Practice Makes Perfect: Lesson Learned from Five Years of Trial and Error Building Context-Aware Systems

Ryan Mullins
Aptima, Inc.
rmullins@aptima.com

Adam Fouse
Aptima, Inc.
afouse@aptima.com

Gabriel Ganberg
Aptima, Inc.
ganberg@aptima.com

Nathan Schurr
Aptima, Inc.
nschurr@aptima.com

Abstract

Recent advances in artificial intelligence have demonstrated that the future of work will be defined by collaborative human-machine teams. In order to be effective, human-machine teams will rely on context-aware systems to enable collaboration. In this paper, we present three lessons learned from the past five years of developing context-aware systems that we believe will improve future system design. First, that semantic activity must captured, modeled, and analyzed to enable reasoning across missions, actors, and content. Second, that context-aware systems require multiple, federated data stores to optimize system and team performance. Finally, that real-time inter-actor communications are the essential feature enabling adaptation. We close with a discussion of the influences and implications that these lessons have on human-machine teaming, and outline future research activities that will be necessary before operationalizing these systems.

1. Introduction

An analysis of recent events illustrates a simple premise: machines are on the rise. This is shown in several domains. In 2016, the Defense Advanced Research Projects Agency (DARPA) hosted their Cyber Grand Challenge, in which fully-autonomous cyber reasoning systems successfully identified vulnerabilities in software [1, 2, 3]. That same year, AlphaGo scored the first machine victory over a human grand master in the game Go [4]. In 2019, OpenAI released what may be considered the first truly competent generative language model in GPT-2 [5].

Despite these monumental successes, machines have limitations. What they provide in scalability (the velocity with which tasks can be completed) they lack in adaptability (the variety of tasks they support). Garry Kasparov, the first world champion chess player to lose a match to a machine, posits a simple formula [6]: human-machine teams engaged in the optimal process will, eventually, outperform any combination of humans and machines using a lesser process.

It is this premise that has motivated our work for the last five years; designing, developing, and evaluating context-aware systems to improve human-machine team performance. How can we design systems that optimally enable human-machine collaboration? How do we reshape the capability curve (Figure 1) from convex to concave?

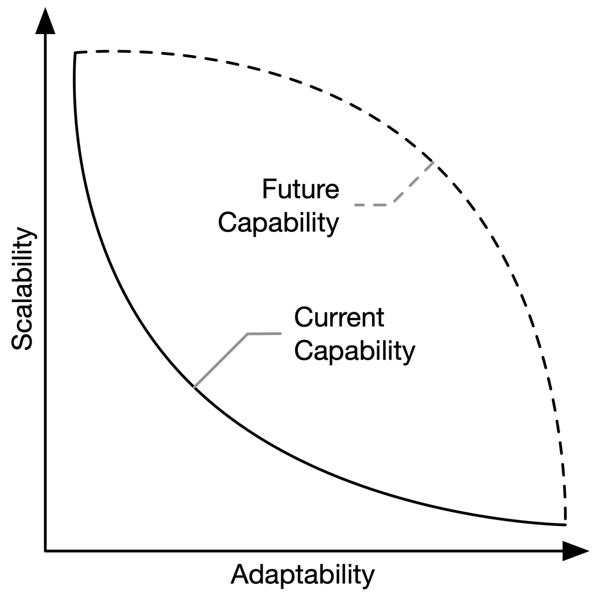

Figure 1. Current and future capability curves of human-machine teams.

Research in context-aware systems encompasses many domains, from ubiquitous computing [7, 8], adaptive user interfaces [9, 10], to knowledge bases [11, 12], to recommendation engines [13, 14, 15]. These efforts have addressed conceptual aspects of context-aware systems, such as how context is defined and what role it plays, technical aspects, such as selection of computational representations for context, and outcomes, such as the performance benefits of

URI: https://hdl.handle.net/10125/63774
978-0-9981331-3-3

context awareness.

In this paper, we contribute three lessons learned to this line of research, drawn from prototyping and evaluating context-aware systems, to aid in the practice of human-machine collaboration. In our reference systems machines act in supporting roles – as assistants optimizing workflows, as monitors of health reporting on system status, and as workers automating common tasks. These systems were built for two domains: information analysis, including long-running and real-time analysis, and command and control of cybersecurity operations. We close with a discussion of the implications these lessons have for advancing the design and evaluation of context-aware systems enabling human-machine collaboration.

2. Related work

2.1. Knowledge management

Knowledge graphs [11, 16] have become a best-practice for knowledge management in context-aware systems [17, 18]. While there is yet to be a universally accepted definition of a knowledge graph [16], for the purposes of this research we define the term as a datastore (another term for database) connecting entities through relationships to enable contextualized content retrieval.

The scope of knowledge graphs is determined by the intended adaptability of the system. Large knowledge graphs tend to focus narrowly on one task, information retrieval, for example, as exemplified by Google [11, 12] and others. Knowledge graphs of this scale optimize global relevance, from a statistical perspective, at the expense of addressing any specific context. While these knowledge graphs are insufficient to qualify as human-machine teaming systems, the data enrichment methods they use – such as topic modeling, named-entity extraction, and entity resolution – are essential to enable those richer systems.

