On-line Inference Comparison with Markov Logic Network Engines for Activity Recognition in AAL Environments

Manuel Fernandez-Carmona, Nicola Bellotto
Lincoln Centre for Autonomous Systems (L-CAS)
University of Lincoln, LN6 7TS, UK
Email: {mfernandezcarmona, nbellotto}@lincoln.ac.uk

***Abstract*—We address possible solutions for a practical application of Markov Logic Networks to online activity recognition, based on domotic sensors, to be used for monitoring elderly with mild cognitive impairments. Our system has to provide responsive information about user activities throughout the day, so different inference engines are tested. We use an abstraction layer to gather information from commercial domotic sensors. Sensor events are stored using a non-relational database. Using this database, evidences are built to query a logic network about current activities. Markov Logic Networks are able to deal with uncertainty while keeping a structured knowledge. This makes them a suitable tool for ambient sensors based inference. However, in their previous application, inferences are usually made offline. Time is a relevant constrain in our system and hence logic networks are designed here accordingly. We compare in this work different engines to model a Markov Logic Network suitable for such circumstances. Results show some insights about how to design a low latency logic network and which kind of solutions should be avoided.**

I. Introduction

Ageing population has arisen new challenges in healthcare systems [1]. Among other concerns, chronic illnesses have a huge social and economic impact on modern societies. Addressing these conditions in their early stages aims to extend user autonomy, avoid acute care expenses, delay user institutionalization an thus increase their well-being. Also, adapting a dwelling to fit patient needs is preferable [2] from a medical point of view, because it eases the psychological impact of illness.

Patients with mild cognitive impairments are functionally independent. This means they usually cope with their Activities of Daily Living (ADLs): such as grooming, cooking or toileting. They may have occasional episodes of disorientation, memory loss or cognitive difficulties [3]. Activity Recognition (AR) is a valuable tool to effectively identify ADLs, so that experts can track anomalies in users condition while keeping as much of their autonomy as possible.

During this monitoring, it is important to minimize changes on patient lifestyle. Domotic sensors are a good choice for this task: they are cheap, pervasive and unintrusive. A domotic sensor is an electronic measurement device typically used to provide information to home appliance control systems. They are designed for mass production and as part of everyday home appliances. On the other hand, they provide a limited amount of information. Despite the wide range of different manufacturers and protocols, they usually are not open.

In the best case, we will be dealing with partial and incomplete information. Markov Logic Networks (MLNs) allow to perform reasoning based on such kind of observations [4]. At the same time they offer a formal knowledge representation, which can easily be used by experts to outline new elements. These advantages have already been addressed by other authors. Nevertheless, they mostly focus on offline analysis of recorded datasets.

The main contribution of this paper is to provide a comparison of three different MLNs frameworks that provide AR inferences about on-going activities. These are evaluated in the context of the ENRICHME[1] project, which proposes an Ambient Assisted Living (AAL) environment for users with mild cognitive impairments that interacts through a robot companion. The latter can also be seen as a mobile sensor, capable of improving AR with ad-hoc information. To be effective, however, it must rely on prompt ambient sensors based inferences. Furthermore, in our evaluation we stress out the impact of network complexity on the inference computing times.

The reminder of the paper is organized as follows. We present different approaches to AR and MLN in Section II. A brief theoretical introduction to MLNs is given in Section IV. We also make a short survey on different MLN engines. Our inference system is part of an AAL system described in Section III. In order to meet its requirements, in Section V we describe a set of experiments comparing different frameworks. Lastly, conclusions on outcomes from each framework are depicted in Section VI.

II. Related Work

Requisites like modularity, distribution and invisibility have been identified by [5], [6] as key elements in Smart Homes Design. Domotic sensors easily cover these requisites, making them a popular solution in AAL. European funded projects like SWEET-HOME [7], LsW [8] or Aladin-e [9] already

[1]ENabling Robot and assisted living environment for Independent Care and Health Monitoring of the Elderly (ENRICHME) - http://www.enrichme.eu

978-1-5090-4056-8/16 $31.00 © 2016 IEEE
DOI 10.1109/IE.2016.29

take advantage from this kind of sensors in AAL. Most domotic sensor protocols support wireless interfaces. Wireless sensors can be easily deployed on pre-existing houses, usually with a lower cost and overall impact. For instance, the CASAS project [10] relies on wireless sensors to achieve an "easy-to-install" smart home with AR capabilities. They use a Support Vector Machine (SVM) system to acquire knowledge of the environment status, user interests, habits and capabilities.

