Knowledge Reactor: A Contextual Computing Work in Progress for Eldercare

Scott N. Gerard, Aliza Heching, Susann M. Keohane, Samuel S. Adams
IBM Corporation
{sgerard,ahechi,skeohane,ssadams}@us.ibm.com

Abstract—The world-wide population of people over 60 years of age is growing rapidly. The explosion is placing increasingly onerous demands on individual families, multiple industries and entire countries. Current, human-intensive approaches to eldercare are not sustainable, but IoT and AI technologies can help.

The Knowledge Reactor ("KR") is a contextual, data fusion engine built to address this and other similar problems. It fuses and centralizes IoT and System of Record/Engagement data into a reactive knowledge graph. Cognitive applications and services are constructed with its multiagent architecture. The KR can scale-up and scaledown, because it exploits container-based, horizontally scalable services for graph store (JanusGraph) and pub-sub (Kafka) technologies. While the KR can be applied to many domains that require IoT and AI technologies, this paper describes how the KR specifically supports the challenging domain of cognitive eldercare.

Rule- and machine learning-based analytics infer activities of daily living from IoT sensor readings. KR scalability, adaptability, flexibility and usability are demonstrated.

Keywords—ambient sensing, AI, artificial intelligence, eldercare, IoT, internet of things, knowledge graph

I. Introduction

For the first time in history, older adults outnumber children under the age of five. By 2050, roughly 2 billion people will be 60 or older, and in countries like Japan and Singapore, it will be more than half of the population. Unfortunately, in many countries, there is a shortage of caregivers to go around. This is a long-term societal issue with far-reaching implications that transcend industries and borders.

This Knowledge Reactor ("KR") research project constructs a system from Internet of Things ("IoT") and Artificial Intelligence ("AI") technologies to address parts of the eldercare problem. These technologies will never fully replace humans in eldercare, but they can be used to automate tedious tasks, freeing caregivers to spend their time on problems humans alone can uniquely address. IoT is crucial because it enables round-the-clock, in-dwelling monitoring of elders. AI and analytics are crucial to intelligently convert the variety of sensor readings types into a coherent situational context. Only then can intelligent and well-informed decision making occur. The KR can be productively applied to other, non-eldercare domains, but this paper focuses exclusively on the eldercare domain.

Using ambient IoT sensor data, the research objective is to learn and model individual patterns for sleeping, eating, moving, grooming, toileting, and bathing and make connections

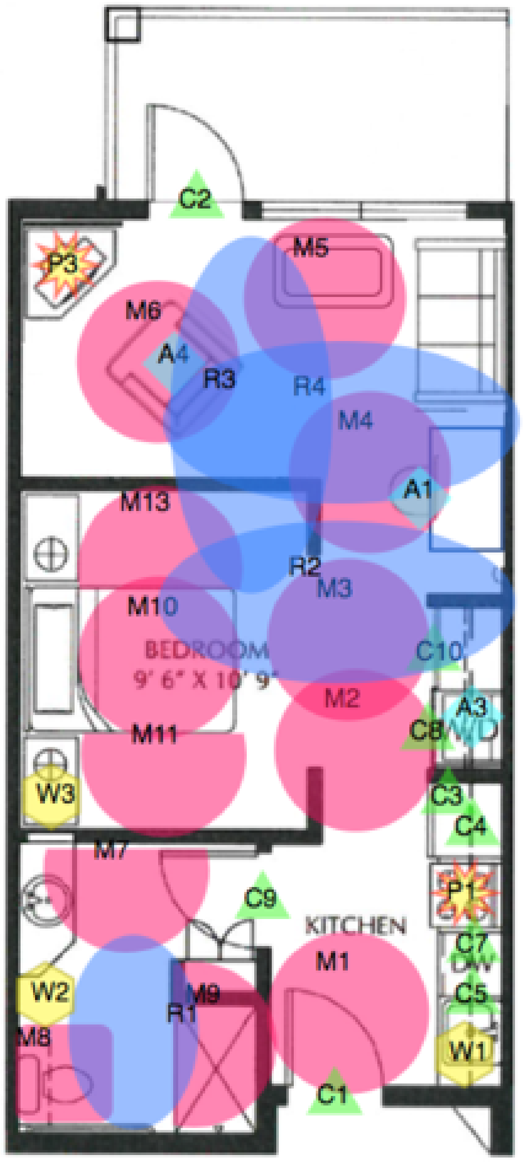

Fig. 1. Topview visualization of a studio apartment. Colored ovals are approximate sensing regions.

