2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)

Enhancing Decision-Making by Leveraging Human Intervention in Large-Scale Sensor Networks

Enrico Casini[1], Jessica Depree[2], Niranjan Suri[1,3], Jeffrey M. Bradshaw[1], Teresa Nieten[2]
[1]Florida Institute for Human and Machine Cognition (IHMC), Pensacola, FL USA
[2]Modus Operandi, Melbourne, FL USA
[3]U.S. Army Research Laboratory, Adelphi, MD USA

***Abstract*—Extensive deployment of sensor networks in recent years has led to the generation of large volumes of data. One approach to processing such large volumes of data is to rely on parallelized approaches based on architectures such as MapReduce. However, fully-automated processing without human intervention is error prone. Supporting human involvement in processing pipelines of data in a variety of contexts such as warfare, cyber security, threat monitoring, and malware analysis leads to improved decision-making. Although this kind of human-machine collaboration seems straightforward, involving a human operator into an automated processing pipeline presents some challenges. For example, due to the asynchronous nature of the human intervention, care must be taken to ensure that once a user-made correction or assertion is introduced, all necessary adjustment and reprocessing is performed. In addition, to make the best use of limited resources and processing capabilities, reprocessing of data in light of such corrections must be minimized. This paper introduces an innovative approach for human-machine integration in decision-making for large-scale sensor networks that rely on the popular Hadoop MapReduce framework.**

Keywords: sensor data processing; sensor networks; human-machine teamwork; human-in-the-loop architectures; human-assisted architectures; data pipelines; decision-making; MapReduce; Hadoop

I. INTRODUCTION

The ability to make decisions based on the best available information at the tactical edge is critical to successful mission accomplishment. Navy and Marine Corps units at the lowest tactical levels often operate in disconnected or disadvantaged network environments with spotty connectivity. This requires that nearly all of their decision-making be based on local data that is immediately available to them.

To provide customized data for decision-making in support of tactical operations, we propose a pipeline-inspired architecture that combines machine data processing with human assistance. This architecture allows for much of the sorting, analysis, and predictions to be done in the cloud. Such an approach will significantly enhance field decision-making by empowering commanders with locally adapted, semantically processed data while retains human involvement in selecting among options.

In order to achieve this level of fidelity with current technology, disconnected and disadvantaged users must rely on briefs and products from higher echelons or direct sensor feeds and reports which, while containing much of the information they need, also inevitably includes potentially large volumes of irrelevant results. By storing and processing this information in the cloud and delivering results to tactical units on an on-demand basis, commanders can quickly obtain access to data that normally would not be available until later, when a link-up with higher echelons becomes possible. Of course, in order to validate this approach, architectures like the one proposed would need to be employed in multiple tactical scenarios in Field User Evaluations (FUEs) in order to both prove the concept as well as to determine where and when tactical units gain the greatest advantage for the delivery of this information.

Other markets of interest, outside the DoD, include the Department of Homeland Security and state-level emergency management organizations. Allowing individual first responders to tailor the data they receive while executing time critical operations enhances their ability to make decisions using the most relevant information available without requiring them to sort through mounds of immaterial data.

A relevant tactical scenario for the evaluation of this architectural approach is that of named-entity recognition, with the ultimate goal of tracking a specific high-value individual (HVI). Sightings of HVI's are reported in intelligence reports derived multiple sources of potentially conflicting information. It is here that people can make a substantial difference by weighing the conflicting reports, relying on additional sources of personal expertise as well as strengths in inductive reasoning and the exercise of context-sensitive judgment.. The ultimate goal is to predict the HVI's next move with human-vetted results. A reasonable dataset for this evaluation can be a corpus of HVI-centric unclassified data, mixed with publicly available news reports. Further demand for research in this area is experiencing a strong momentum, especially at the tactical edge where interests in big data analysis and the leveraging of rich, real-time information are of prime importance.

Researchers in this field specify that the standard design process for developing human-machine approaches either starts with a human approach and enhances it with decision-support or starts with an automated approach and enhances it with operator input. We are introducing a mixed-initiative, pipeline-based approach that incorporates the best of both worlds. The aim of our approach is increase performance and throughput in the automated processing and delivery of the data throughout

978-1-4799-8015-4/15/$31.00 ©2015 IEEE

the pipeline while also providing the advantages of human participation at key intervention points along the pipeline through intuitive user interfaces.

II. RELATED WORK

The concept of automation—which began with the straightforward objective of replacing whenever feasible any task currently performed by a human with a machine that could do the same task better, faster, or cheaper—became one of the first issues to attract the notice of early human factors researchers. Pioneering researchers such as Paul Fitts attempted to systematically characterize the general strengths and weaknesses of humans and machines [1]. The resulting discipline of *function allocation* aimed to provide a rational means of determining which system-level functions should be carried out by humans and which by machines.