MLNs are also widely adopted in AR for AAL projects. They were first proposed by Richardson and Domingos [4] to combine both probabilistic and logical reasoning. A MLN can be described as a set of weighted first-order logic formulas or clauses. These clauses describe a knowledge base which is able to deal with imperfect and contradictory evidences. This flexibility is provided by their respective clause weights. They describe probability of occurrence, thus providing additional tolerance to their logic.

Typically, MLNs have been used for vision based AR [11], [12]. They are a very flexible tool, e.g., they have been extended to support event calculus to model vision based activities with some degree of temporal inertia [13].

In [14], authors make an extensive use of domotic sensors to detect ADL using MLN, supported by acoustic information gathered by ambient microphones. They statistically process sensor outputs before generating network evidences. This is a key step with high influence on results of the AR. Hybrid approaches address this step using Neural Network (NN) [15] or a Hidden Markov Model (HMM) [16] to generate evidences for a MLN.

Online reasoning is another relevant issue in AR for AAL. A responsive system needs to be aware of current user needs, which will likely depend on ongoing activities. The CASAS project [10] performs this with real-time activity labeling, while Chen et al. [17] do online AR using a sliding window to support an ontology-driven approach.

Although formal methods exist to model situations in the healthcare domain [18], in the current work there is no need to identify and describe specific activities, since ADL are widely accepted and used since first proposed in [19].

We consider a system that uses MLNs to support *online* reasoning. This constrain limits the complexity of the MLN, so it must be taken into account.

III. System overview

In most of AAL projects, sensor selection and deployment define what can be detected and where. Sensor locations remain fixed once the system is running, limiting possible sensing reconfigurations. Uncovered areas may affect the ability to detect and react to some situations, which undermines system usability.

The ENRICHME project proposes the introduction of a robot companion into the AAL environment. The robot interacts with the user and, at the same time, provides additional sensing information about user activities. In this way, sensing can be adapted to focus always on the user. The robot sensing capabilities range from human tracking information to physiological data, multimodal interaction and quality of the living environment (e.g. light, particles in the air, temperature). In order to provide this information, the robot makes use of advanced sensing devices such as thermal camera, RGBD camera, environmental sensor and RFID reader. RFID tags provide information about relevant items being moved or used, which are related to user activities.

The robot can also make use of the distributed sensing environment to improve its behaviour with additional data. This distributed sensing environment consists in commercial wireless domotic sensors (e.g. presence detectors, contact sensors, temperature sensors, etc.). Using both sensing sources, the ENRICHME system can perform long-term human behaviour analysis. This analysis will extract information about typical behaviour patterns and anomaly detection to health professionals, who will assess the evolution of the user's cognitive impairment.

An Ambient Intelligence Server (AIS) is in charge of gathering all sensor information coming from the robot and the domotic network. It is located on a dedicated embedded computer and it shares a common communication and storage interface with the robot. Domotic sensor information is captured using an abstraction layer based on OpenHAB [20]. This software allows to abstract sensor values from specific domotic protocols or vendors. OpenHAB can simultaneously integrate a broad range of different kinds of domotic network technologies into an uniform interface. It also offers sensor information recording services through different databases. In our case, we use a MongoDB[2] database for persistent storage. Both OpenHAB and MongoDB are interoperable with the Robot Operating System[3] (ROS) running on the robot, making possible the exchange of information with the latter. Sensor information flow from its different sources to the database, as shown on Fig. 1. This feature allows the robot companion to have full access to ambient sensor records and interact with the domotic network.

The inference engine used for AR also resides on the AIS. It uses the database to build evidence, so robot sensor information is also available for inference. Both elements benefit from this common infrastructure.

IV. Markov Logic Networks

This section describes briefly the main concepts behind MLNs, and introduces the MLN engines evaluated in the experiments.

A. Concepts and Theory

A MLN can be described as a collection of pairs $L = (F_i, w_i)$, where F_i is a clause and w_i its weight. Clauses are first-order logic formulas, composed by constants, variables, functions, and predicates.