All authors work for IBM Research. Contact sgerard@us.ibm.com from certain behaviors to increased risks of health decline. The hypothesis is that the right combination of consumer grade sensors deployed in an optimal layout within a room or dwelling will allow daily patterns to be readily observed. The technical challenge is to enable this at scale and personalized for every older adult. The needs of caregivers to proactively monitor elders' health and well-being must be balanced against elders' privacy, security, and digital dignity.

Section II describes relevant background. Section III describes the use case that motivates the application of technology to support ongoing quality of life. Section IV presents the Knowledge Reactor, developed as the foundation for the eldercare solution. Section V describes the eldercare application using data gathered from two real-world environments. Section VI describes the analysis performed on the data from these two real-world environments and interim results. Section VIII summarizes our findings and includes ideas for future work.

II. Background

A common life trajectory for an elder begins with the elder living unassisted in her own home or apartment. After retirement, as her needs and abilities change, she might transition into an *independent living* ("IL") facility. IL is defined by freedom, choice, amenities, and convenient assistance for a range of needs. As her capabilities diminish further, she may transition to an *assisted living* ("AL") facility. AL allow elders to enjoy as much independence and activity as they would like, with the comfort of knowing on-site health care and personal assistance are available when needed. As a next step in the continuum of care, a *skilled nursing facility* ("SNF") is an inpatient healthcare facility that provides continuous (24-hour) nursing supervision to individuals who require medical care but do not require hospitalization. Commonly referred to as "nursing homes", these facilities normally care for incapacitated persons in need of long- or short-term care and assist with many aspects of daily living (walking, bathing, dressing, eating).

Activities of daily living ("ADLs") and instrumental activities of daily living ("IADLs") are important measures of an elder's physical or cognitive status. ADLs are basic self-care tasks such as toileting, bathing, and mobility. IADLs are more complex skills such as shopping, food preparation, and housekeeping. Together, ADLs and IADLs represent the skills required in order for an individual to live independently. ADLs and IADLs are often used by medical practitioners to provide a functional assessment of the individual. Toward this end, this research focuses on the use of sensor data to identify ADLs and IADLs.

III. Use Case

This research proposes the use of IoT and AI technologies to support elders as they transition through the continuum of care. It considers the deployment of sensor technology in an environment comprised of tens, hundred, or thousands of dwellings. Each dwelling (e.g., house, apartment, living facility), is home to typically one elder or two elders (e.g., a couple). Sensors monitor each elder in each dwelling to learn the elder's activity patterns. Because each individual is unique, pattern details are elder-specific, although one may identify cohorts of individuals with similar patterns.

While different types of sensors may be deployed in the environment, the use case focuses on the deployment of ambient sensors. Ambient sensors are used since many of today's elders prefer not to wear sensor devices. Even those who are not opposed to wearables cannot be depended upon to charge their wearables. This is particularly true for elders with cognitive decline or memory dysfunction – a group of elders who are particularly at risk and in need of assistive technologies. Of course, these concerns must be adapted as technology changes and as elders' comfort with technology changes.

Sensor data is gathered locally but a decision must be made whether data should be maintained and analyzed locally or centrally. Retaining all data locally minimizes the risk of a personal information data breach and minimizes network bandwidth. On the other hand, aggregating data centrally allows for data analysis across individuals and dwellings. Since sensors from some vendor ecosystems must pass through the vendor's cloud infrastructure, local analysis was not an option.

Elder-specific and dwelling-specific situations determine the number of sensors required to detect and establish a baseline and monitor changes in an elder's ADLs and IADLs. Section VI discusses the analysis of the sensor signals for six ADLs: toileting, bathing, cooking, dressing, transferring and sleeping. To justify the large investment, these technologies must be scalable to thousands to millions of elders with low incremental costs.

IV. Knowledge Reactor

Existing approaches to IoT data fusion are either ad hoc or highly application specific and not reusable across cognitive applications, resulting in expensive duplicate efforts in data curation, integration and knowledge modeling for each cognitive service or application. The KR is a contextual data fusion engine that centralizes IoT and System of Record/Engagement data to create a reactive knowledge graph. Cognitive applications are written as a multiagent system. The KR scales-up and scales-down, exploiting container-based, horizontally scalable graph store (JanusGraph) and pub-sub (Kafka) technologies that sit logically atop the Watson IoT Platform.

A. Design requirements

The evolving nature of the eldercare problem will require significant evolution of the use cases, ontologies, data structures and target environments as work progresses and domain understanding improves. Therefore, over-arching KR requirements are for a systems that is *contextual*, *flexible*, *adaptable*, *incremental*, *scalable* and *performant* .

