A Machine Learning Approach for Indirect Human Presence Detection Using IOT Devices

Rui Madeira
Instituto de Telecomunicações
Lisbon, Portugal
Department of Information Science and Technology
ISCTE-IUL
Lisbon, Portugal
Rui_Nuno_Madeira@iscte.pt

Luís Nunes
Instituto de Telecomunicações
Lisbon, Portugal
Department of Information Science and Technology
ISCTE-IUL
Lisbon, Portugal
luis.nunes@iscte.pt

***Abstract*—This paper describes the construction of a system that uses information from several home automation devices, to detect the presence of a person in the space where the devices are located. The detection however doesn't rely on the information of devices that explicitly detect human presence, like motion detectors or, smart cameras. The information used is the one available in the Muzzley system, which is a mobile application that allows the monitoring and control of several types of devices from a single program. The provided information was anonymized at the source. The first step was to extract adequate features for this problem. A labeling step is introduced using a combination of heuristics to assert the likelihood of anyone being home at a given time, based on all information available, including, but not limited to, direct presence detectors. The solution rests mainly on the use of supervised learning algorithms to train models that detect the presence without any information based on direct presence detectors. The model should be able to detect patterns of usage when the owner is at home rather than rely only on direct sensors. Results show that detection in this context is difficult, but we believe these results shed some light on possible paths to improve the system's accuracy.**

Keywords—ambient intelligence; internet of things; human presence detection; sensor fusion

I. INTRODUCTION

Human presence detection is an ongoing challenge in several scenarios and applications, one of them is in ambient intelligence and home automation. This projects intends to create a system to be able to infer about presence using general information and statistics of usage and interaction with several types of devices.

Ambient intelligence in computing refers to technology that is non-intrusively integrated in an environment, works without the need of intensive interaction from a user, and adapts, in an automated way, to the necessities of each user and context. The goal of this technology is the creation of complex systems with simple interfaces that enhance the quality of our daily lives [1] [2] [3].

Internet of Things (IoT) is the network of physical objects with electronics, software, sensors and connectivity capabilities that allow these devices to communicate with each other and to exchange information. By fusing the information from each device, and through the interaction of each device with its environment, we are able to build complex systems that may enhance our quality of life [4] [5] [6] [7] [8].

Home automation can be seen as an instantiation of the concept of ambient intelligence in which the environment we want to "make intelligent" is a residence, and the functions we want to control and monitor are those that control the devices that impact the house's variables, such as temperature, energy consumption or humidity for example.

Nowadays it is increasingly common to have connected devices in on our houses. [9] predicts that before 2020 the IoT will be comprised of more than 50 billion devices. Even today there are a lot of different devices available in the market for many different purposes, and the number is growing rapidly.

A problem with these devices is that each manufacturer company has its way of communicating and programming its device. Muzzley is one of the current attempts to integrate all devices by providing a mobile application and a platform that allows for the interaction with an ever growing number of different devices, independently of their brand. Among other features the application gives the user the possibility to create programmed rules with triggers and consequences. One useful trigger (or part of many condition triggers) is the proposition "if someone is at home do …-".

The information about human presence on a domestic house is interesting for several reasons:

- Allows the automation of several actions such as turn on/off devices when someone arrives, turn off when they leave.
- May aid in some forms of intrusion detection.
- Allows learning the user's routines in a non-intrusive way, since it is fundamental to know when a user is at home or when he is simply interacting with the devices remotely.
- Can help detect behavior anomalies related to health conditions.

Specific and reliable detectors are usually expensive for domestic use. The more affordable devices usually have limitations, for example a person sitting down on a couch will not be detected by a regular motion sensor. This project intends to detect presence by using information of objects not built specifically for this purpose.

The idea is to process the data generated from interactions between the user, the devices and the platform in order create general metrics, not related to each device usage or characteristic, like for example average number of interactions per day, or number of interactions in the last half hour. These will then be used to train machine learning models for presence detection.

