ISSN 2007-9737

Word Embeddings for IoT Based on Device Activity Footprints

Kushal Singla, Joy Bose, Nitish Varshney

Samsung R and D Institute, Bangalore, India

{kushal.s, nitish.varshney}@samsung.com, joy.bose@ieee.org

Abstract. With the expansion of IoT ecosystem, there is an explosion of the number of devices and sensors and the data generated by these devices. However, the tools available to analyze such data are limited. Word embeddings, widely used in the natural language processing (NLP) domain, provides a way to get similar words to the current word. In this paper, we extend the theory of word embeddings to the area of IoT devices, proposing a method to generate the word embeddings for IoT devices and sensors in a smart home based on their activity. We model IoT devices as vectors using a concept like Word2Vec and App2Vec, where the time between the device firings is also taken into account. These computed word embeddings can be used for a variety of use cases, such as to find similar devices in an IoT device store, or as a signature of each type of IoT device. We show results of a feasibility study on the CASAS dataset and a private real-world dataset of IoT device activity logs, using our method to identify the patterns in embeddings of various types of IoT devices in a household. We get a probability of more than 0.65 for similar types of devices clustering together, independent of session gap value and embedding vector size for the CASAS dataset. We also get a probability of 0.4 on the private dataset, independent of session gap value and embedding vector size.

Keywords. Word2Vec, IoT2Vec, word embeddings, smart home, internet of things, natural language processing.

1 Introduction

The Internet of Things (IoT) has grown exponentially in recent times, with IoT sensors and devices being used in many real-life use cases such as smart homes. These sensors generate lots of data each second.

Analysis of the data generated by the IoT sensors and devices in a smart home can lead to valuable insights about the usage habits and the devices themselves. However, the number of studies on real-life smart home IoT datasets to get insights about the device usage patterns is limited.

The popular Word2Vec model [1] provides ways to generate word embeddings based on the usage of the words in one or more documents. Here, an IoT usage log can be considered as a document, and the logs of a given IoT device within a given small time window can be treated as a word. Such word embeddings can be used, for example, to find similarity between two documents. App2Vec [2] is a modified adaptation of Word2Vec for apps based on app behavior, with additional weightage based on time of firings.

Using an approach like Word2Vec and App2Vec, we attempt to create embeddings for IoT devices based on their usage, using data obtained from the usage logs of the devices. These embeddings can be used, for example, to find similar IoT devices for a given device. Such a model we call IoT2Vec.

In this paper, we describe the generation of embeddings using some publicly and privately available IoT datasets and describe some principles how this can be adapted for different IoT device usage data. We create a model to take the device usage data as input, create embeddings for the devices and identify a new device of the same type based on similar usage data. There can be several applications for such a model to create embeddings and find similar IoT devices.

One application of such a model is to make a search function to search for similar devices in the vicinity. Using this method, defective devices can be replaced based on their function.

If we know the footprint of the IoT device, we can identify which another device is best suited to replace it. This can also be used to recommend similar IoT devices from different vendors, such as in an online e-store. Another useful application for such a model is routine disruption due to faulty/malfunctioning devices, where a faulty/malfunctioning device in the routine can be replaced by similar devices in the vicinity. It can also be used to transfer a given user's routine from one location to another in case the user has changed their home location or gone for an outing or leisure travel.

An-other common application can be to build a location classifier based on IoT de-vices in that location. For example, given that a pub usually has dim lights, it is likely that another pub will also have similar light settings. Therefore, knowing the IoT devices and their footprint in each location, we can identify the type of location. Another useful application is routines identification, which is the primary enabler for automating user's action in a smart home.

Automatically identified routines allow users to control and automate many aspects of a home without user intervention and without going into unnecessary hassle of creating home automation recipe for themselves. Especially for elder persons living alone at home, it is very difficult as they need to be aware of their own routine, variations followed to pursue routine and technical knowledge of the relevant home automation devices in home.

The rest of this paper is organized as follows: in the following section we sur-vey approaches to current approaches related to identifying similar IoT devices. Section 3 describes the theory and method which we use to generate the embeddings. Section 4 gives the results of our method applied on public and private IoT datasets, along with validation and real-life use cases. Section 5 concludes the paper.

2 Related Work

There are a few instances of related work in applying machine learning to find similar IoT devices.

Xu [3] proposed a system for searching for and finding similar IoT devices as a result of user queries, based on a similarity measures based on the semantic and other properties of the IoT objects such as the object location. Kang [4] suggested various methods to identify correlations between IoT devices, including attributes such as location, usage count, sensor list, service name etc. They also suggested using Word2Vec model to calculate the adjacency between IoT devices. However, they did not provide concrete details on exactly how the vectors would be calculated and the issues involved when working with real datasets.