B. Architecture

Context is crucial to a proper solution. Many IoT solutions are instances of a single pattern: *one sensor; one rule; one* *alert*. A single sensor rarely sufficiently distinguishes between "good" and "bad" events. Consider kitchen fires as a simple example that is grossly unable to distinguish between good and bad. It is easy enough to point a flame sensor at the top of the stove (one sensor), and when flames are sensed (one rule), call emergency services (one alert). However, what if a grandmother hosts her grandson's birthday party. Burning candles on a birthday cake are certainly a fire but should not trigger a call to emergency services. Not all flames in the kitchen are "bad".

Adding context substantially changes the interpretation of events. Knowing (a) not everyone in the dwelling is currently performing a distracting activity (e.g., talking on the phone), (b) multiple people are in or around the kitchen, and (c) the elder's social network includes people whose birthday or anniversary are close to today's date would justify classifying the kitchen fire very differently. *Contextualized* data is the key to reducing both false negatives and false positives.

The use case requires a diverse range of data structures: (a) wide variety of personal information, (b) EMR (electronic medical record) data, (c) spatial information about dwellings, (d) social networks, (e) taxonomic and ontological structures, and (f) lists of sensor readings. A graph store is the heart of the KR, since graphs are capable of concisely representing these different data structures in an integrated and natural fashion. The reactive graph database is scalable, and is a modern version of the classic "blackboard" architecture [1]. A modular, *multiagent* approach enables *flexible* and *incremental* applications. Each agent performs a limited well-defined task and agents inter-operate. Agents may be incrementally added and recombined to satisfy emerging use cases.

A single analytic—even a machine learned one—is inadequate to span from sensor inputs to recommendation outputs; that is too big of a leap. Rather, the KR's overall approach for constructing context is to build up multiple layers of context, each layer at progressively higher conceptual levels.

C. Implementation

These requirements led to the implementation of the Knowledge Reactor as shown in Figure 2. It is the core infrastructure to gather, maintain, and analyze the sensor data and finally generate actionable insights.

The graph store is implemented using JanusGraph [2], because it natively supports graph structures (vertices and edges) and has demonstrated both *scalability* and *performance*.

All transaction commits are intercepted so that a simplified representation of all changed vertices and edges can be written to Kafka [3] (a publish/subscribe messaging infrastructure), which has also demonstrated *scalability* and *performance*. This feature makes the graph *reactive*: any agent can subscribe to one or more Kafka topics and listen for changes. Interacting with other data stores (e.g. Mongo) and messaging infrastructures (e.g. Watson IoT platform [4] and MQTT) are also possible.

To support *flexibility* and quick turn-around of development changes, automated "devops" techniques automatically build and deploy KR instances to physical computer servers. Docker [5] container technology enables servers to be quickly and easily deployed on computer servers. Containers consume far fewer system resources than hypervisors. Docker is a key element of the KR build process, delivering speed and repeatability to build and deploy KR instances. There are currently around 70 running KR instances for various live, test and demo purposes. The KR currently contains eight Docker images: Zookeeper, Kafka, Cassandra, JanusGraph, Tomcat, Node-RED, Mongo, and TinkerTools (a locally developed tool).

A rapid and automated build process efficiently supports the large volume of *incremental* and evolutionary changes across multiple server types. And a rapid and automated deployment process supports deployment of the many KR server instances. Both are necessary to support production-scale deployments.

D. Agents

Many of the applications needed to support the use case must be long running. This is accomplished by decomposing applications into a collection of *agents*.

Agents are reusable, long running functions that communicate with one another. They run on top of the KR infrastructure, using its facilities. Agents can read from and write to the graph using TinkerPop3's [6] Gremlin graph query language. They can read from and publish to Kafka topics. Agents are self-contained and isolated pieces of code (they don't share application-level, data structures, but do share low-level, infrastructure-level, data structures). This enables each individual agent to be moved between servers to improve load balancing.

Applications are built as multiple layers of agents. The lowest layer agents read data from the sensors and write that data into the graph with the required structure. Agents in higher layers listen for lower level changes to the graph and perform further processing. They communicate with each other primarily through the graph ("blackboard"). Some agents aggregate sensor readings from the graph into time windows of readings, which are written back to the graph as new "streams" of data. Another layer of agents read the time window data and classify them into ADLs. Higher layers compare current ADL patterns with historic patterns, and write any anomalies back to the graph. The highest layer of agents decide which action, if any, to execute. A more complete flow through multiple agents layers is: (a) in-dwelling sensors, (b) situational awareness, including behavioral awareness via ADLs and IADLs, (c) identifying abnormalities (d) accessing problems and risks, (e) planning and proposing multi-step responses, (f) decision making between proposed responses, and finally (g) notifications/alerts and in-dwelling effectors (e.g. cut power to stove).