- *Constants* are possible objects in the domain. In our case, one of the domains is `activity`, and

[2]http://www.mongodb.org/
[3]http://www.ros.org/

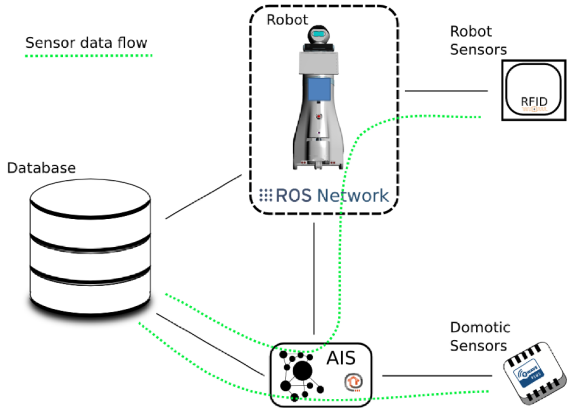

Fig. 1. Sensor information flow in ENRICHME.

possible constants on it are `cooking`, `toileting`, `resting`, `etc`.

- *Variables* describe a set of objects in a domain.
- *Functions* establish mappings between different objects.
- *Predicates* define logical attributes or relationships over domain elements (e.g., `currActivity(x)` or `isLevel(High,temperature)`). Predicates can be combined into more complex formulas using logical connectors ($\Rightarrow, \wedge, \vee, \Leftrightarrow$).

Functions, variables and constants are called *terms*. If they do not contain variables, they are *ground terms*. A predicate that contains only ground terms is a *ground predicate*. When a logical value is assigned to all grounded predicates in a network, we have a *possible world*.

As a result, using a finite set of constants $C = \{c_1, c_2, \ldots c_N\}$ applied over clauses F_i in a MLN, we can define a Markov network $M_{L,C}$. This network is an undirected graph with a binary node for every ground predicate. Ground predicates appearing in the same formula are linked, forming a *clique*. Depending on the given constants, each formula may have more than one clique associated to it. Each clique has an associated logic feature and weight, provided by the evaluation of the clause and its weight.

Probability distribution over a possible world x described by a Markov network can expressed as:

$$P(X = x) = \frac{1}{Z} exp\left(\sum_i w_i n_i(x)\right) \quad (1)$$

where $n_i(x)$ is the number of true features corresponding to clause i in the possible world x, and Z is a normalization factor $Z = \sum_x exp\left(\sum_i w_i n_i(x)\right)$.

In essence, MLNs are a compact way to produce Markov networks. Once all clauses are applied over a domain, a MLN is grounded into a Markov network, and this network describes the probability of all possible combinations of grounded clauses.

Inference in Markov networks is usually done using an approximate method, called MC-SAT, which is an extension of the Markov Chain Monte Carlo methods proposed in [21]. This method is available on all the considered engines. Note also that this is a #P-complete problem, therefore exact methods are usually intractable.

B. Inference Engines

The networks here considered have been implemented using three different MLN engines. Their key features are summarized in Table I.

The first and most complete MLN software is Alchemy [22]. It is an open source package, developed in C++, which offers several methods for inference, weight learning, as well as very flexible syntax for formulas.

ProbCog [23] has a very flexible syntax for clauses too. Internally it also makes a conversion to conjunctive normal form (CNF) logic clauses. This open source software is based on a python MLN toolbox called PyMLN. It has additional modules, developed in Java, for Bayesian Networks, and it offers a ROS package to allow queries from other ROS nodes.

Tuffy [24] is an open source inference engine developed with Alchemy as a reference, but scoping to scalable systems. Tuffy supports only disjunctive clauses, but it shares most of its syntax with Alchemy. It is programmed using Java and uses PostgreSQL to store intermediate data.

Engine	Alchemy	Tuffy	ProbCog
Language	C++	Java	Python/Java
Rule Syntax	Any	CNF	Any
MAP Inference	MaxWalkSat	WalkSAT, SweepSAT	MaxWalkSAT
Marginal inference	Belief Propagation, MC-SAT, Gibbs sampling, Simulated tempering	MC-SAT	MC-SAT, Gibbs Sampling, Enumeration
Weight learning	Voted Perceptron, Conjugate Gradient, Diagonal Newton, Generative	Diagonal Newton	Maximum pseudo-likelihood, Maximum likelihood

TABLE I
MLN ENGINES KEY FEATURES

V. Experiments

Following the description of our system requirements and the foundations of our inference engine, we present a set of experiments in order to identify and compare the relative speed of different MLN engines.