Methods to generalize knowledge graphs to include nascent representations of actor preferences in the task are now common. Consider commercial recommender algorithms used by Netflix [13], Hulu [14], and Amazon [15], which treat context as a series of sparse vectors that can be isolated and assessed for similarity [19]. However, these techniques have a critical limitation that inhibits them from providing generalized support: they assume that individual preferences never change [14] which does not hold for real-world work environments. As a result of this assumption, recommender systems are designed to support a single task, and can require multiple knowledge graphs and analytical capabilities to support multiple tasks.

Ganberg et al. [20] proposed a general framework for modeling human-machine teaming context by representing four distinct types of data – environment, performers, mission, and interactions – as concrete entities, and integrating them into a unified domain model. Of particular relevance to this paper are their representations of environment and interactions. Environments are representations of a predominantly physical world, with an emphasis on data collectors and the performance space characteristics that capture work in the physical space in a digital model. Interactions are directed representations of communications and actions, typically separated into pair-wise relationships – performer-performer, performer-entity, and entity-performer. Pfautz et al. [21] implemented an expanded version of the domain model in a property graph [22] database. This method has two limitations that we seek to address with this paper: 1) the highly explicit and static definitions of major concepts, especially of the environment and interactions, that inhibit adaptability; and 2) the reliance on a single method for storing and analyzing data that can restrict representation possibilities or induce performance bottlenecks.

2.2. Temporal nature of context

A commonly used definition of context is ”any information that characterizes a situation related to the interaction between humans, applications and the surrounding environment” [23]. While this situation-focused definition recognizes the changing nature of context over time, little attention has been paid to the different time-scales at which context exists. Of particular importance to our work is the relationship between events in the world and changes to context.

To ground understanding of this relationship in models of human behavior, we look to psychological theories of event perception and segmentation. According to this research [24, 25] human perception of activity in the world is managed by segmenting the activity into discrete events, which become the basic unit. These theories posit (with support from behavioral and neurological evidence) that events are represented on multiple time-scales and organized in a hierarchy ranging from short to long duration.

If we enrich the definition of context provided above to posit that situations are partially defined by the ongoing stream of events, it naturally follows that the hierarchical nature of event perception should influence how we model context and implement context-aware systems. For example, take the event of writing a research paper. This event may be built out of smaller events such as creating an outline, writing a section, and editing. These may be further broken down into small events such as re-writing a sentence.

To model the context of these events, the hierarchical relationships need to be considered, and the support provided would depend not only on the aggregate context, but on the changes to the smaller-scale events and their relationship to other scales in the hierarchy. Previous work has applied these theories of event segmentation to the design of information system, such as guiding the timing of notifications [26] and the design of planning software [27]. These applications have shown improved performance, as measured by reduction in interruption cost for notifications, and increased accuracy of planning.

3. Methods

We developed the lessons learned presented in this paper through an action research [28, 29] program spanning five years. The goals of this program were to simultaneously provide tools to improve performance in the domains of study, while using the iteration across multiple applications to identify principles of context-aware system design through reflection on the outcomes obtained from each iteration.

This program involved close collaboration with users in three different organizations, participatory design involving recently retired members of those organizations, development of tools to be used by members of those organizations, and iteration of approach between efforts. Each application began with domain analysis using techniques such as hybrid Cognitive Task Analysis [30] and Contextual Design [31]. These analyses provided grounding in the specifics of each domain, an understanding of critical challenges faced by users, guidance for interface requirements, and definition of the components of context.

At a technical level, the three systems we developed shared common characteristics: web-based user interfaces and visualizations, with back-end components providing data storage, processing, and analytics. The specific details of each implementation followed from the domain analysis, but common components allowed for rapid development to enable frequent feedback from users and iteration of design.

During development of those systems, we used several techniques to collect data regarding the design of these systems, with the technique chosen based on the readiness of the software, availability of users, and technical limitations of the users' environment. These data collection techniques included usability studies, semi-structured interviews [32], and cognitive walk-through [33]. At least two members of the research team observed and recorded notes for each session, and we used a grounded theory approach [34] to identify themes that emerged from across sessions concerning benefits and limitations of the systems we developed. These themes form the basis of the lessons learned that are reported in the following sections.

4. Lesson 1: Modeling semantic activity

Our first lesson is drawn from our experience designing web applications for the information analysis domain. Here, our typical human user is an analyst researching and answering questions from one or more content modalities, such as text, images, video, etc., that are streaming new content and updates into the system. We use Pirolli and Card's information synthesis process [35] as a generic model of the analyst's work. Further, tasking is assumed to be driven by a mix of top-down, influenced by customer requirements, and bottom-up, emerging from the analytical process, factors.

Under these assumptions, we prototyped a system that addresses the critical need of modern analysts: dealing with information at scale. How can the analyst find the relevant content? How can the analyst know that the content they find is reliable and credible? How can the analyst leverage this content effectively in their analysis?