The goal of data fusion is to build world models at different conceptual levels, including situational awareness, proposed plans and decision making. KR agents incrementally fuse a wide variety of sensor data types together into a combined graph model. This includes fusing different sensor modalities and (in the future) fusing sensor and Electronic Medical Record (EMR) data.

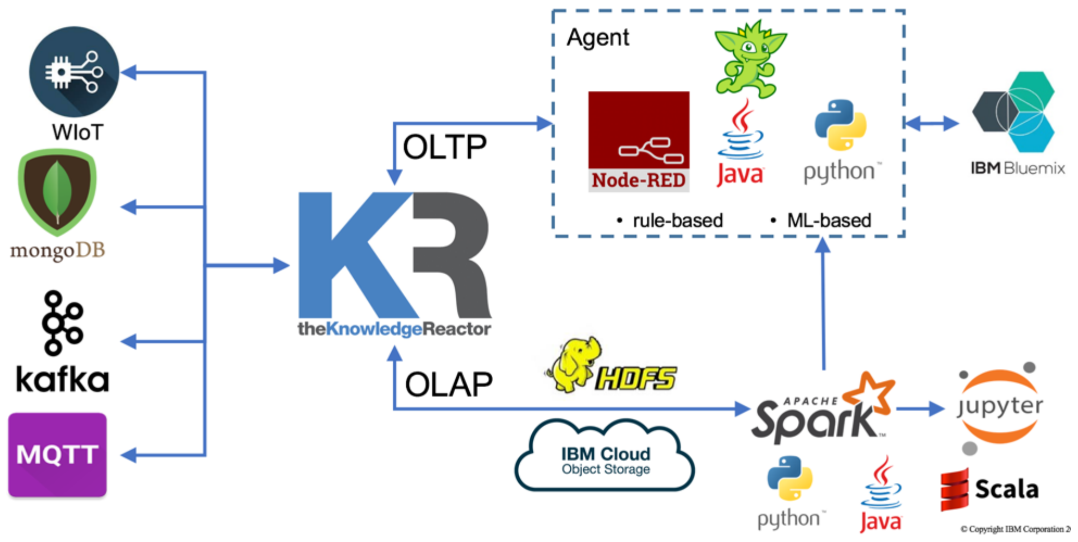

Fig. 2. Major elements of the Knowledge Reactor environment.

The KR supports two approaches to analytics: OLTP (on-line transaction processing) for small, localized graph processing and OLAP (on-line analytical processing) for "whole graph" analytics. OLTP agents can be written in Node-RED [7], Java or Python. Agents can read and write to the graph using TinkerPop3 APIs, subscribe and publish to Kafka topics, and perform any other necessary processing. For large analytic tasks (OLAP), data can be extracted from the graph and processed in Spark [8] using either Jupyter notebooks [9] or stand-alone programs.

Node-RED promotes a *reactive*, stream-based, programming model in which an application is constructed from a collections of inter-connected "nodes". Messages flow into a node, are transformed inside the node, and then flow out of the node and into the nodes following it. Reactive programming is a very natural and convenient way to write long running agents, and Node-RED supports it natively.

Trying to write reactive Java agents with just primitive threads and queues was significantly harder than Node-RED's approach. To improve Java development, various *reactive* approaches were examined. Java 8 Streams are not multi-threaded and do not support merging streams, so it was not appropriate. A custom "Block" framework was implemented that mirrors Node-RED's reactive nodes, and it was used for many agent implementations. Blocks transparently supports two execution modes: one thread per block (which can be performance intensive) and blocks that submit work items to a Java ExecutorService where items are executed by a common pool of threads. More recently, RxJava [10] is being investigated for future agents.

E. Ontology

Data must be given a structure and interpretation for agents to manipulate it. The term *ontology* describes that structure of data elements and their interrelationships. Because data is based on TinkerPop3 rather than RDF [11], the KR ontology is less formal than ontolgies defined using W3C's formal OWL [12].

A large number of existing ontologies were reviewed: [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. While each have merits, they were either too detailed or didn't sufficiently cover the concepts we required. In the end, the KR needed new ontologies, adapting ideas from those existing ontologies where ever possible. Given the emerging and evolving use case(s)—particularly during early versions—ontological innovation and experimentation is expected. So the ontology must be *flexible* and *adaptable*.