If this proves to be a good method, it will allow to infer information about presence for users that don't have presence sensors. And may increase the overall fidelity of houses equipped with presence detectors, for instance by discarding the presence of pets, which can trigger some presence sensors.

II. Related Work

In recent years the increased general availability of computing technology, is leading to an increase accessibility of devices related to ambient intelligence and internet of things to the general audiences [3] [10] [11]. The main devices that can be found in common technology stores are light bulbs, capable of network connection and being remotely controlled. Sensors, like thermostats, carbon dioxide sensors, motion sensors. Also some houses are now built with home automation in mind [12].

This led also to an increase in research in the area. The Dream Green House [13] project, has the goal of creating the world's most intelligent house, but also focusing on ecology. The project consists of a series of subprojects, each one focusing on one aspect to improve the house, for example, a module to control energy consumption, another for temperature control, luminosity control, water consumption monitor, etc.

In 2013 [14] developed a presence detection system at a room or zone level for the house. This system infers about presence by using a large quantity of information gathered from several sources. The type of data used and its source may indicate presence explicitly, such is the case for pressure sensors placed in beds and chairs, or implicitly, in the case for example of the detection of network traffic coming from a PlayStation system.

Another example of house that implements these technologies is the Gator House [15], which has the objective of creating a smart home that can help and monitor the daily life activities of the senior population or people with special needs. One of the most interesting contributions of the project is the creation of a model architecture for the middleware to control and coordinate all the systems implemented in the house. This model is structured in layers, the base layer has to do with the raw data produced by the devices, and then, progressively through the layers the system becomes more complex and abstract in the sense of the created information.

Some devices available in the market, such as the Nest thermostat, already use information from several sources in order to infer about human presence. Nest for example uses temperature, carbon dioxide levels and noise levels for this effect. As stated before, this project is not about using specific sensors or information, it is about indirectly doing presence detection through the interaction messages and their statistics for generic users with different devices and routines. This is what makes this a unique project.

III. Processing the Data

A. The Muzzley Platform

The system generates data related to the interaction between the user application and the platform, and then between the platform and each device manufacturer systems. In order to be able to cope with the large diversity of devices and information created by these interactions, the application has a large and well defined ontology that coordinates the way each device transmits its information.

The messages from a "raw" data-set gathered from the platform will be divided in three types:

user_reads: these messages concern direct requests from the application or the platform to a specific device in order to obtain the value of one of its properties. These requests might be triggered by the user, for example opening the application, or by the rules module if it needs to know a property at a certain time.

user_writes: these messages are instructions, generated by the user or a rule, to a device in order to change the value of one of its properties. They are very similar to the *user_reads* messages, however these have some fields more dependent on the schema of the message. For example, if the message is for an RGB light bulb to change its color than it will have the additional r, g and b fields.

device_updates: these messages are quite different from the other types, they are sent throughout the platform and inform the concerned entities, a user application for example, about the value of a device property. Because they inform the value, these messages have, like the *user_writes*, more fields depending on the type of schema.

Regardless of type, a message will have ids and names related to the device hierarchy that serve as a way to organize the devices and their components to be able to manage them. These are: profile id, linking to a brand of light bulbs for example, and component id, one of the specific light bulbs sold in a set, and property id, in this case the brightness of a specific bulb. The message also contains a timestamp field.

B. Plugins developed

The provided anonymized datasets are then processed by plugins that extend an internal program suited to work with this data created for this project. These will generate information for the machine learning algorithms.

devicesPerUser: this plugin iterates through user_reads and writes datasets and creates information per user about its devices, such has: number of devices, number of components, the complete device hierarchy of that user, different schemas

used by these devices. For this it uses the ids present on the messages

loadEventsToDB: the main idea of this plugin is to insert into a database all of the events of all types of datasets, so that it is easy create other information in other plugins by query. The algorithm also maps the *device_updates* messages to anonymous users using information from the *devicesPerUser* plugin, since these messages have no information about user. It registers action times using the message timestamp and the user time-zone, if available. And marks which update occurred without a previous *user_write* message, because this indicates that either it changed by itself, ex: the value of a thermometer, or it changed because of a manual interaction. This is done by comparing update messages to previous write messages within a time-frame.