Here, the role of the machine is that of an assistant; finding, delivering, and characterizing content that is relevant to their work. The machine must be able to understand the information needs of the analyst, which is to say, map the available content onto the analytical structure, quantitatively assess the relevance of the content given that mapping, and present content to the human without overloading them.

This problem is analogous to challenges faced by Netflix and others [13, 14, 15]. Typical solutions leverage collaborative filtering [19] to predict relevance. While this method has proven reliable and scalable, the static preference assumption [14] negates its applicability in highly-dynamic work environments. Analyst's needs can change from moment to moment in response to myriad factors, such as the information in the content they are examining, the availability of new content, or the requirements of their customer. The only way for the machine to adapt to or, ideally, anticipate the human's information needs is to analyze their activity.

This leads us to our first lesson: ***modeling and quantifying semantic activity is the essential feature space over which context-aware systems must operate***. This lesson poses two research challenges for context-aware system design: modeling semantic activity and capturing semantic activity.

4.1. Activity in knowledge graphs

Our approach builds upon the groundwork laid by [20] and [21], relying on a graph-centric representation, specifically a property graph [22], to model semantic activity. Here, we define semantic activity as an observable unit of work within the system. As with [20], we model semantic activity explicitly as nodes within the graph. Each node represents an action that is performed by an actor and, optionally, operates over content to achieve a mission objective. This definition allows us to simplify and enhance the domain modeling approach used by Pfautz et al. [21] to four layers:

1. ***The Actor Layer*** is comprised of a two node types, the actor node, which represents discrete human or machine entities within the system, and the group node, which represent related collections of human or machine actors.

2. ***The Content Layer*** is a heterogeneous layer representing the data over which actors operate. It is equivalent, in many regards, to the traditional form of a knowledge graph [11], linking discrete source data nodes together through myriad extracted nodes derived from automated analytical methods or explicitly created from actor interactions with the system.

3. ***The Mission Layer*** is a heterogeneous layer that represents the goals, objectives, tasks, requirements, constraints, products, and the relationships there-between, that define the workflow. The relationships between these entities are typically defined by the domain, policy, or best practices of the organization. One common example would be *TaskN* having a Create relationship to *ProductQ*.

4. ***The Activity Layer*** is a homogenous collection of the semantic activity nodes described above. The core of each activity node is the action, start time, and end time attributes, stored as properties on the node. Additional properties can be stored depending on the domain needs. The semantics of that action are modeled as four types of relationships: inputs, outputs, actors, and mission. Inputs and outputs model the semantic of content transformation within the work process, which can be extracting or aggregating data into relevant information. Actor and mission relationships capture the semantics for who is performing the necessary work. An activity node must always have an actor and a mission relationship. Input and output relationships are not required unless the action mutates content.

In our layered model, the activity layer acts as a sort of connective tissue to fuse the knowledge graph, allowing for use case-specific implementation of the actor, content, and mission layers in the system. Figure 2 illustrates our approach for an information analysis use case. Activity (orange) is represented as discrete nodes in the graph, with edges pointing the associated actor (green), mission (red), and/or content (blue) entities. Activities 1, 2, 3, and 4 represent content extraction by an actor. They use by relationships to denote the actor, from relationships to identify the source content, and extracted relationships to identify the newly-created content. Activities 5 and 6 represent the actions of referencing and quoting content in a product. They use by relationships to denote the actor, in relationships to denote the product, and relationships with semantically meaningful labels (referenced and quoted) to denote the specific type of action that was taken. The results of these activities are represented as new relationships (black) in the graph that link the associated actors, content, and/or mission elements to each other directly using a separate set of semantic layers, such as provenance or structure. The advantage of this approach is that it enables multiple types of pattern recognition in the graph using different graph traversals. These differences can be leveraged independently or jointly by analytics, enabling several assessments of the data space, as in [36].

4.2. Capturing semantic activity

Activity logging is a well-documented practice in both academic and commercial realms. Traditional approaches model system-level events, such as mouse clicks, keystrokes, and/or window focus changes, then use post-hoc analyses to derive insights [37, 38, 39]. We find these methods to be insufficient for our purposes, as they require significant additional information to imbue the system events with process semantics, and are not well-suited to dynamic task changes.

Instead, we capture and represent activity semantics explicitly, as in [36, 40], by translating system-level events into semantic events imbued with the requisite data from our activity modeling approach explicitly within the user interface. The minimum tuple that must be captured is $\langle actor, target, action \rangle$, where target can be an entity in the mission or content layers. This tuple is appropriate when the actor is mutating a property of an element, such as rating the credibility of the information