Given the wide span of concepts that must be represented, the complete ontology is composed as a collection of multiple, overlapping ontologies ("mini-ontologies") including (a) people, (b) spatial environments (c) physical sensors, and (d) sensor readings. Additional ontologies will certainly be added over time, for example the elder's social network. Ontologies can reference elements in other ontologies, connecting the data together.

The Person ontology contains names, identification numbers (e.g. facility and federal medical record numbers, SSN, etc), addresses, phone numbers, emails, etc. The spatial ontology adds the dwelling, rooms, connectors between rooms, and shape information. The sensor ontology describes the sensor hardware make and model, unique sensor id, the room in which it is located, and information about the types of data it collects. A graphical store naturally represents all these ontologies.

Since the bulk of data items are sensor readings, that ontology is currently the most fully developed. All sensor readings are timestamped, and the graph data structure grows linearly over time, with each new sensor reading adding vertices and edges to the end of a growing list of readings for each sensor. Importantly, this structure parallels the structure of feature vectors used by many machine learning algorithms.

Some agents read from one sensor reading list and aggregate readings into time windows and store the result as a new list. Some agents read from one or more lower-level sensor reading lists and construct lists of higher-level feature values. Higher-level readings (vertices) can explicitly reference the lower-level readings from which they were derived. All have a similar ontological structure. Multiple layers of agents construct multiple layers of lists.

Incremental evolution requires support for data sets from multiple ontologies co-existing, side by side, within the same graph. Since agent implementations are closely tied to the ontologies of the data they manipulate, multiple agent sets will also co-exist, side by side. Other than the additional effort for agent development and maintenance, the KR easily supports multiple ontologies and agent sets.

F. Tooling

Given the infrastructure above and data ingested into the graph, multiple, locally-written tools quickly visualize the data and enable users to work with large data volumes.

One particularly valuable tool is the locally-written *Topview* tool (see Figure 1), which displays a spatial/temporal replay of color-coded "sensing regions" overlaid on top of a background floor plan of the dwelling. It then replays a stream of recorded, sensor reading events, allowing users to adjust the replay speed from 1X, 10X, 100X, 1000X to 10000X. When a motion sensor turns on or off, Topview displays or hides the sensing region shape. The user sees the dwelling's floor plan, overlaid with sensor regions flashing on and off. Topview represents each type of sensors and placement using a unique letter, color and shape. Table I shows Topview's mapping of letters to sensor types. Visualizing sensors streams—particularly at high speed—makes it much easier to "see" activities in the dwelling, than by looking at a spreadsheet of lifeless on/off events.

TABLE I
TOPVIEW LEGEND

Sensor Letter	*Sensor type*
A	accelerometers
C	contact sensors on doors
M	motion sensor on ceiling
R	motion sensor on walls
W	weather sensor

V. Real World Application

This section describes an application of the Knowledge Reactor to a real world eldercare environment. We partnered with Avamere Family of Companies [28] to deploy the solution in two different Avamere environments: 20 SNF dwellings located on one floor of a SNF and five apartments in an IL. Each of the 25 dwellings were unique in size and configuration.

The research objective collects data from sensors to infer changes in ADLs and IADLs, and to then assess each resident's physical and cognitive status.

A. Sensor Deployment

The sensored dwellings feed sensor data into the KR for analysis. Because this study does not include ethnographic observation, the high-density distribution of sensors and the sensor placement were critical for analysis and modeling of ADLs and IADLs.

The Samsung SmartThings [29] and Netatmo [30] suites of IoT products were deployed in each dwelling. Flowing through Watson IoT, the KR ingests sensor readings from both ecosystems' cloud platforms and places them in the graph.

SmartThings Motion sensors were attached to fixtures such as walls and ceilings to monitor motion. The motion sensors were carefully placed so that one could differentiate between motion in different areas of the dwelling. For example, motion sensors were placed on the ceiling to detect movement below. The lens of motion sensors over toilets and showers were partly covered to narrow their sensing region for more localized detection. Placing motion sensors on both the walls and ceiling creates an intersecting sensing regions for better localization.

SmartThings Multipurpose sensors were attached to fixtures such as doors, washing machines, and furniture to monitor the opening/closing and detect presence from acceleration measurements along the three coordinate axes (x,y,z). These sensors were deployed on bathroom, shower, closet and cabinet doors as well as on furniture, bed rails and hand sanitizers.

SmartThings Outlet sensors were installed to measure power consumption of televisions and microwaves. Low consumption indicates the device is off (many devices never draw zero power); higher consumption indicates the device is in use.

Netatmo weather sensors measure temperature, humidity, carbon dioxide, and noise levels both inside and outside of the dwelling. They were placed in the bedroom, bathroom and kitchen of each dwelling.