presenceFromDevices: this plugin is responsible for collecting events that can give us information about presence. This information is a mixture of heuristics, for example *device_updates* from light bulbs that were independent, this probably means that someone is home (notice that the detected presence can be someone that is not the application owner). This also collect other sensor information, for example cameras with motion sensors. The events are divided in two types: continuous, gathered from devices like alarms whose state provides a continuous source of presence information, and non-continuous like motion sensors or light bulb interactions which have a time window, heuristically defined, in which the occurrence of the event is relevant.

interactionStatsPerUser: this plugin is responsible for creating usage statistics for each user with different levels of granularity, taking into account the type and time of day. The metrics are created by iterating through the stored messages and counting events and time passed, then averages are calculated using this information. Examples of created metrics are: average number of update messages per hour, average number of write messages per weekend day, average number of independent update messages per hour on a weekday.

createMLDatasetGenStats: this plugin generates more immediate metrics. It will also iterate through the stored messages and depending on the chosen mode it will either create an entry per message, or an entry per window of time. The metrics on this entry are for example: number of write messages in the past 5 minutes, number of independent updates in the past half hour. This algorithm also creates information about presence per entry by querying the *presenceFromDevices* collection for events relevant to the time of the entry.

createCSVsForPresence: this plugin uses the information generated previously to create csv files that will later be used with machine learning algorithms. It creates a line of csv per *createMLDatasetGenStats* plugin entry, writing in the file the information generated in this plugin plus other like the ones present in the interaction stats and *devicesPerUser*. It can create datasets with some variations such as: only writing in the csv lines of users with more than X devices, only using information presence from continuous sources, among other options.

IV. Creating the machine learning models

The first test was of a small anonymized dataset, gathered from 3 days of interactions. The generated csv size ranged from 2MB to 300MB depending on pre-processing options. This data was from roughly 900 users, but depending on the mode only that data from some users could be used.

The classifiers were trained in Weka [16] and the results obtained in this phase showed that the approach is interesting but more exploration was needed. Tests were made using regression, treating presence as a continuous values, and discretizing the presence value gathered in the plugins into two or more classes. An example result, gathering information only from continuous sources, with only two classes to predict, which are presence and unknown, using a j48 decision tree, the Weka implementation of the C4.5 algorithm [17], and 10-Fold cross validation as evaluation method, we were able to achieve overall 99% of correctly classified instances. But for the same case, using non-continuous sources for presence gathering the results for the presence class were below 50%. Feature selection algorithms were also used to eliminate most features, since with the described plugins over 300 features are generated but only 30 to 60 remained after applying feature selection filters. This step helped training the classifier faster, with better precision and to understand what information was more useful.

After this phase, a larger version of the problem was tackled and at this point Weka was replaced by Scikit-Learn [18]

A. Scikit-Learn Experiments

The first part of the experiment (feature and parameter selection) uses Pandas [19] to read and transform the data to matrices supported by Scikit-Learn. The presence attribute is discretized to {0, 1} meaning unknown or presence. Feature selection is done choosing an algorithm from these: SelectKBest, with Chi2 [20] as the scorer function, LinearSVC [21] or a Random Forest [22], the last two cases are classifying algorithms which in their normal operation will rank the features in terms of their contribution to classification, so they can be used in this feature selection phase.

The next phase is parameter search, the classifier chosen, in this case a Random Forest [22], is fitted and tested to see which options for the classifier best suit the problem. This process also has 3 modes, the first is random parameter search, the second grid search and in the last both types of searches are used. Parameters are scored by testing the data with a 3-fold cross validation, and the best parameters are chosen. Finally the algorithms will be evaluated with a 5-fold cross validation for reference. This process is done with a small subset of the data due efficiency constraints.