Tian [5] mentioned a mechanism to automatically collect security related information from an IoT app. Palit [6] mentioned a system to identify IoT resource requirements such as sensor accesses from service descriptions, using NLP techniques to parse the Android app descriptions to determine which sensors were required. Hong [7] used similarity measures between IoT devices to provide context aware services to users. Truong [8] proposed a method for searching similar IoT sensors, computing a similarity score based on fuzzy sets.

The patent of Derek Lin [21] describes analysis of device similarity using methods such as principal components analysis. However, most of these approaches require prior information about the IoT devices, such as the parameters needed to determine similarity. Such prior information might not be readily available.

Word embeddings have been used for a variety of use cases including product and item recommendations [15, 16], similar nodes in a network [17], and multi-media [18-20]. However, they have not been applied to determine similarity for IoT devices as of late. In this paper, we use IoT sensor or device usage patterns to create the word embeddings and identify the type of IoT device. Having such an approach has the advantage of not needing prior information about the devices, only their usage data is needed instead.

3 Model for Generation of Word Embeddings

Most IoT devices are used in certain patterns that repeat over time. Similar kinds of IoT devices will have similar activity footprints. The IoT device type can be identified by the pattern of usage.

The time of usage and location of devices also carries useful information. A model can be trained to encode this pattern as word embeddings, which will help to identify the IoT device that has similar patterns.

3.1 Theory of Word Embeddings for IoT Devices

In the Word2Vec model [1], a neural network is trained to map a word to a vector in such a way that the probability of predicting a word (target) given a context of words (in CBOW model) is maximized. The model is trained on a large dataset of documents, so it is expected to capture all possible variations and patterns in which the words are used together. So it aims to maximize the following log likelihood for all words in the dictionary:

$$\log P(W_i|context) = similarity(W_i|context) / \sum_k similarity\ (W_k|context), \quad (1)$$

where Wi is the word vector corresponding to the ith word, and similarity between vectors is measured by cosine distance. For example if the model is trained on the sentence ‘the cat is on the table’, the aim is to maximize the probability of predicting ‘cat’ when the following words are presented to the model ‘the ___ is on the table’. In the Skip-Gram model, on the other hand, the objective is to present a word and predict its surrounding words in a given window (or context length). So if the word ‘cat’ is presented as input, the model should learn to predict ‘the, is, on, the, table’ as output.

In this paper, for IoT devices and sensors, we aim to create a word vector for each unique IoT device such that its activity can be predicted given its context. If sensor 1 fires along with sensor 2 and 3, then given the sensor 2 and 3 firing, the model should be able to predict sensor 1. In this sense, it is like the Word2Vec model. We train our model on a dataset of IoT device activities and hope to capture the patterns of cooccurring sensor activity for different types of sensors.

In our approach, we define an IoT device session sequence as being similar to the app sequence defined in the App2Vec paper: If D_1, D_2, D_3 are three IoT devices or sensors in a household, and g_1, g_2, g_3 etc. are the time gaps between the transition times of these devices, then an example usage session can be represented as (D_1, g_1, D_2, g_2, D_3, g_3, D_1, g_4, D_2).

Here, we define a session as a certain length of time, say 60 seconds or 600 seconds. We only consider the transitions of the sensors (OFF to ON and ON to OFF for binary states, or a defined range of values for bins in case of sensors like thermostat) for our purpose. Within a session, the aggregate activities of all the sensors in sequential order (D_1, D_2, D_3, D_1, D_2) is a sentence and the activity of any single device or sensor (D_1) is a word. After getting the words and sentences for all the sessions in our dataset, we analyze them to create the embeddings vectors using the Word2Vec or App2Vec method.

We have two choices regarding the time gap between sensor activations (g_1, g_2, g_3):

1. Ignore the time gap and consider only the order of transitions of devices within a session (D_1, D_2, D_3, D_1, D_2) when creating the sentence and words for the embeddings vector.
2. Consider the time gap and introduce a weight x^t for the context words in the CBOW model, where t is the time gap in minutes from the target word and $x<1.0$ is a similarity factor that decays with time difference between the activation of the target device and context device. The App2Vec model [2] used a similar weightage concept (with x empirically determined as 0.8) to determine the similarity of app usage vectors. The idea is that if two IoT devices have a small time gap, their vectors should be more similar than if two devices have a larger time gap.