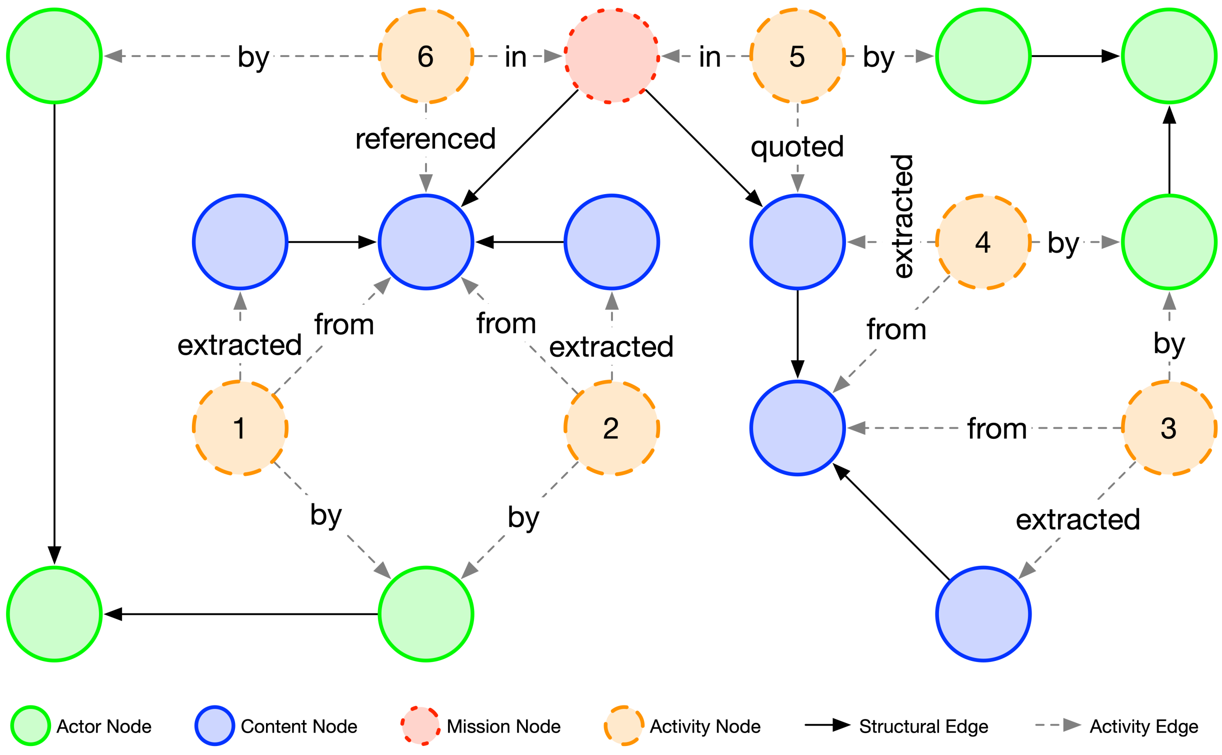

Figure 2. An example model illustrating how activity nodes and edges (orange) connect the disparate subgraphs of users (green), content (blue), and missions (red).

contained in a snippet of text. More complex activity may also include an output. In this case, the tuple would be adjusted to be $\langle actor, source, output, action \rangle$. We do not impose limits on the tuples that are captured, and recommend that the system capture as much detail as possible. The critical element of information is to label the actions of the activity nodes in a semantically meaningful way. Looking again at Figure 2, the activity nodes that capture the creation of derivative knowledge elements use from and extracted relationships to denote the source and the output. The precise semantics of these relationships and analytics thereover will change with the operational domain, use case, etc.

5. Lesson 2: Optimize representation

Our second lesson was drawn from the development of situational awareness tools for organizations. Here, we define an organization as a collection of actors who can be divided into teams based on the nature of their work. Our model organization was a security operations center (SOC) [41] providing real-time and long-term network defense to a large organization. The system was designed to support the leader's situational awareness [42], given the specific considerations of the SOC and the cyber domain [43]. The leader manages defensive activities of multiple constituent teams with different roles, such as incident response, forensic analysis, and vulnerability assessment. Depending on the nature of their tasking, teams may be able to execute in parallel, require sequencing, or operate in opposition to each other's goals.

For this use case, we used SOC health as a proxy for the leader's situational awareness. To understand health, the system requires real-time status monitoring for individuals actors, teams, their tasking, and the network infrastructure they are defending. SOCs are inclusive of machine actors, who provide autonomous functions like intrusion detection or network load monitoring.

Health monitoring in the cyber domain requires two distinct types of information to define system context: 1) data at the edge, such as packet capture data from a network traffic sensor, and 2) derived data, such as measures of performance on a task, or the impact on organizational capability if an asset goes down. There exist significant data aggregations and transformations between these two types of data, requiring a non-trivial model of a data generating process to communicate the derived health measures effectively. Further, the data required for these analyses are significant in volume and velocity, therefore, the system must process these data in place and communicate back only aggregate findings to the user.

The combination of analytical complexity for decision-support and analysis-in-place presents our second lesson: ***context-aware systems should use a datastore federation around a central knowledge graph***.

5.1. Multiple representation approach

As discussed in Lesson 1, we have found that a knowledge graph is essential for context-aware system design. However, the role of the knowledge graph should be understood: these are fused representations over heterogeneous data. Complex work environments, such as SOCs, require multiple datastores operating as a federation in order to optimally store and analyze data for a specific task. When architecting the federation for a context-aware system, there are five datastore classes to consider: relational [44], indexes [45], spatial [46], temporal [47], and blobs [48]. Depending on the constraints and affordances, the system could be architected to include any combination of datastores.