The second part of the experiment uses the previous selected features and parameters to load a bigger part of the dataset and train and test on this new data. A 10-fold cross validation is used, generating the classification reports that will be presented in this paper.

The full dataset used contains 1 month of data, ranging, roughly, between 1—GB and 15GB, depending on the preparation in the plugins step. Differences in the pre-processing will be detailed in the next section.

B. *Experiments made*

Due to the way the instances are tagged, the experiments are mainly divided between excluding and not excluding the presence events in the calculated metrics. In the "excluding" scenarios events that were used to calculate presence information are not taken into account in the counts and averages calculated by the plugins. This is done using an identifier that uses profile, component and property id. So no brightness events of a light bulb will enter the statistics if the brightness information of this bulb was used to gather presence information for a user in the heuristic labelling step. But if, for example, the color-RGB property of this bulb wasn't used for presence labeling, the events related to this will enter the statistics. As stated before, the gathered presence value from the plugins was discretized to 0.0 or 1.0, and will be referred as unknown and presence from now on. This is a very strict procedure since it throws away information that, after pre-processing and merging with other information, can hardly be used to infer presence directly.

The first experiments yielded very good results, over 0.9 precision and recall was achieved for both classes, excluding and not excluding labeling information. However, after further exploration we realized the algorithm was using the metrics to detect each user almost individually and then classifying according to if the user has mostly presence tagged instances or unknown ones. This was possible because users with devices that contribute with continuous type information had much more presence tagged instances than the ones which only had non-continuous type information.

This led to the conclusion that for now the information gathered from each separated type of event: non continuous sources, ex: light bulbs, and continuous sources, for example: alarms shouldn't be all gathered in the same dataset. So the experiments were repeated with different types of datasets.

Results will now be presented stating the difference between them, depending on whether events were excluded or not, and which type of events were used to gather presence information from. These results are obtained with the process previously explained in the Scikit-Learn Experiments section with the difference that the parameter search phase was not done due to time constraints.

In the first new experiment events were not excluded and only presence gathered from non-continuous events was used:

TABLE I. CLASSIFICATION REPORT USING NON-CONTINUOUS EVENTS FOR LABELING NOT EXCLUDING ANY INFORMATION FROM METRICS

classes	precision	recall	f1-score	support
unknown	1.0	1.0	1.0	4979672
presence	0.87	0.49	0.62	20330
avg/total	1.0	1.0	1.00	5000002

TABLE II. CONFUSION MATRIX FOR THE SAME SCENARIO OF TABLE I

actual classes	classes predicted	
	unknown	***presence***
unknown	4978142	1530
presence	10427	9903

As with the initial experiments, we have now removed the events that were used to collect presence information from the statistics and repeated the experiment with the previous dataset:

TABLE III. CLASSIFICATION REPORT USING NON-CONTINUOUS EVENTS FOR LABELING AND EXCLUDING INFORMATION USED FOR LABELING FROM METRICS

classes	precision	recall	f1-score	support
unknown	1.0	1.0	1.0	4980543
presence	0.54	0.14	0.22	19459
avg/total	0.99	1.0	1.00	5000002

TABLE IV. CONFUSION MATRIX FOR THE SAME SCENARIO OF TABLE III

actual classes	classes predicted	
	unknown	***presence***
unknown	4978246	2297
presence	16726	2733

Then the same experiment was done but this time a minimum of devices, schemas and interaction per day was enforced. And then a maximum limit of these features was used. So only users who fulfilled these conditions were in the dataset. This tested if the model would behave better when there was information from more or less devices available. The results weren't better, recall dropped to 0.12 and 0.13 respectively. Precision however climbed to 0.60 in the case of the maximum devices, which suggests the algorithm is more certain of presence cases with less devices.