3.2 Method to Create a Word Embeddings Vector for IoT

Using the previously discussed theory, we define some steps to generate and analyze the word embeddings from IoT device sensor logs.
Our method includes the following steps:

1. Filter out the IoT sensors whose data is not meaningful or we cannot make sense of the data.

2. Examine the activity data of the selected sensors to see whether it shows meaningful activity or actions.
3. Extract only the values where the sensor state is in transition (e.g. ON to OFF or OFF to ON).
4. Build a session of the sensor values (similar to sentence in NLP domain) by choosing a session gap. Session gap is the gap of time where we construct the boundaries of each session. Within a session, we consider both the discussed approaches (a) only consider the order of firings of different devices or sensors. The exact time gap between firings within a session is ignored and (b) set a weightage x^t where $x<1.0$ and t is the time gap in minutes.
5. Once the sessions are defined, we treat each session as a sentence and the device Id as a word. Each sentence will contain a sequence of IoT device Ids such as (M008, M009, D010) which is the order of firings of the devices within the session.
6. Train the Iot2Vec model using the session data extracted from the dataset. The input to the training model is the document comprising of the created sentences in the previous step. The output of the model is the embedding vector for each type of sensor or device. We can select a certain dimension, such as 100, for the size of the vector embeddings.
7. Compute the similarity between the vector embeddings of each sensor/device with the other sensors. Furthermore, we perform dimensionality reduction and construct a t-SNE plot for easier visualization of the sensor activities in terms of contextual similarity, i.e. which IoT devices or sensors are being activated together.
8. Visually examine the t-SNE plots to detect patterns of similarity in the activity data for each type of IoT device or sensor with other sensors.Following the above steps, the embeddings vector of a given device or sensor type can be generated from its activity logs.

Table 1 shows the algorithm to generate the word embeddings with and without the weighed similarity factor.

Table 1. Algorithm to identify the embeddings vector for a given device

Input: Input: Device activation sequence for the devices D_1, D_2, … D_n

Output: Embeddings vector for devices D_1, D_2, … D_n

1. Break the device activation sequence into sessions for a given value of session gap, considering only device state transitions. The session represents a sentence.
2. Train a model using CBOW, similar to word2vec, using the generated session sequences
3. Once training is completed, the embeddings for each device are generated
4. Repeat the steps 2-3, using a weighed CBOW model with weight x^t where t is the time gap in minutes between device activations and x is a similarity factor <1.
5. **exit**: end procedure

Table 2. Algorithm to identify device type from activity logs

Input: Stored device embeddings for different device or sensor types D_1 (E_1), D_2 (E_2) etc.

Output: Device type of a new device D_i given its usage data

1. Generate embeddings vector E_j from the usage data of the new device D_j
2. Compute the similarity of the embedding vector E_j with each of the stored embedding vectors E_1, E_2 ...
3. Find the device D_i whose similarity value of the embedding vector E_i with E_j is highest and above a threshold
4. Define the device type of D_j as equal to the device type of D_i
5. ***exit***: end procedure

3.3 Method to Identify the Device Type of a New or Unknown IoT Devices from its Usage Logs

The table 2 shows the algorithm for identifying the device type of a new or unknown IoT sensor or device from its activity logs, once we have stored device embeddings of a set of devices. The principle is to generate the embeddings vector for the new device and determine which of the stored embeddings is closest to the generated embeddings vector.

4 Experimental Details and Results

For our experiments to validate our method and to explore the possibility of generating embeddings for IoT devices, we needed one or more datasets that would provide us with data from multiple IoT devices in the same locations over a period of time.

For our purpose, we tried several candidate datasets [9-12] and finally chose a dataset 20 from the Kyoto dataset list of CASAS [11, 12]. This dataset has 2 years' worth of data from a household consisting of two residents, with various IoT devices including motion sensors, doors (fridge, freezer, and microwave), shelves etc. Each data item consists of the following fields: time, sensor name, sensor state.

Fig. 1 shows an extract from a layout of a room in a house in the CASAS Kyoto dataset 20 [11, 12].

We then created word embeddings for various devices in the dataset for which we have data. We then used this embeddings to identify the devices. We analyzed the dataset in Spark and used Word2Vec to find patterns in the data.

During the preprocessing step, we ignored the light sensor, gyro sensor and a few others, since they were firing without any discernible patterns. We selected the following sensors for analysis: Motion sensor, door sensor, item sensor, shake sensor, fan sensor, experimental switch.

We then obtained a sequence of sensor states, belonging to multiple sensors, ordered by time. We ignored the actual time of sensor state change and only noted the sequence.

Our objective, as mentioned earlier, was to determine the similarity between different sensors on the basis of their activity.

In our chosen dataset [11, 12], the device D008 is a door sensor corresponding to a freezer door, where the freezer is located in the kitchen.