In these architectures, the knowledge graph serves as the central fusion point, but it can also act as a sort of memory for the system with minor enhancements to the content and mission layers. For the content layer, in addition to modeling the content actors operate over, we have found that modeling the location and access method for this content as relationships in the graph. These relationships are only necessary for the content entities that the actors explicitly use in their work. For the mission layer, we have found that modeling the methods actors use to complete their work provide useful insight both into the data generating process, as required by the SOC use case, and the general workflow used by human-machine teams.

When enriched in these ways and fused through the semantic activity entities described in Lesson 1, this approach benefits both research and operations. First, the knowledge graph provides a record of the work that can be done by the system (e.g., tasking, goals, and other mission layer entities), the work that was done by the system (e.g., tasks completed and associated entities in the activity layer), and, through these enhancements, how that work was done. Critically, our approach does this without over-burdening the graph, minimizing the impact that centrality [49] and other metrics impose on analytics. Second, these data provide a complete record of the human-machine team workflow. To utilize this record, the activity layer can be exported and transformed into an index or time series of events for further analysis. Some potential uses include real-time augmentation via workflow recommendations [36], and team optimization through post-hoc pattern analysis [50, 51]. Finally, this record can be used to provide data to myriad training or simulation tasks outside the confines of operational use.

6. Lesson 3: Adapt to real-time needs

Our final lesson stems, again, from the information analysis domain, this time exploring human-machine collaboration for real-time video annotation. Here, our typical human user is an analyst extracting data from a live video stream as part of a team of analysts. They are supported by automation in the form of computer vision algorithms performing real-time image segmentation, track extraction, etc. Despite this automation support, humans remain in-the-loop to perform more complex functions, such as confirming the output from computer vision algorithms, translating that output into real-time communications, or developing informational products for customers.

The operational challenge is rapidly delivering on dynamic customer requests. The primary measures of performance are the volume, velocity, accuracy, and precision with which products are generated. Tasking in this environment comes from standing (known prior to the start of operations) and emergent (arising out of the observed events) requests. Each request generates an informational product, which may have a time-bounded performance metrics. For example, an emergent request for an annotated video frame must be delivered to the customer within X minutes.

As researchers, our challenge was imbuing the system with a sufficient model of context to rectify the interaction between the timescales of an ongoing effort (e.g., past missions in the same area), an individual analyst's activity (i.e., their tasking in-the-moment), and unfolding activity in the world that is captured by the video. Although the approaches described in Lessons 1 and 2 incorporate models of context that update over time as a function of user activity, the changes in context are typically gradual, as opposed to rapid changes as a result of events in the data. This brings us to our final lesson: ***real-time reactions require that context-aware systems maintain representations of the temporal qualities of a changing knowledge graph***.

6.1. Real-time adaptation approach

Real-time adaptation requires the system to reason over multiple time scales. In addition to absolute real-world time, we also need to consider relative time as it passes during the course of a mission, and relative time as it relates to comparison to past data or events [52]. For example, activity at the beginning of a mission may be interpreted differently than activity at the end of a mission, and the recency of past data may determine whether the analyst or system should prioritize its inclusion in the analysis process.

Our approach to addressing these challenges of real-time adaptation enriches the heterogeneous knowledge graph approaches described earlier in this paper with label-based representations of temporal data. This enrichment builds upon the notion of absolute time, which is stored as a timestamp property on all relevant content, mission, and activity nodes in the knowledge graph, and augments it with time category labels. For mission-relative events, the system dynamically generates, assigns, and persists labels on key data. For example, a node representing a product in the content layer could have a time category label "$mission/id/created + 01:02:00$" to denote that the product was created one hour and two minutes into mission execution. Similarly, for event-relative time, the system dynamically generates and attributes labels to events observed within a mission relative to their last observation in the broader operational history. Triggers for updating the system's reasoning processes leverage a distributed event architecture [53] keyed off of label mutations.

6.2. Characterizing priorities

Analysts in these settings potentially need support from the system across a variety of tasks, such as understanding the activity they see in the video, identifying which frames of video to annotate, determining what annotations to create, or understanding which products of the analysis need to be collated and distributed. The system needs to understand these different possible priorities, determine when to provide support, and how that support may have changed due to ongoing events.

Our methods for determining support rely on multiple categories of time in combination with a streaming data model. As new data comes into the system, the data is evaluated both in isolation as well as in the context of the knowledge graph. This enables reasoning about intrinsic qualities of data (e.g., a priority request or a stated goal) as well as changes to the overall context.

A key input to changing context in this model is communication between actors in the system. The ongoing activity of the teams, especially in real-time settings such as the analyst team we consider here, is often orchestrated through frequent communication. This includes structured and unstructured communication, both within the team and with external organizations.

While the activity node structure was originally intended to represent interactions between actors and systems, this model can also be adapted to capture this communication between actors and each other. By representing those communications within the temporal knowledge graph, context reasoning can reflect the temporally changing priorities of the team.