A. System Set-up

A test scenario was deployed in the L-CAS[4] facilities. It comprised three typical ADL locations: kitchen, living room

[4]Lincoln Centre for Autonomous Systems - http://lcas.lincoln.ac.uk/

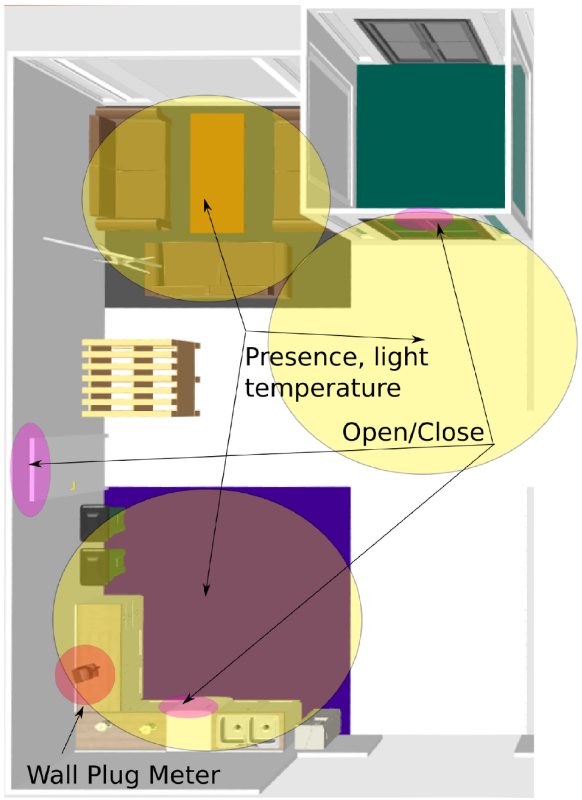

Fig. 2. L-CAS domotic sensor setup.

and entrance area. On these premises, human presence, light and temperature were monitored. In addition, door contact sensors where placed on main, toilet and fridge's doors. Finally, a power consumption sensor was placed on the coffee machine to check its usage. Fig. 2 illustrates the sensor deployment and the approximate coverage areas.

We used commercial ZWave wireless domotic sensors, some of which are shown in Fig. 3, which were provided by the Fibar Group[5] for the ENRICHME project. This kind of sensors are easy to deploy, widely available and have a long battery life. Their reduced size makes them also very unintrusive. All processing was made using the same machine, a desktop PC i5-3570, with 8 GB of RAM and running a Linux OS Ubuntu 14.04 64 bits.

B. Evidence Building

Our AR system queries the sensor database to develop specific evidences for the MLN. Other authors suggested that a reasonable time to detect an activity is approximately 1 minute [25]. Therefore, in our system the sensor activity is checked every minute and discretized into a finite set of states.

Temperature and light sensors are compared against its average value in the database. Their average values during the last minute are discretized into three ranges:

- Low (under 75% of the average value)

[5]http://www.fibaro.com/

Fig. 3. ZWave domotic sensors.

- Mid (between 75% and the average value)
- High (above the average value)

The power consumption sensor readings are discretized against the typical power required to perform a cup of coffee. The binary sensors are discretized counting the number of activations (i.e. $Off \rightarrow On$ transition) in the last minute:

- Low (no activations)
- Mid (one activation or already active)
- High (more than one activation)

These discrete levels merge different sensors into a common domain of sensor values. Under this domain, a *Low* sensor value will have a common meaning, no matter which sensor it came from, allowing the use of identical logical terms for all of them. Our other domains are *sensor*, which identifies information source and *activity* for the detected action.

We tested all the engines using the same evidence. We built ten evidence sets over a 10 minutes period, which was chosen for its complexity and variety. For example, on the first time slot, the activity to be recognized was "cooking in the kitchen". In this particular scenario, some movement was detected in the entry zone, but there was no activation of the door sensor, so it was difficult the trigger the recognition of the activity. Other evidence sets were empty, or represented simultaneous activities. Fig. 4 shows some snapshots captured during this evidence building, originally gathered for benchmarking spatio-temporal models of changing environments in [26].

It could be argued that our reduced dataset may bias our tests. This is, a bigger dataset may present a different trend on inference times. In order to simulate a larger dataset, we artificially created a bigger one. This dataset was built randomizing evidences provided to engines. Sensor activity was given a random value rather than a level depending on domotic sensor status.