The similarity between vector embeddings that we obtained for this D008 sensor for the 60 second session gap is as below:

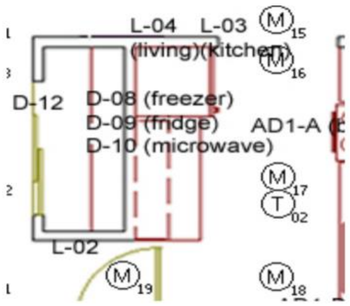

Fig. 1. Extract from the layout of a room in the CASAS Kyoto dataset [11], showing how some motion sensors (beginning with M) and doors (beginning with D) are located close together in the kitchen

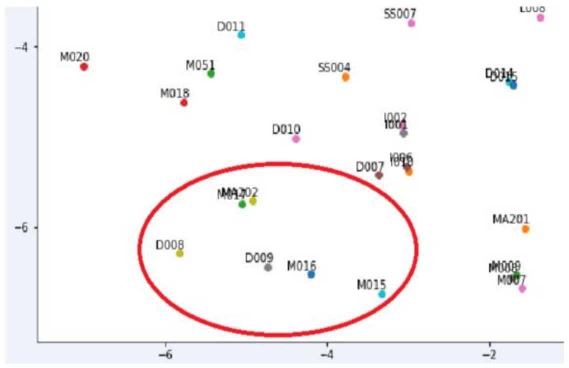

Fig. 2. Extract from the t-SNE plot of the activity of sensors from the CASAS Kyoto dataset [10], using 60 seconds as the interval, showing the contextual proximity of the motion sensors close to the kitchen (M015, M016) and door sensors of fridge (D009) and freezer (D008)

```
D008 [('M017', 0.49945521354675293),
('M016', 0.48164984583854675),
('MA202', 0.4487079977989197),
('M018', 0.4332207143306732),
('D009', 0.41653889417648315),
('D015', 0.3721662163734436),
('M015', 0.3238069415092468),
('M051', 0.2985246777534485),
('D010', 0.2684941589832306),
('D014', 0.24952027201652527)].
```

From the above similarity between vector embeddings, we can derive the same conclusion, that door sensor D008 is close to motion sensors M017 and M016 which are in close proximity.

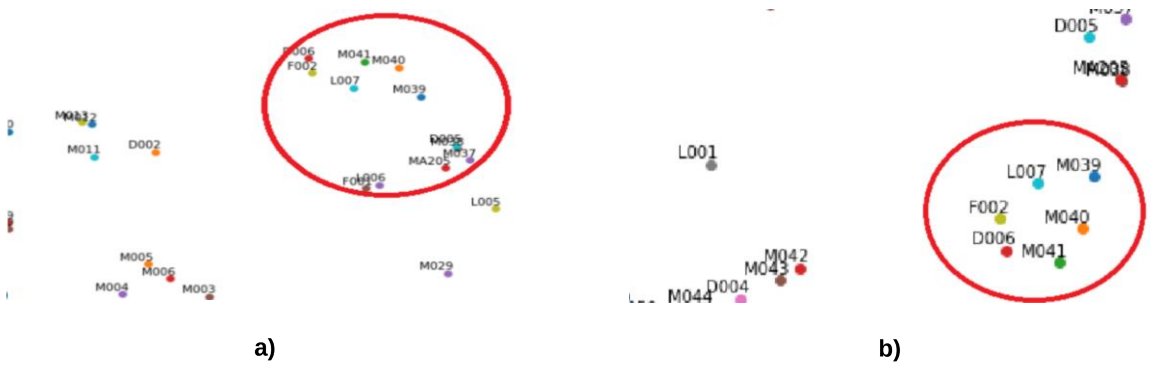

Fig. 3. (a) Extract from the t-SNE plot for Kyoto-20 Dataset (b) Sensor activity for 600 seconds gap. Here, sensors located near the toilet M038, M039, M040, M041, D006, D005 show similar patterns of activity across session gaps

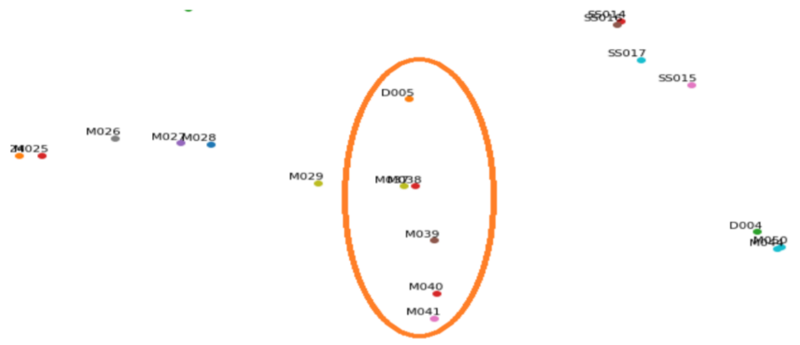

Fig. 4. Extract from the t-SNE plot of the activity of sensors from the CASAS Kyoto-20 dataset [11], using 600 seconds as the interval and time decay factor weight x^t for x= 0.9 and t being the time gap in multiples of 15 sec, showing the contextual proximity of the motion sensors close to the toilet (M038 to M041) and door sensor D005