To organize the deployment of the high density of sensors, a sensor naming convention was created that identifies each sensor and its placement. The naming convention consist of a short code (e.g. M6 as shown in Figure 1) and an expanded, unique 16-character string (e.g. I01BBB-lrscl200). The short code enables quick and convenient identification of a sensor on the sensor deployment schematic.

There are no best practices indicating the optimal number and positioning of sensors to measure ADLs and IADLs with a level of accuracy that enables the prediction of change in the elder's health status. Rather than experimenting with different sensor densities during the deployment phase, this research begins with a dense sensor deployment (”over-sensoring”) of the dwelling. The most efficient deployment is then determined during the analysis phase by selectively eliminating sensor data before analysis. Such studies will form the basis for more informed sensor deployments in the future.

B. Sensor Data

Table II shows summary statistics of the number of entities in the sensor deployment. In total, 519 sensors were deployed across 73 rooms in 25 dwellings (a dwelling can have multiple rooms). Most dwellings in the SNF were private bedroom with two bedrooms sharing a common bathroom. In the IL, each dwelling has multiple rooms: living room, kitchen, bedroom, bathroom, etc. Data was collected for 124 residents across 158 continuous days.

Table III shows a total of 8.9 million readings were collected across all dwellings and all sensor-types. This represented

TABLE II
NUMBER OF ENTITIES BY TYPE

Entity Type	*Count*
Persons	124
Dwellings	25
Rooms	73
Sensors	519
Days	158

2.6 GB of data. Motion sensor readings account for the bulk of the readings at 65%. Only 7% of the readings were generated by Netatmo Weather Stations, which only reported enironmental conditions every 10 minutes.

TABLE III
NUMBER OF SENSOR READINGS BY SENSOR TYPE

Entity Type	Records	Percent
Motion readings	5,828,361	65%
Contact readings	1,106,332	12%
Accelerometer readings	1,416,067	16%
Weather readings	612,594	7%
Total readings	8,963,354	100%

VI. ANALYTICS

This research builds KR knowledge graphs to detect and establish a baseline to monitor changes in an elder's ADLs and IADLs as well as any other changes in patterns observed in the data. Having just recently received the sensor data, only preliminary results are available.

Two approaches to ADL and IADL classification are used: rule-based and ML-based. Avamere required no cameras, microphones, ethnographic observations, surveys or interviews were used in this study. Therefore, traditional ML classification accuracy measures that require ground truth could not be calculated. However, intuition and a basic understanding of human behavior enables mapping key sensors to detecting ADL and IADLs.

A. Sensor Classification of ADLs and IADLs

The data models define how ADLs/IADLs and sensor signals relate to each other and how they are classified and analyzed. The models are based on the knowledge of the activities and their basic properties.

To classify activities (ADLs and IADLs), sensors are identified as (a) highly relevant, (b) corroborating, or (c) conflicting. A highly relevant sensor is required to detect the activity. A corroborating sensor enhances belief the activity occurred. A conflicting sensor diminishes belief the activity occurred. Consider classifying the toileting ADL based on the sensors in Figure 1. The sensor M8 above the toilet is highly relevant to toileting. The adjacent sensors R1, C9, M7, M9, and W2 in the bathroom corroborate toileting. Lastly, the motion sensors M10 above the bed and A4 above the living room chair are conflicting because a person can't generate readings from those sensors while toileting.

B. Rule-based ADL classification

The first approach to ADL classification is rule-based classification. Specifically, sensor readings are organized into one minute time intervals. Intervals are created by truncating the seconds and millisecond values in the event's timestamp. The window counts the number of events from each unique sensor that occured during that time window. To reduce data volumes, no time windows are created for minutes with no events.

Next, hand-crafted rules were developed based on the set of sensor readings received during each one minute time interval. The rules labeled each time window with either an ADL or as "no activity". Some rules are fairly straight-forward. For example, presence detected by motion sensors over toilet are highly relevant evidence for toileting. Or presence detected by motion sensor in shower is highly relevant for showering. Other rules are more subtle as they involve multiple sensors. For example, motion detected in kitchen together with cabinet door accelerometers corrobarate evidence for the IADL of preparing a meal. Once the data is labeled, ADL and IADL behavior patterns are studied over time to establish a baseline of normal and abnormal behavior for each individual.

In addition, ADLs and IADLs are aggregated into routines (e.g. the sequencing of ADLs and IADLs throughout the day). The routine of one elder showed a decline in transferring and an increase of time in bed, over the period of a month. This type of routine change would alert a care provider of a possible illness or injury. Based on the sensor deployment, experience suggests that ADLs and IADLs in some dwellings can be difficult to identify via sensor readings. For example, no rules could be identified to detect a dressing ADL in the studio apartment in Figure 1, because there was no independent clothing closet or dedicated dressing area in the apartment.