As a result, evidence sequences are nor consistent or meaningful but arbitrary long. Inference results from these queries are also not to be taken into account. However, inference times are still relevant for our benchmarking.

Fig. 4. Snapshots captured during evidence generation for MLN inference.

C. Proposed Networks

In order to measure each MLN performance, three different setups have been chosen to analyse increasing levels of complexity, and tested:

- *Network 1 (N1)* – sensor based network: each sensor is related to an activity;
- *Network 2 (N2)* – sensor/time based network: time is included as a new domain to describe activities;
- *Network 3 (N3)* – sensor/time based network with inertia: activities depend both on current and past sensor values.

Each network includes additional domains and predicates, increasing the complexity of the inference process. More domains implies new groundings an thus an increase in the number of possible worlds. Similarly, introducing additional predicates, which link different domains, contributes to this increase. Effectively, each network adds complexity to the overall inference process.

The above setups have been implemented using the three different MLN engines described in Sec. IV-B: Alchemy, ProbCog and Tuffy. The clause weights learning was performed using a pseudo-log likelihood algorithm. Each engine has some differences in its syntax that must be taken into account. Tuffy does not allow the use of implications ($\Rightarrow$) or inclusions ($\wedge$) in clauses, so we have restricted all the functions to disjunctive clauses. Also, Alchemy does not support domain declaration in the network description file. Apart from these minor differences, we used the same network configuration for all the engines.

The first network configuration maps sensor values with a particular activity. However, since several activities might be happening simultaneously, such evidences do not provide reliable information.

The second network configuration includes time as an additional domain. Activities happen within a given time slot. We used the temporal extension of MLNs proposed by [27] to model time slots. This network's test was performed with small sets of timeslots to limit the network complexity by introducing a high number of constants.

The complexity of the system could be varied by increasing the number of sensors and rooms, expanding the number of constants in their respective domains. However, a similar effect can be obtained by with the size of the considered time domain, which can be arbitrarily constrained by considering different numbers or lengths of time slots. The latter has the advantage of not requiring additional resources or datasets with different networks.

The third network included also more complex activity descriptions. For instance, the *entering* activity was modelled by an activation of the presence detector at the entrance, *preceded* by an activation of the main door's contact sensor. An evidence was selected for each network.

D. Results

Using the aforementioned common network configurations, each engine was tested 50 times on both datasets. The results of our comparison are illustrated in Fig. 5, which shows the average completion time and standard deviation for all the engines and networks: black for network N1, grey for N2, and light grey for N3. As expected, the computing time was higher as the complexity of the network increased (light grey bar).

Alchemy clearly outperforms the other engines. Even N3's inference time is lower on Alchemy than the average inference time on Tuffy for N1. Its standard deviation is also very low, which implies a highly predictable inference time.

ProbCog perform reasonably well in terms of average times, but the degradation of the deviation times is worse. Its inference time ranges are therefore bigger.

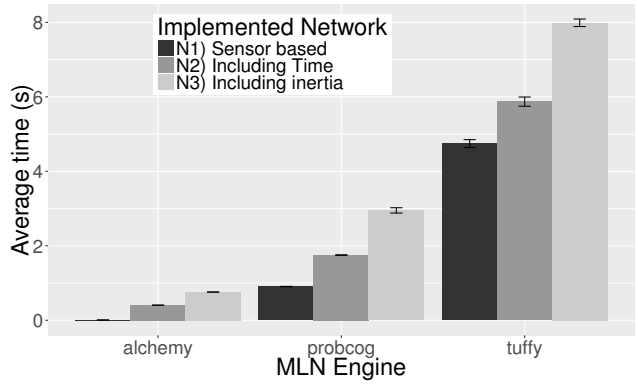

Fig. 5. MLN engines inference times comparison.

Tuffy has the worst performance of all the engines considered in these tests. Its inference times are constantly bigger than the ProgCog worst time results.

We have also obtained time metrics from each query, presented in Fig. 6. Each network has a different colour and symbol: red circle for N1, green triangle for N2 and blue square for N3. Again, more complex networks (N2, N3) have longer inference times, independently from the query. In all the considered situations, Alchemy's inference time -first set of plots- is almost independent from the given evidence and network. Also, ProbCog's inference times are evidence-independent too, but only for networks N1 and N2. Its performance decreases considerably for network N3, showing not only longer times, but also higher deviations, depending on the query. These effects are even more evident for the Tuffy engine, where a given query may affect remarkably the respective inference times.