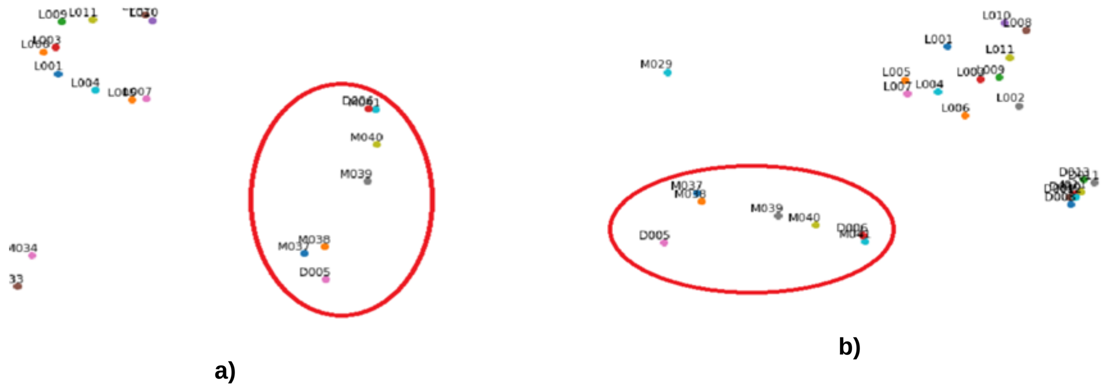

Fig. 5. Extract from the t-SNE plot for Kyoto-11 Dataset of CASAS, with (a) 60 seconds and (b) 600 seconds as the interval. In both cases we can see the sensors D005, D006, M037 to M040, which are in the toilet area, having similar activity

Examining the figure 2, we see that door sensor D008 (freezer, located in the kitchen) comes contextually close to motion sensors M015 and M016, located close to the kitchen, and the door sensor D009 of the fridge, also located in the kitchen. Looking at the layout of the house in fig. 1, we see that the sensors D008 and D009 are located in the kitchen and motion sensors M015, M016, M017 are located close to the kitchen. We can explain their contextual similarity as follows: when a person comes into the kitchen, they would activate the motion sensors close to the kitchen, after which they would open the fridge and freezer to get or make some food.

Based on this, we conclude that it is feasible to identify IoT devices based on their contextual similarity. In the following subsections, we analyze a few trends of device activity based on various parameters.

4.1 IoT Device Activity for a Given Value of Session Gap, with Time Gap Ignored

In this subsection, we attempt to find some trends in IoT device activity plotted using the t-SNE method using our word embeddings computed for various devices.

First, we determine whether the IoT device activity across different session gaps shows any significance. For example, it is possible that when we choose a small gap such as 10 seconds, device A and B are close, however when we increase the time gap to say 600 seconds, device A and C are close. So, we visually inspect the T-SNE plots [13] of the activity data to see if that can indeed be the case, or whether similar devices always cluster together.

Fig. 3(a) shows the device activity and sensor locations near the toilet area for a 10 seconds session gap. On visual inspection, we can see that motion sensors M039, M040, M041, D006, D005 etc. show correlated activity patterns. This is also confirmed from a look at the similarity measure of distance, where the vector embeddings for these sensors show closest Euclidean similarity between each other.

Fig. 3(b) shows the same sensors activity for a 600 seconds gap for a session. We see that the same sensors that were active together for a 10 second session gap are also active for a 600 seconds session gap.

Hence, we conclude that for this choice of sensors, the proximity of location (all these sensors are in the toilet area) translates into contextual proximity as well, regardless of session intervals. This could be because whenever someone uses the toilet, the motion sensors and door would always be triggered together. However, for a different choice of sensors, this might not be the case and the timer could be a factor in deciding which sensors trigger together.

4.2 IoT Device Activity, Weighing for Time Gaps within a Session

In this subsection, we repeat the previous experiment for session gap 60 seconds but weighing for time gaps within a session. We choose a session gap of 600 seconds, and time decay factor weight x_t for x= 0.9 and t measured in multiples of 15 seconds. The results are shown in fig. 4.