7. Discussion

7.1. Limitations and scope

The systems we have described in this paper can be broadly described as decision and analysis support tools, where the support is tailored to the needs of users by representing and analyzing context. These systems have two characteristics that bound the scope of applicability for the lessons we have provided to the field of human-machine teaming. First, these systems were designed to support a limited view of the role that machines can play in work – as assistants in knowledge work that does not require physical embodiment in order for machines to effectively participate on the team. Extending these approaches to support embodied work, such as human-robot collaboration, would require a representation of the environment as a layer in the knowledge graph, as described in [20, 21]. Support for more autonomous machines requires a better model of business processes, using techniques such as process notation [54], and the incorporation of these processes into the knowledge graph such that other actors can be made aware of and are able to incorporate the outputs of a machine actor's work products [55]. Second, the lessons presented in this paper were derived from qualitative research methods, as discussed earlier. We have not quantified the changes in team performance as a result of the lessons presented, nor do we suggest generalizable measures and metrics against which performance can be compared. Further research is required to address this question.

7.2. Implications for human-AI collaboration

The future of humans interacting with machines is likely to be one of true collaboration. Recent advances in the field of artificial intelligence – for example, successes in application of deep learning techniques to play complex games such as Dota2 [56] – indicate an ever-increasing level of autonomous reasoning by machines. But collaboration between humans and highly autonomous machines remains elusive.

The lessons described in this paper apply directly to this problem. The first lesson, of the central importance of representing semantically-meaningful activity, applies not only to what support a system should provide, but to when and how that support should be provided. Knowing when and how to interact with others is a critical feature of successful collaborative teams [57].

The second lesson, of optimizing performance through multiple federated data stores, underscores the importance of flexibility and composability in context-aware systems, that is, having the ability to link native representations and analyses of data with model-based representations of context. Such hybrid techniques have been shown to provide advantages for accelerating learning [58], but also provide a mechanism for combining semantically meaningful representations with powerful learning techniques.

The third lesson, of temporal representations that include inter-actor communications, is critical for heterogeneous teams. Successful collaborative teams adapt their priorities and communication structures to match the needs of the work at hand and anticipated work of the future [59]. Collaborative systems cannot require humans on such teams to provide explicit representation of these mission and organization factors; this would be inefficient and brittle. Rather, intelligent collaborative systems should infer these changes based on changing context, and let human teammates know when that context needs additional input due to high degrees of uncertainty.

7.3. Future work

Looking ahead, we see numerous challenges in the future of context-aware systems and human-machine teaming. In the short term, and for the purposes of this paper, we have prioritized two: management and trust.

A major limitation of the current state-of-the-art in human-machine teaming is that systems require human intervention to manage configuration. In particular, the schema underlying knowledge graph requires human-driven updates to incorporate structural changes. New methods for architecting and delivering systems will be required to make this process more resilient.

Considering the state-of-the-practice, we believe that human understanding and inclusion of machines as teammates is the major limiting factor on human-machine collaboration and team performance. Human understanding and inclusion of machines is a classical problem of trust in autonomy. Hoffman et al. [60] explore the difference between interpersonal trust and human-machine trust, finding that the trust is generally similar between these two framings, but that machines lack the capabilities necessary to reestablish trust once it has been eroded. Adding this capability will be essential to enable true human-machine collaboration in the future. Rahwan et al. [61] recently noted the emergence of the field of artificial science, focused on the empirical study of the mechanisms, development, functions, and scale associated with machine behavior. A key critique they make is that many new advances in artificial intelligence (AI) are based on opaque methods, such as deep learning. Efforts are being made to make these methods less opaque, for example DARPA's Explainable AI program [62]. We plan to enhance our approach to incorporate explainable methods, and to associate them with mission, content, and activity elements.

Acknowledgements

The authors would like to thank Ms. Stacy Pfautz, Dr. Caroline Ziemkiewicz, and Mr. Roger Dziegiel for their contributions to this work.

This work was conducted in connection with Contracts #FA8750-14-C-0124, #FA8650-16-M-6736, and #FA8750-17-C-0203 with the United States Air Force Research Laboratory. The views, opinions, and findings contained in this document are those of the authors and should not be construed as an official position of the United States Air Force.

References

[1] M. Walker, "Machine vs. machine: Lessons from the first year of cyber grand challenge," 2015.

[2] J. Song and J. Alves-Foss, "The darpa cyber grand challenge: A competitor's perspective," *IEEE Security & Privacy*, vol. 13, no. 6, pp. 72–76, 2015.

[3] J. Song and J. Alves-Foss, "The darpa cyber grand challenge: A competitor's perspective, part 2," *IEEE Security & Privacy*, vol. 14, no. 1, pp. 76–81, 2016.

[4] F.-Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang, and L. Yang, "Where does alphago go: From church-turing thesis to alphago thesis and beyond," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 2, pp. 113–120, 2016.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

[6] G. Kasparov, *Deep thinking: where machine intelligence ends and human creativity begins*. PublicAffairs, 2017.

[7] L. Barkhuus and P. Dourish, "Everyday encounters with context-aware computing in a campus environment," in *International Conference on Ubiquitous Computing*, pp. 232–249, Springer, 2004.