Network	Engine	Average inference time (s)	
		gathered evidences	randomized evidences
N1	Alchemy	0.01	0.01
	ProbCog	0.90	0.97
	Tuffy	4.75	3.53
N2	Alchemy	0.41	0.39
	ProbCog	1.75	1.89
	Tuffy	5.87	4.46
N3	Alchemy	0.76	1.40
	ProbCog	2.95	3.52
	Tuffy	7.99	7.21

TABLE II
INFERENCE TIME COMPARISON BETWEEN GATHERED AND RANDOMIZED EVIDENCES

Regarding random evidence experiments, random sensor activity level shows very similar results compared with our previous tests. Table II compares inference times between gathered and randomized evidences. Alchemy is still the fastest engine in all cases, followed by ProbCog and Tuffy.

However, if we compare results within the same network and engine, there is some differences between both datasets. Tuffy engine significantly increases its performance using randomized inputs with all network configurations. In contrast, ProbCog performance is worse using randomized datasets than using gathered evidences with all network configurations. Its inference time increases more than a 7% with any network. Finally, Alchemy presents none or little change with networks N1 and N2, but a significant increase in its average times with network configuration N3. It almost doubles its inference time after randomizing the evidences. Hence, engine comparison with our datasets can be considered as a reasonable approach, but not as a reliable metric of each network complexity.

VI. CONCLUSIONS

In this paper we introduced and compared the responsiveness of different MLN engines, in the context of our ENRICHME monitoring system for elderly with mild cognitive impairments, to check if these tools can address the required needs for online inference. All the considered engines were able to process the proposed networks. Their execution times were all under 10 seconds, which unsurprisingly increase with the complexity of the network.

In order to overcome the limitations of our dataset, we also run the experiments using randomized evidences. They showed similar trends regarding to engines comparison, but average inference times themselves presented some differences. An specific dataset does have influence on the average inference times of a MLN, no matter the selected engine. Tuffy engine performed better with these inputs than with our dataset. Therefore, our results are useful to compare different engines, but analysis of a network implementation under an specific engine does require bigger datasets for validation.

In general, Java based engines were the ones performing worst. Since we are dealing at the moment with small inference problems, the Java Virtual Machine causes a significant overload compared with the total inference time. It also makes use of databases to store temporal information, which is an additional overload justifiable only on larger and more complex systems. Obviously, Tuffy is optimized for different kinds of MLNs.

From these preliminary results, Alchemy seems to have the best response of all, followed by the ProbCog engine. ProbCog is an interpreted code in Python, which has its impact on performance in this case. Alchemy is the only engine relying on a compiled code and, as a result, it is clearly faster. Ironically, using some of their features could have decreased both engines performance. Indeed, they allow the use of implications ($\Rightarrow$) and inclusions ($\wedge$) in clauses, but we have restricted our implementation to disjunctive clauses only to ensure that all engines used the same formulas. This had a positive impact on Alchemy and ProbCog' speeds, thanks to the avoidance of an internal clause conversion stage. A good selection of network clauses has a great impact on the overall computing times.

This paper presented essential but still preliminary results, which must be extended to include more complex scenarios and additional quality metrics. Labelled datasets with associated images will be used to establish a baseline for our future networks. Also, comparisons with other datasets will be carried out to prove network validity and reliability.