C. LDA-based ADL Classification

Machine learning ("ML") is the second approach to ADL classification. Latent Dirichlet Allocation (LDA) is a standard approach to classify a corpus of textual documents into the hidden (latent) "topics" inside each document, based on the words they contain.

Adopting an approach similar to [31], [32], LDA classifies each one minute time interval of sensor readings into seven "topics". These seven topics include the six common ADLs (toileting, showering, grooming, dressing, eating, and movement) as well as an additional "relaxing" ADL.

For the LDA-based classification process, the sensor readings from the KR are extracted from the graph and writen to IBM's Cloud Object Store. A Jupyter notebook runs the LDA algorithm in a Spark cluster on IBM Cloud to find the seven topics. Each topic is then manually assigned an ADL by looking at its most prominent sensor readings. Finally, the ADL for each LDA topic are manually determined by examining which sensors in the topic are highly relevant, corroborating and conflicting.

VII. RESULTS

While research projects are primarily measured based upon successfully demonstrating workability, projects aiming for production demand scalability in both ease of deployment and volume of data. The initial implementation of the Knowledge Reactor in the Avamere environment highlighted some factors that must be considered when deploying a large-scale sensor-based monitoring system.

The KR devops process can incrementally regenerate a new build in 3 minutes, and can automatically deploy 20 KR server instances from scratch in 90 minutes.

To demonstrate *scalability*, 20 copies of a graph dataset of nearly 1.3 million sensor readings were loaded into the KR. This created a graph of 103+ million graph elements (vertices and edges). It took 42 hours using a single-threaded, Java ingestion program, or 5.9 milliseconds/reading. This linear performance generally validates the choice of JanusGraph and Kafka. Furthermore, implementation changes have been identified to make the ingestion program multi-threaded, which will dramatically reduce total ingestion time for large graphs.

A code base usable only by its creators is worthless. A company-internal, month long, "code-a-thon" invited 18 world-wide participants to use the KR and provide feedback on the KR's technology and usability of its supporting educational materials. Working on their own time, a third of the participants successfully completed all exercises which included the setup of the Knowledge Reactor, writing agents, reading and writing to the graph, and basic data analysis. This is preliminary demonstration of the KR's *usability*.

Running the LDA algorithm on a two server, Spark cluster identified seven ADLs. Six of the seven topic were easily identified with ADLs; the seventh topic could not be identified. Table IV shows the seven topics, and the number of time windows assigned to each ADL. However, further refinement is needed since sleeping should be the most frequent ADL; not bathing. Possible causes for this are (a) the number of topics to construct is an input parameter to LDA and a wrong value could incorrectly merge dissimilar ADLs; (b) a motionless sleeper does not generate events, and time windows without readings are eliminated; and (c) poor manual assignment from topic to ADL;

An interesting insight was discovered. One topic contained both the motion sensor above the toilet and the accelerometer sensor attached to the front door. At first, the correlation was puzzling, but now makes sense because people often use the toilet just before leaving and just after arriving.

Since no ground truth for ADLs is available, accuracy results aren't available. Future work plans to analyze our data in two separate ways. First, compare the rule- and (refined) LDA-based classification results. ADL agreement between two dissimilar approaches would lend credibility that they are both detecting the same phenomenon. Second, run a pseudo-ethnographic analysis, where humans view elder activities—as seen through the limited lens of just sensor data—on a pre-defined set of time windows. Then compare the ethnographic classification with the rule- and ML-based classifications.

TABLE IV
ADL FREQUENCY FROM LDA

ADL	Time Windows	Percent
Cooking	9,581	10%
Unclear	11,658	12%
Transferring	9,585	10%
Toileting	13,263	14%
Bathing	25,393	26%
TV	12,601	13%
Sleeping	13,846	14%
Total	95,927	100%

VIII. CONCLUSION

This research successfully demonstrated many points. The Knowledge Reactor infrastructure is a useful approach to applying contextual data fusion and AI techniques toward easing the eldercare problem. It is scalable, and is usable beyond its developers. A large and valuable data set for elders in both SNF and IL facilities was created.

Two analytical approaches to matching sensor data with ADLs were demonstrated with encouraging initial results. However, the LDA analytic still requires manual assignment to topics to ADLs. This will not scale to thousands of unique dwellings. Future work will investigate using ontological labels (similar to [31]) to automatically transfer topics-to-ADL assignments across dwellings.