We can see that here too we get the same trend as when ignoring time gaps: the sensors in close proximity M38 to M41 also show closeness in the t-SNE plot.

Perhaps repeating the experiment in the future for a larger time gap (to allow for the distance to be more pronounced when the sensors fire further apart) and varying the x and t parameters might show more interesting observations in trends.

4.3 IoT Device Activity for Varying Datasets for the Same Sensor Type, with Time Gap Ignored

In this analysis, we seek to learn if the co activity of different types of sensors is sustained across different datasets.

For this experiment, we chose the Kyoto 11 dataset which was part of CASAS [11, 12]. It had the same layout as the Kyoto 20 dataset that we used earlier. However, the year the data was collected is different. The sensors M40, M41, door sensor D006 are located in the toilet-cum- bathroom.

As earlier, we plotted the word embeddings for the sensors with 60 seconds and 600 seconds gap. The plots are shown in Fig. 5. As we can see, here too the sensors near the toilet area have similar activity as before. Although a larger study is needed, we can postulate from this data that it is feasible to believe that the patterns of sensor co-activity as per contextual similarity can hold across datasets.

4.4 Validation: Probability of Similar Device Types Clustering Together

In order to validate the usefulness of out IoT2Vec model to predict similar devices, we also calculated and plotted the vector similarity of the embedding vectors of similar devices from other devices of the same device type. For each unique device, we first determine the closest device, then determine the type of the closest device. We compute the ratio of positive matches Vs the number of total devices and plot this across different session gap values. We repeat this experiment with and without the weighed decay factor, as explained in earlier sections.

The result is plotted in Fig. 6 for an embedding size of 300. As we can see, the probability is higher than 0.65 across session gaps for similar types of devices clustering together, thus validating our approach. The session gap of 600 seconds produced the best results. The probabilities with a weighed decay factor of 0.9 (where longer time differences between device activations is penalized with a decay factor) were higher across session gaps than the probabilities with no decay factor, indicating that most device activations appear close together in time.

Fig. 7 is a plot of the probability when the word embedding size is varied from 50 to 300, for a session gap of 600s. As we can see, the choice of word embedding size does not affect the probability significantly. Regardless of the embedding size, we get more than 0.65 probability that similar types of devices have similar word embedding vectors.

4.5 Private Dataset Validation: Validating the IoT2Vec Model on Real Life Use Cases

As part of developing a home automation solution, our organization has collected smart home users' data.

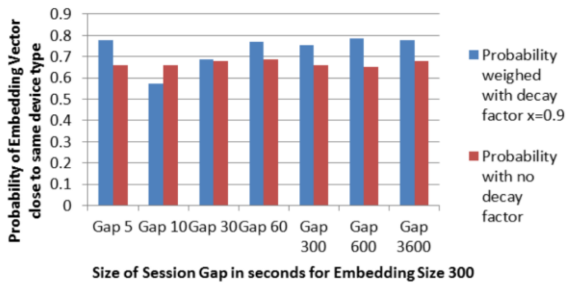

Fig. 6. Graph showing the probability of the word embeddings vectors coming close to the same device types, for an embedding size of 300, plotted with and without the decay factor

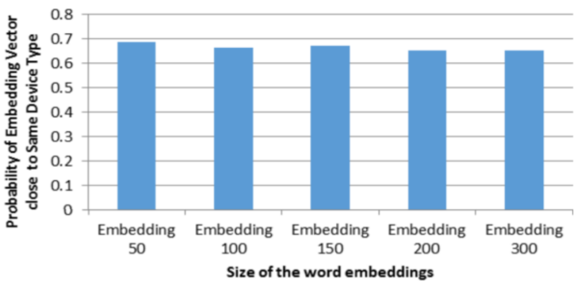

Fig. 7. Graph showing the probability of the word embeddings vectors coming close to the same device types, for a gap of 600 seconds, plotted with and without the decay factor

The dataset has 2 weeks' worth of data, which has been collected from smart home flats of 17 different users.

These smart homes had a mixture of single users and couples, with a variety of IoT sensors and appliances including Monoprice Z-Wave Plus Door/Window Sensor, Nest Weather, Switch, Smoke Detector, Smart Plug, TED5000, uDTH, Arlo Pro Basestation Siren, Evohome Heating Zone, Camera, Moisture Sensor, Rachio Zone, Aeon Key Fob, Smart Lock, Samsung SmartCam, Network Audio, Light, Hue Dimmer Button Controller AB, Lightwave On Off Device, Weather, Simulated Contact Sensor, Aeon Minimote, Motion Sensor, Z-Wave Device Multichannel, Remotec ZRC-90 Scene Master, Z-Wave Switch Generic, LAN Hue White Ambiance Bulb, Lightwave Dimmer Switch, LIFX Color Bulb, Door Bell, Multi-functional Sensor and Logitech Harmony Hub