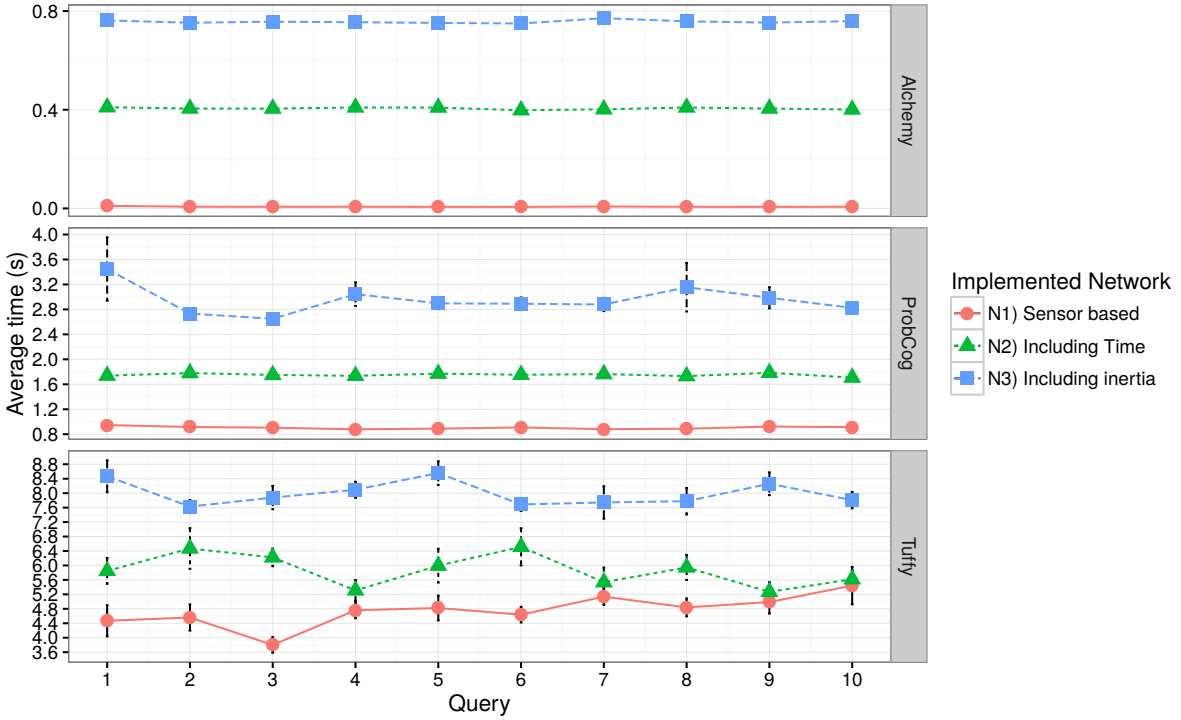

Fig. 6. MLN engines inference time comparison among different networks.

Finally, massive datasets with enough variability could reveal input combinations where engines perform differently, as our experiments with randomized input suggest. On-line performance of MLN engines is a challenging topic that still requires further analysis.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EC H2020 Programme under grant agreement No. 643691, ENRICHME. Authors also want to thank Fibar Group for providing the wireless domotic sensors for this work.

REFERENCES

[1] B. Rechel, E. Grundy, J. M. Robine, J. Cylus, J. P. MacKenbach, C. Knai, and M. McKee, "Ageing in the European Union," *The Lancet*, vol. 381, no. 9874, pp. 1312–1322, 2013.

[2] K. Van Haitsma, K. Curyto, A. Spector, G. Towsley, M. Kleban, B. Carpenter, K. Ruckdeschel, P. H. Feldman, and M. J. Koren, "The preferences for everyday living inventory: scale development and description of psychosocial preferences responses in community-dwelling elders." *The Gerontologist*, vol. 53, no. 4, pp. 582–95, aug 2013.

[3] R. C. Petersen, B. Caracciolo, C. Brayne, S. Gauthier, V. Jelic, and L. Fratiglioni, "Mild cognitive impairment: a concept in evolution," *Journal of Internal Medicine*, vol. 275, no. 3, pp. 214–228, 2014.

[4] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[5] M. Amiribesheli and A. Bouchachia, "Smart Homes Design for People with Dementia," *2015 International Conference on Intelligent Environments*, pp. 156–159, 2015.

[6] R. Li, B. Lu, and K. D. McDonald-Maier, "Cognitive assisted living ambient system : a survey," *Digital Communications and Networks*, vol. 1, no. 4, pp. 229–252, 2015.

[7] M. Vacher, F. Portet, P. Chahuara, S. Caramihai, C. Munteanu, I. Mocanu, and S. Mocanu, "Indoor Personal Monitoring , Supervising and Assistance Sweet-Home and AmiHomeCare case studies," *Journal of Control Engineering and Applied Informatics*, vol. 16, no. 1, pp. 50–61, 2014.

[8] F. Müller, P. Hoffmann, M. Frenken, A. Hein, and O. Herzog, "Lsw: Networked home automation in living environments," *Ambient Assisted Living, Advanced Technologies and Societal Change*, pp. 19–24, 2014.