Fusion of sensor and EMR data is both challenging and exciting. It is an critical future work item.

Current methods can not differentiate between multiple individuals in a single dwelling. This issue must be addressed before any practical deployment.

While still a work in progress, the Knowledge Reactor has demonstrated many important capabilities required to address the eldercare problem at large scales. It is also well positioned to address other, similar IoT and AI problems.

ACKNOWLEDGMENTS

Thanks to the Avamere Family of Companies for all of their insights, hard work and valuable data throughout this project.

REFERENCES

[1] A. Newell, "Some problems of basic organization in problem-solving programs," RAND, Tech. Rep. RM-3283-PR, 1962.
[2] "Janusgraph," 02 2018. [Online]. Available: http://janusgraph.org
[3] "Kafka," 02 2018. [Online]. Available: http://kafka.apache.org
[4] "Watson IoT," 02 2018. [Online]. Available: http://www.ibm.com/Watson/IoT
[5] "Docker," 02 2018. [Online]. Available: https://www.docker.com
[6] "Tinkerpop3," 02 2018. [Online]. Available: http://tinkerpop.apache.org
[7] "Node-red," 02 2018. [Online]. Available: http://nodered.org
[8] "Spark," 02 2018. [Online]. Available: https://spark.apache.org
[9] "Jupyter," 02 2018. [Online]. Available: http://jupyter.org
[10] "Rxjava," 02 2018. [Online]. Available: https://github.com/ReactiveX/RxJava
[11] "Resource description framework," 02 2018. [Online]. Available: https://www.w3.org/RDF
[12] "Web ontology language," 02 2018. [Online]. Available: hhttps://www.w3.org/OWL
[13] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Berges, D. Culler, R. Gupta, M. B. Kjærgaard, M. Srivastava, and K. Whitehouse, "Brick: Towards a unified metadata schema for buildings," in *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, ser. BuildSys '16. New York, NY, USA: ACM, 2016, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/2993422.2993577
[14] "Sensor model language," 02 2018. [Online]. Available: http://www.opengeospatial.org/standards/sensorml#schemas

[15] "W3C semantic sensor net ontology," 02 2018. [Online]. Available: http://www.w3.org/2005/Incubator/ssn/wiki/Semantic_Sensor_Net_Ontology

[16] "Review of sensor and observations ontologies," 02 2018. [Online]. Available: http://www.w3.org/2005/Incubator/ssn/wiki/Review_of_Sensor_and_Observations_Ontologies

[17] "Ontology design patterns," 02 2018. [Online]. Available: http://ontologydesignpatterns.org/wiki/Submissions:ContentOPs

[18] "Quantities unitsdimensions and types," 02 2018. [Online]. Available: http://www.qudt.org

[19] H. Lee and J. Kwon, "Ontology model-based situation and socially-aware health care service in a smart home environment," vol. 7, pp. 239–250, 09 2013.

[20] "Indoorgml," 02 2018. [Online]. Available: http://docs.opengeospatial.org/is/14-005r4/14-005r4.html#12

[21] "Citygml," 02 2018. [Online]. Available: http://www.opengeospatial.org/standards/citygml

[22] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," in *IN PROCEEDINGS OF COMMUNICATION NETWORKS AND DISTRIBUTED SYSTEMS MODELING AND SIMULATION CONFERENCE*, 2004, pp. 270–275.

[23] M. Miraoui, S. El-etriby, C. Tadj, and A. Abid, "Ontology-based context modeling for a smart living room," 10 2015.

[24] "Daml+oil," 02 2018. [Online]. Available: http://www.daml.org/2001/03/daml+oil-index.html

[25] S. Tazari, "universaal/ontology," 02 2018. [Online]. Available: https://github.com/universAAL/ontology/wiki

[26] "Sample decision ontology," 02 2018. [Online]. Available: https://www.w3.org/2005/Incubator/decision/XGR-decision-20120417/Sample_Decision_Ontology.html

[27] "Relationship: A vocabulary for describing relationships between people," 02 2018. [Online]. Available: http://vocab.org/relationship/

[28] Avamere, "Avamere," 2016. [Online]. Available: https://www.avamere.com/

[29] "Smartthings," 02 2018. [Online]. Available: https://www.smartthings.com

[30] "Netatmo," 02 2018. [Online]. Available: https://www.netatmo.com

[31] I. Ihianle, U. Naeem, and S. Islam, "Ontology-driven activity recognition from patterns of object use," pp. 654–657, 09 2017.

[32] F. J. Ordonez, 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. [Online]. Available: http://www.mdpi.com/1424-8220/13/5/5460