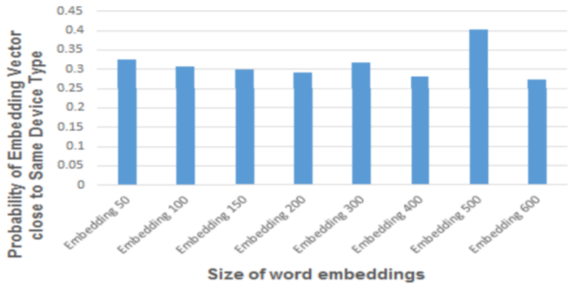

Fig. 8. Graph showing the probability of the word embeddings vectors coming close to the same device types, for an embedding size of 50 to 600, plotted with the decay factor, for the private dataset

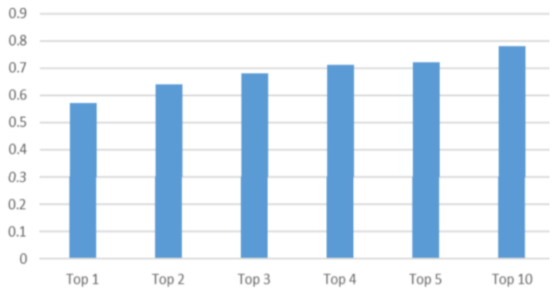

Fig. 9. Graph showing the probability of minimizing user discomfort in case of Routine Disruption, for the private dataset. The X axis plots K, where the top K similar devices are identified, and Y axis plots the probability of minimizing user discomfort

C2C. In this dataset, the device data is time ordered and is represented by a sensor event tuple as follows: <timestamp;masked device id;device state; device category>

This dataset is more significant than the previous (public) dataset, since the variety of the users and the complexity of the devices / appliances and number of possible device states is higher. In addition, the home layout is unknown, so we can only find information on the type of device along with state transitions, and not where it is physically situated.

However, our organization has extracted routines for these users, which are available to us. Fig. 8 is a plot of the probability when the word embedding size is varied from 50 to 600, for a session gap of 600s for the private dataset. We can see that the embedding size of 500 provides the best results. We have utilized this dataset to validate the usefulness of the IoT2Vec model on real life use cases. One of these real-life use cases where we applied the solution was identifying alternative for faulty/malfunctioning devices, which are part of the user's routine.

Whenever any device, which is part of the user's daily routine, becomes faulty or malfunctioning, it would lead to the user's discomfort. Therefore, there is a need to find an alternative device, which can be used to act as a replacement of malfunctioning device. To evaluate the usefulness of our IoT2Vec model in this case, we tried to identify the replacements for devices (randomly assuming one of the devices are faulty/malfunctioning) in user's routine. For each user routine, we first randomly choose a device and assume it is faulty. Then we try to identify contextually similar top-K devices to the faulty device.

The identified contextually similar device is plugged into the unmodified user routine and provided to the routine identification team for evaluation. The assumption here is that using the modified routine (with the replacement device), the user should be able to perform his or her daily tasks, which the same user was originally performing using unmodified routines, without any discomfort. Fig. 9 is a plot of the probability of minimizing user discomfort in case of routine disruption by providing a new user routine after replacing faulty/malfunctioning device with top-k alternatives.

As we can see, the probability of minimizing user discomfort increases with the increase in the number of closest devices matched for similar type. However, after closest devices is set to 3, it does not increase any further. This is because for some devices such as a refrigerator, the user does not have alternate devices in the house which can be used as replacements in case of some fault in the original device.

5 Conclusion

We have proposed a method to generate word embeddings for IoT devices, based on their usage patterns. We showed that IoT devices in similar areas in a given household can be found to have similar usage patterns. We get a probability of at least 0.65 similar types of devices clustering together regardless of the session gap or word embedding size chosen. We found that the session gap does not affect the similarity, whereas using a decay factor showed a higher value of clustering similarity. Thus, it is feasible to recognize IoT devices based on the embeddings.

In future, we plan to focus on smart city scenarios. We plan to build a location classifier based on IoT devices used in that location and embedding similarity could capture the location type. We also plan to focus on activity generated by smart fridge and TV, to get higher level understandings of the patterns.

We also plan to generate routines using the IoT2Vec model. In such a scenario, rather than creating a word vector for each unique IoT device, we will generate sentence vectors using Sentence2Vec [22] for each session. Further, clustering could be utilized to identify recurring patterns in the data. Then, routines can be generated from each segment cluster using CLIQUE [23]. Another such routine generation approach we plan to evaluate is by Chanda et. al [24] who used NLP techniques such as language modeling to recommend routines.