[9] E. Maier and G. Kempter, *ALADINa magic lamp for the elderly?*, Berlin, Heidelberg, 2010, p. 12011227.

[10] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A Smart Home in a Box," *Computer*, vol. 46, no. 7, pp. 62–69, jul 2013.

[11] M. Beetz, M. Tenorth, D. Jain, and J. Bandouch, "Towards automated models of activities of daily life," *Technology and Disability*, vol. 22, no. 1-2, pp. 27–40, 2010.

[12] G. Cheng, Y. Wan, B. P. Buckles, and Yan Huang, "An Introduction To Markov Logic Networks and Application in Video Activity Analysis," in *International Conference on Computing, Communication and Networking Technologies*, Hefei,, 2014.

[13] A. Skarlatidis, G. Paliouras, and A. Artikis, "Probabilistic Event Calculus for Event Recognition," *ACM Transactions on computational Logic*, vol. 16, no. 2, 2015.

[14] P. Chahuara, A. Fleury, and M. Vacher, "Using Markov Logic Network for On-Line Activity Recognition from Non-visual Home Automation Sensors," *Ambient Intelligence*, pp. 177–192, 2012.

[15] K. S. Gayathri, S. Elias, and B. Ravindran, "Hierarchical activity recognition for dementia care using Markov Logic Network," *Personal and Ubiquitous Computing*, vol. 19, no. 2, pp. 271–285, 2015.

[16] F. J. Ordóñez, P. de Toledo, and A. Sanchis, "Activity Recognition Using Hybrid Generative/Discriminative Models on Home Environments Using Binary Sensors," *Sensors*, vol. 13, no. 5, pp. 5460–5477, 2013.

[17] L. Chen, C. D. Nugent, and H. Wang, "A Knowledge-Driven Approach to Activity Recognition in Smart Homes," vol. 24, no. 6, pp. 961–974, 2012.

[18] A. Coronato and G. D. Pietro, "Formal specification of wireless and pervasive healthcare applications," *ACM Transactions on Embedded Computing Systems*, vol. 10, no. 1, pp. 1–18, 2010.

[19] M. P. Lawton and E. M. Brody, "Assessment of older people: self-maintaining and instrumental activities of daily living." *The Gerontologist*, vol. 9, no. 3, pp. 179–186, 1969.

[20] L. Smirek, G. Zimmermann, and D. Ziegler, "Towards Universally Usable Smart Homes How Can MyUI , URC and openHAB Contribute to an Adaptive User Interface Platform ?" in *CENTRIC 2014 : The Seventh International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services*, no. c, Nice, France, 2014, pp. 29–38.

[21] H. Poon and P. Domingos, "Sound and Efficient Inference with Probabilistic and Deterministic Dependencies," in *AAAI'06 Proceedings of the 21st national conference on Artificial intelligence*, vol. 1, 2006, pp. 458–463.

[22] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, J. Wang, A. Nath, and P. Domingos, "The Alchemy System for Statistical Relational AI: User Manual," Department of Computer Science and Engineering. University of Washington, Seattle, WA., Tech. Rep., 2010.

[23] D. Jain, L. Mosenlechner, and M. Beetz, "Equipping robot control programs with first-order probabilistic reasoning capabilities," *2009 IEEE International Conference on Robotics and Automation*, pp. 3626–3631, 2009.

[24] F. Niu, C. Ré, A. Doan, and J. Shavlik, "Tuffy: Scaling up Statistical Inference in Markov Logic Networks using an RDBMS," *Proceedings of the VLDB Endowment*, vol. 4, no. 6, pp. 373–384, 2011.

[25] J. Rynkiewicz, "Hybrid hmm/mlp models for time series prediction," in *Proceedings of 7th European Symposium on Artificial Neural Networks (ESANN 1999)*, Bruges, Belgium, april 1999, pp. 455–462.

[26] J. M. Santos, T. Krajnik, J. Pulido Fentanes, and T. Duckett, "Lifelong information-driven exploration to complete and refine 4d spatio-temporal maps," *Robotics and Automation Letters*, 2016.

[27] M. Biba, S. Ferilli, and F. Esposito, "Modelling (Bio) Logical Sequences through Markov Logic Networks," in *International Joint Conference Intelligent Information Systems (IIS 2009)*, Warsaw, 2009, pp. 1–14.