In order to use the rest of the presence information, datasets were created separately for users who had continuous information:

TABLE V. CLASSIFICATION REPORT USING CONTINUOUS EVENTS FOR LABELING AND NOT EXCLUDING ANY INFORMATION FROM METRICS

classes	precision	recall	f1-score	support
unknown	0.72	0.62	0.67	412795
presence	0.83	0.89	0.86	884421
avg/total	0.80	0.80	0.80	1297216

TABLE VI. CONFUSION MATRIX FOR THE SAME SCENARIO OF TABLE V

actual classes	classes predicted	
	unknown	***presence***
unknown	257197	155598
presence	100776	783645

These experiments were then reproduced excluding the events used to gather presence information, but results were very similar. Finally, we experimented a set with users who had both types of information and this experiment achieved good results, similar to the previous presented in table V.

After this, all these variations were tested again but with the difference that each line of the csv was created per event instead of using a time window. In the non-continuous dataset this approach lowered the scores overall but raised recall in some cases. For the continuous cases this approach also lowered the obtained scores. Finally, for the dataset of users who had both types of information, the results remained practically the same.

The number of users present in each dataset varied greatly, with the non-continuous having approximately 1200 users depending on the case, and the continuous 200. The difference in number of instances of each class was also very big, mostly just the continuous dataset had more lines of the presence class.

In order to cope with these differences between number of instances of each class, new experiments for the most interesting cases were done adding an additional random down-sampling phase to the process in order to have balanced datasets. Scores improved significantly. Results for the window-based, non-continuous information and excluding events follow:

TABLE VII. CLASSIFICATION REPORT FOR WINDOW BASED, USING NON-CONTINUOUS EVENTS FOR LABELING, EXCLUDING INFORMATION USED FOR LABELING FROM METRICS, AND WITH RANDOM DOWN-SAMPLING

classes	precision	recall	f1-score	support
unknown	0.74	0.81	0.78	19459
presence	0.79	0.72	0.76	19459
avg/total	0.77	0.77	0.77	38918

TABLE VIII. CONFUSION MATRIX FOR THE SAME SCENARIO OF TABLE VII

actual classes	classes predicted	
	unknown	***Presence***
unknown	15762	3697
presence	5411	14048

The event based approach was also better with down sampling, but in this case recall for the presence class was around 0.5:

TABLE IX. CLASSIFICATION REPORT FOR EVENT BASED, USING NON-CONTINUOUS EVENTS FOR LABELING, EXCLUDING INFORMATION USED FOR LABELING FROM METRICS, AND WITH RANDOM DOWN SAMPLING

classes	precision	recall	f1-score	support
unknown	0.65	0.93	0.77	168323
presence	0.88	0.50	0.64	168323
avg/total	0.77	0.72	0.72	336646

TABLE X. CONFUSION MATRIX FOR THE SAME SCENARIO OF TABLE IX

actual classes	classes predicted	
	unknown	***presence***
unknown	157047	11276
presence	83495	84828

Finally, these classifiers trained using down sampling were tested with the full data for each case, without cross-validation, in order to see the full capabilities of this technique. In the window based sets recall values obtained were close to their counterpart experiments without down sampling training, but in precision for the presence class dropped to 0.08. Because of the nature of the problem, the low precision scores might be unknown class instances that actually correspond to presence cases but we simply don't have information to tag them as such. These experiments may suggest that different classes or ways to rate the classification could be explored.

Despite these differences in the datasets and the results obtained, the many experiments gave us an insight to the problem. As expected the algorithm as a hard time identifying possible presence for the non-continuous case. Even using information from events that were used to determine the user's presence and build the training-set's labels, is not a guarantee of a high accuracy, which means it probably makes sense to continue counting these in the metrics instead of handicapping so heavily the training data just to assure the model generalizes the rule.

Work so far has focused mainly on pre-processing and exploration of standard approaches to the problem with these experiments. We also intended to assess the impact of using all or parts of the information. Future work will aim at creating more sophisticated features and other ways to improve the prediction. The presented results were obtained not using parameter search for the classifiers, because of time constraints, so that extra step can also help. Also, the creation of a specific hand-labeled data set (provided by volunteers) could greatly improve the confidence in the results.