References

1. **Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).** *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781. pp. 1–12.
2. **Ma, Q., Muthukrishnan, S., & Simpson, W. (2016).** App2vec: Vector modeling of mobile apps and applications. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM),* pp. 599–606. DOI: 10.1109/ASONAM.2016.7752297.
3. **Xu, W. (2015).** Modeling and exploiting the knowledge base of web of things. *Ubiquitous Computing*. Université Pierre et Marie Curie - *English. ffNNT : 2015PA066009ff.*
4. **Kang, H., Kim, M., Kwon, S., & Kim, N.S. (2016).** A Design of IoT Device similarity vector based workflow management system. *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1036–1038. DOI: 10.1109/ICTC.2016.7763361.
5. **Tian, Y., Zhang, N., Lin, Y.H., Wang, X., Ur, B., Guo, X., & Tague, P. (2017).** Smartauth: User-centered authorization for the internet of things. *26th USENIX Security Symposium*, pp. 361–378.
6. **Palit, A., Srivatsa, M., Ganti, R., & Simpkin, C. (2017).** Identifying sensor accesses from service descriptions. *IEEE International Conference on Big Data (Big Data),* pp. 3006–3011. DOI: 10.1109/BigData.2017.8258271.
7. **Hong, I. & Lee, Y. (2016).** Key-Device based place recognition using similarity measure between IoT spaces. *IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–5. DOI: 10.1109/ SMARTCOMP.2016.7501701.
8. **Truong, C., Römer, K., & Chen, K. (2012).** Sensor similarity search in the web of things. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–6. DOI: 10.1109/WoWMoM.2012.6263791.
9. **NOAA (2018).** http://tidesandcurrents.noaa.gov/ gmap3
10. **Intel lab (2018).** http://db.csail.mit.edu/labdata/ labdata.html
11. **WSU Casas (2018).** http://ailab.wsu.edu/casas/ datasets/
12. **Cook, D.J., Crandall, A.S., Thomas, B.L., & Krishnan, N.C. (2013)**. CASAS: A Smart Home in a Box. *Computer*, Vol. 46, No. 7, pp. 62–69. DOI:10.1109/MC.2012.328.
13. **Maaten, L.V.D. & Hinton, G. (2008).** Visualizing data using t-SNE. *Journal of machine learning research*, pp. 2579–2605.
14. **Singla, K. & Bose, J. (2018).** IoT2Vec: Identification of Similar IoT Devices via Activity Footprints. *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 198–203. DOI: 10.1109/ICACCI. 2018.8554398.
15. **Vasile, F., Smirnova, E., & Conneau, A. (2016).** Meta-prod2vec: Product embeddings using side-information for recommendation. *10th ACM Conference on Recommender Systems*, pp. 225–232. DOI: 10.1145/2959100.2959160.
16. **Ozsoy, M.G. (2016).** *From word embeddings to item recommendation.* arXiv preprint arXiv:1601.01356.
17. **Grover, A. & Leskovec, J. (2016).** node2vec: Scalable feature learning for networks. *22nd ACM SIGKDD International Conference on Knowledge discovery and data mining*, pp. 855–864. DOI: 10.1145/2939672.2939754.
18. **Wang, D., Deng, S., Liu, S., & Xu, G. (2016).** Improving music recommendation using distributed representation. *Proceedings of the 25th International Conference Companion* on *World Wide Web*

(WWW) Conference, pp. 125–126. DOI: 10.1145/2872518.2889399.

19. **Habibian, A., Mensink, T., & Snoek, C.G. (2016).** Video2vec embeddings recognize events when examples are scarce. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 39, No. 10, pp. 2089–2103. DOI: 10.1109/TPAMI.2016. 2627563.
20. **Klein, B., Lev, G., Sadeh, G., & Wolf, L. (2015).** Associating neural word embeddings with deep image representations using fisher vectors. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4437–4446.
21. **Lin, D., Kaufman, A., & Villa, Y. (2016).** *Patent and Trademark Office.*
22. **Le, Q. & Mikolov, T. (2014**). Distributed representations of sentences and documents. *International conference on machine learning*, pp. 1188–1196.
23. **Dehghan, A., Modiri-Assari, S., & Shah, M. (2015).** Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4091–4099.
24. **Chanda, P.K., Varshney, N., & Subash, A. (2017).** Applications of Natural Language Techniques in Solving SmartHome Use-Cases. *Natural Language Processing and Information Systems (NLDB)*, pp. 214–217

Article received on 25/01/2019; accepted on 04/03/2019.
Corresponding author is Kushal Singla.