Embedding Methods (2022)
10. Ristoski, P., Rosati, J., Di Noia, T., De Leone, R., Paulheim, H.: Rdf2vec: Rdf graph embeddings and their applications. Semantic Web **10**(4), 721–752 (2019)
11. Vandewiele, G., Steenwinckel, B., Agozzino, T., Ongenae, F.: pyrdf2vec: A python implementation and extension of rdf2vec. arXiv preprint arXiv:2205.02283 (2022)
12. W3C: Web of Things (WoT) Thing Description (2020), https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
13. van der Weerdt, R., de Boer, V., Daniele, L., Nouwt, B., Siebes, R.: Making heterogeneous smart home data interoperable with the SAREF ontology. International Journal of Metadata, Semantics and Ontologies **15**(4), 280–293 (2021)