V. CONCLUSIONS

This paper presented a system for human presence detection in a generic ambient intelligence context. The system can be further improved but still it performs well in its current state and shows a promising capability for detection using unspecific data from several IOT devices. In order to create the system an extensive pre-processing phase was applied to anonymized data gathered from a real application (the Muzzley platform), generating metrics for each new example. Datasets were labeled with presence information inferred by heuristic methods. The main challenges of the project were the pre-processing needed, the diversity of devices that interact with the platform, the creation of adequate features and not having explicit presence information. Although many results obtained have a low precision and recall for the presence class, the approach described provides insights to build upon. Presence detection, and, possibly in the future, activity detection, or household

number determination, are very important challenges to ambient intelligence and hopefully this project will help shed a little light on the current possibilities with these devices and technologies.

ACKNOWLEDGMENTS

We thank everyone at Muzzley for the access to the data and their continuous help in interpreting the properties and metrics.

REFERENCES

[1] C. Zelkha, Eli; Epstein, Brian; Birrell, Simon; Dodsworth, "From devices to 'ambient intelligence," Digit. Living Room Conf., 1998.

[2] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: technologies, applications, and opportunities," Pervasive Mob. Comput., vol. 5, no. 4, 2009, pp. 277–298.

[3] E. H. L. Aarts and J. L. Encarnacão, "True Visions: The Emergence of Ambient Intelligence," Springer, 2006.

[4] F. Mattern and C. Floerkemeier, "From the internet of computers to the internet of things," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6462 LNCS, 2010, pp. 242–259.

[5] Lopez Research, "An introduction to the internet of things (iot)," Lopez Res. Llc, vol. Part 1 , pp. 1–6, November 2013.

[6] N. Gershenfeld, R. Krikorian, and D. Cohen, "The internet of things," vol. 291, 2004, no. 4.

[7] G. Santucci, "The internet of things: between the revolution of the internet and the Metamorphosis of objects," Forum Am. Bar Assoc., 2010, pp. 1–23.

[8] C. Reports, "Reaping the benefits of the internet of things," no. May 2014.

[9] D. Evans, "The internet of things - how the next evolution of the internet is changing everything," CISCO white Pap., pp. 1–11, April 2011.

[10] J. C. Augusto and D. Shapiro, "Advances in ambient intelligence: volume 164 frontiers in artificial intelligence and applications," IOS Press Amsterdam, The Netherlands, 2007.

[11] D. J. Cook and S. K. Das, "How smart are our environments? An updated look at the state of the art," Pervasive Mob. Comput., vol. 3, no. 2, 2007, pp. 53–73.

[12] G. J. H. Patrício, "Redes sem fios de microcontroladores com acesso remoto aplicado à Domótica," Cad. Saude Publica, 2009.

[13] R. Collingridge, "Dream green house," 2009. [Online]. Available: http://dreamgreenhouse.com/index.php. [Accessed: 28-Dec-2015].

[14] R. Collingridge, "Dream green house: occupancy & presence," 2013. [Online]. Available: http://www.dreamgreenhouse.com/projects/2013/presence/index.php. [Accessed: 28-Dec-2015].

[15] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The gator tech smart house: a programmable pervasive space," Computer (Long. Beach. Calif), vol. 38, no. 3, 2005, pp. 50–60.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The Weka data mining software," SIGKDD Explor. Newsl., vol. 11, no. 1, 2009, p. 10.

[17] J. R. Quinlan, "C4.5: programs for machine learning," San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: machine learning in Python," Mach. Learn. vol. 12, 2012, pp. 2825–2830.

[19] W. McKinney, "Data structures for statistical computing in Python," Proc. 9th Python Sci. Conf., vol. 1697900, no. Scipy, 2010, pp. 51–56.

[20] R. L. Plackett, "and the Chi-squared Test," vol. 51, 1983, pp. 59–72.

[21] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 1992, pp. 144–152.

[22] Tin Kam Ho, "Random decision forests," Proc. 3rd Int. Conf. Doc. Anal. Recognit., vol. 1, 1995, pp. 278–282.