[8] A. K. Dey, G. D. Abowd, *et al.*, "The context toolkit: Aiding the development of context-aware applications,"

in *Workshop on Software Engineering for wearable and pervasive computing*, pp. 431–441, 2000.

[9] A. F. Norcio and J. Stanley, “Adaptive human-computer interfaces: A literature survey and perspective,” *IEEE Transactions on Systems, Man, and cybernetics*, vol. 19, no. 2, pp. 399–408, 1989.

[10] P. Langley, “Machine learning for adaptive user interfaces,” in *Annual Conference on Artificial Intelligence*, pp. 53–62, Springer, 1997.

[11] A. Singhal, “Introducing the knowledge graph: things, not strings,” *Official google blog*, vol. 5, 2012.

[12] H. Paulheim, “Knowledge graph refinement: A survey of approaches and evaluation methods,” *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[13] C. A. Gomez-Uribe and N. Hunt, “The netflix recommender system: Algorithms, business value, and innovation,” *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, p. 13, 2016.

[14] Y. Zheng, B. Tang, W. Ding, and H. Zhou, “A neural autoregressive approach to collaborative filtering,” *arXiv preprint arXiv:1605.09477*, 2016.

[15] G. Linden, B. Smith, and J. York, “Amazon. com recommendations: Item-to-item collaborative filtering,” *IEEE Internet computing*, no. 1, pp. 76–80, 2003.

[16] L. Ehrlinger and W. Wöß, “Towards a definition of knowledge graphs.,” *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, 2016.

[17] L. O. Colombo-Mendoza, R. Valencia-García, A. Rodríguez-González, G. Alor-Hernández, and J. J. Samper-Zapater, “Recommetz: A context-aware knowledge-based mobile recommender system for movie showtimes,” *Expert Systems with Applications*, vol. 42, no. 3, pp. 1202–1222, 2015.

[18] J. Francis, A. Oltramari, S. Munir, C. Shelton, and A. Rowe, “Context intelligence in pervasive environments,” in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 315–316, IEEE, 2017.

[19] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, *et al.*, “Item-based collaborative filtering recommendation algorithms.,” *Www*, vol. 1, pp. 285–295, 2001.

[20] G. Ganberg, J. Ayers, N. Schurr, M. Therrien, and J. Rousseau, “Representing context using the context for human and automation teams model,” in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[21] S. L. Pfautz, G. Ganberg, A. Fouse, and N. Schurr, “A general context-aware framework for improved human-system interactions,” *Ai Magazine*, vol. 36, no. 2, pp. 42–49, 2015.

[22] M. A. Rodriguez and P. Neubauer, “Constructions from dots and lines,” *Bulletin of the American Society for Information Science and Technology*, vol. 36, no. 6, pp. 35–41, 2010.

[23] A. K. Dey, G. D. Abowd, and D. Salber, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” *Human–Computer Interaction*, vol. 16, no. 2-4, pp. 97–166, 2001.

[24] J. M. Zacks and B. Tversky, “Event structure in perception and conception.,” *Psychological bulletin*, vol. 127, no. 1, p. 3, 2001.

[25] G. A. Radvansky and J. M. Zacks, “Event boundaries in memory and cognition,” *Current opinion in behavioral sciences*, vol. 17, pp. 133–140, 2017.

[26] S. T. Iqbal and B. P. Bailey, “Oasis: A framework for linking notification delivery to the perceptual structure of goal-directed tasks,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 17, no. 4, p. 15, 2010.

[27] D. Billman, L. Arsintescucu, M. Feary, J. Lee, A. Smith, and R. Tiwary, “Benefits of matching domain structure for planning software: the right stuff,” in *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 2521–2530, ACM, 2011.

[28] R. L. Baskerville and A. T. Wood-Harper, “A critical perspective on action research as a method for information systems research,” *Journal of information Technology*, vol. 11, no. 3, pp. 235–246, 1996.

[29] M. De Villiers, “Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory,” in *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pp. 142–151, South African Institute for Computer Scientists and Information Technologists, 2005.

[30] J. M. Tappan, D. J. Pitman, M. L. Cummings, and D. Miglianico, “Display requirements for an interactive rail scheduling display,” in *International Conference on Engineering Psychology and Cognitive Ergonomics*, pp. 352–361, Springer, 2011.

[31] H. Beyer and K. Holtzblatt, *Contextual design: defining customer-centered systems*. Elsevier, 1997.

[32] N. J. Cooke, “Varieties of knowledge elicitation techniques,” *International Journal of Human-Computer Studies*, vol. 41, no. 6, pp. 801–849, 1994.

[33] C. Wharton, “The cognitive walkthrough method: A practitioner’s guide,” *Usability inspection methods*, 1994.

[34] K. Charmaz, *Constructing grounded theory*. sage, 2014.

[35] P. Pirolli and S. Card, “The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis,” in *Proceedings of international conference on intelligence analysis*, vol. 5, pp. 2–4, McLean, VA, USA, 2005.

[36] A. Fouse, R. S. Mullins, G. Ganberg, and C. Weiss, “The evolution of user experiences and interfaces for delivering context-aware recommendations to information analysts,” in *International Conference on Applied Human Factors and Ergonomics*, pp. 15–26, Springer, 2017.

[37] P. Cowley, J. Haack, R. Littlefield, and E. Hampson, “Glass box: capturing, archiving, and retrieving workstation activities,” in *Proceedings of the 3rd ACM workshop on Continuous archival and retrival of personal experences*, pp. 13–18, ACM, 2006.

[38] V. Deufemia, M. Giordano, G. Polese, and G. Tortora, “Capturing users interest from human-computer interaction logging,” in *International Conference on Web Information Systems and Technologies*, pp. 312–327, Springer, 2012.

[39] Z. Hinbarji, R. Albatal, N. OConnor, and C. Gurrin, “Loggerman, a comprehensive logging and visualization

tool to capture computer usage," in *International Conference on Multimedia Modeling*, pp. 342–347, Springer, 2016.

[40] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. R. Lipford, and R. Chang, "Recovering reasoning processes from user interactions," *IEEE Computer Graphics and Applications*, vol. 29, no. 3, pp. 52–61, 2009.

[41] S. C. Sundaramurthy, J. Case, T. Truong, L. Zomlot, and M. Hoffmann, "A tale of three security operation centers," in *Proceedings of the 2014 ACM workshop on security information workers*, pp. 43–50, ACM, 2014.

[42] M. R. Endsley, "Design and evaluation for situation awareness enhancement," in *Proceedings of the Human Factors Society annual meeting*, vol. 32, pp. 97–101, SAGE Publications Sage CA: Los Angeles, CA, 1988.

[43] U. Franke and J. Brynielsson, "Cyber situational awareness–a systematic review of the literature," *Computers & Security*, vol. 46, pp. 18–31, 2014.

[44] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins, "A comparison of a graph database and a relational database: a data provenance perspective," in *Proceedings of the 48th annual Southeast regional conference*, p. 42, ACM, 2010.

[45] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*, vol. 520. Addison-Wesley Reading, 2010.

[46] P. Rigaux, M. Scholl, and A. Voisard, *Spatial databases: with application to GIS*. Elsevier, 2001.

[47] R. Snodgrass *et al.*, "Temporal databases," *Computer*, no. 9, pp. 35–42, 1986.

[48] R. Sears, C. Van Ingen, and J. Gray, "To blob or not to blob: Large object storage in a database or a filesystem?," *arXiv preprint cs/0701168*, 2007.

[49] M. Piraveenan, M. Prokopenko, and L. Hossain, "Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks," *PloS one*, vol. 8, no. 1, p. e53095, 2013.

[50] G. Levchuk, K. Pattipati, A. Fouse, R. McCormack, and D. Serfaty, "Active inference in multi-agent systems: Context-driven collaboration and decentralized purpose-driven team adaptation," in *2018 AAAI Spring Symposium Series*, 2018.

[51] A. Duchon, R. McCormack, B. Riordan, C. Shabarekh, S. Weil, and I. Yohai, "Analysis of c2 and c2-lite micro-message communications," in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[52] S. M. Janssen, A. G. Chessa, and J. M. Murre, "Memory for time: How people date events," *Memory & cognition*, vol. 34, no. 1, pp. 138–147, 2006.

[53] M. Fowler, "Event sourcing," *Online, Dec*, p. 18, 2005.

[54] M. Zur Muehlen and J. Recker, "How much language is enough? theoretical and practical use of the business process modeling notation," in *Seminal Contributions to Information Systems Engineering*, pp. 429–443, Springer, 2013.

[55] N. Schurr, A. Fouse, J. Freeman, and D. Serfaty, "Crossing the uncanny valley of human-system teaming," in *International Conference on Intelligent Human Systems Integration*, pp. 712–718, Springer, 2019.

[56] J. M. F. Fernandez and T. Mahlmann, "The dota 2 bot competition," *IEEE Transactions on Games*, 2018.

[57] C. McComb, J. Cagan, and K. Kotovsky, "Lifting the veil: Drawing insights about design teams from a cognitively-inspired computational model," *Design Studies*, vol. 40, pp. 119–142, 2015.

[58] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, IEEE, 2018.

[59] A. Fouse, G. Levchuk, N. Schurr, R. McCormack, K. Pattipati, and D. Serfaty, "Aligning teams to the future: Adapting human-machine teams via free energy," in *International Conference on Intelligent Human Systems Integration*, pp. 471–477, Springer, 2019.

[60] R. R. Hoffman, M. Johnson, J. M. Bradshaw, and A. Underbrink, "Trust in automation," *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 84–88, 2013.

[61] I. Rahwan, M. Cebrian, N. Obradovich, J. Bongard, J.-F. Bonnefon, C. Breazeal, J. W. Crandall, N. A. Christakis, I. D. Couzin, M. O. Jackson, *et al.*, "Machine behaviour," *Nature*, vol. 568, no. 7753, p. 477, 2019.

[62] D. Gunning and D. W. Aha, "Darpa's explainable artificial intelligence program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.