Obviously, however, the suitability of a particular human or machine to take on a particular task may vary over time and in different situations. Hence, early research in *adaptive* function allocation and adjustable autonomy was undertaken with the hope that shifting of responsibilities between humans and machines could be made dynamic, in those situations where human and machine capabilities for a given task overlap [2] [3] [4].

Eventually, however, it became plain to researchers that things were not as simple as they first appeared. For example, many functions in complex systems are shared by humans and machines; hence the need to consider synergies and conflicts among the various performers of joint actions. Moreover, it has become clear that function allocation is not a simple process of transferring responsibilities from one component to another. Automated assistance of whatever kind does not simply enhance our ability to perform the task: it changes the nature of the task itself [5].

As automation becomes more sophisticated, the nature of its interaction with people will need to change in profound ways. In non-trivial interaction of this sort, the point is not to think so much about which tasks are best performed by humans and which by automation but rather how tasks can best be shared by both humans and automation working in concert. In 1960, Licklider called this concept *man-computer symbiosis* [6]. To counter the limitations of the Fitts' list, which is clearly intended to summarize what humans and machines each do well on their own, Robert Hoffman has summarized the findings of David Woods in an "un-Fitts list" [7], which emphasizes how the competencies of humans and machines can be enhanced through appropriate forms of mutual interaction. Of course, certain tasks, such as those requiring sophisticated judgment and nuanced assessment of situations and contexts, cannot be shifted to machines, and other tasks, such as those requiring ultra-precise calculations and high-tempo operations on large volumes of data, cannot be performed by humans. But we believe that there are significant limitations to the current "automation only" approaches that can be addressed only by human-machine teamwork.

Over a number of years, we have identified several cross-cutting requirements for successful, resilient human-machine teamwork [5]. The first two requirements are *observability* and *directability*. Lack of *observability* affects our ability to understand and evaluate what is currently happening in the world, while lack of *directability* limits our ability to implement our goals for what we want to happen in the future. Additional cross-cutting requirements, *predictability* and *learning*, are closely related to each other. All these requirements provide valuable design guidance as different options for implementation are considered.

In addition to these cross-cutting requirements, the specific requirements of the tasks must be addressed. These are addressed through a process we call "coactive design" [8]. Coactive design recognizes that the underlying interdependence of participants in joint activity is a critical factor in the design of human-machine systems. The term "coactive" highlights the fact that both humans and machines mutually provide active assistance to improve system performance, often in close and continuous interaction.

In order to "design for interdependence," we have sought to analyze specific ways in which we can exploit all human capabilities — sensing, decision-making, acting — to assist machines, and vice versa. These constitute the opportunities for human intervention that are discussed above. Having identified opportunities for human intervention, the next step in the design process is to modify algorithms and design user interfaces from the ground up to make partial results observable, predictable, and adaptive for each of these cases of human intervention.

III. HUMAN INTERVENTION MODALITIES

In our design, we developed three specific forms or modes of human intervention:

a) System asks the Human Operator for Clarification: This intervention mode captures the situation described earlier, where the system identifies documents and intermediate results with confidence values that fall below a configured threshold. These documents are queued up in an asynchronous "inbox" for introspection by the operator. The operator may examine these documents and provide the necessary feedback and oversight that would improve the accuracy of the processing outcome for the documents analyzed. An important aspect of this architectural approach is asynchronous processing of human and machine contributions to decision-making. The system does not demand that a human operator immediately respond to any specific request and the human operator does not hold up the processing of the system by taking time to examine and determine the outcome of specific assumptions.

b) Random Inspection by Human Operator: The second intervention mode involves random inspection by human operators into any of the intermediate results of document analysis or on the conclusions/assertions generated by the Hadoop algorithms. This mode of interaction is analogous to statistical sampling in a production-line, where the operator randomly selects and examines documents and generated output to evaluate the accuracy of the automated algorithms. At any point in this process, the operator may correct the

system's assumptions and conclusions and insert them back into the system. These corrections may result in the system re-evaluating other documents that have been processed that are affected by the changes. Once again, an important aspect of this interaction mode is the asynchrony – the system does not have to wait for the operator and the operator does not have to be continuously monitoring the system.

c) Human Operator Drill-down: The third intervention mode involves an operator choosing to inspect the processing chain that led to specific assertions and conclusions. While performing a query or other analysis operation, the operator may question an assertion or conclusion in the system and examine the evidence chain that leads back to the original documents that were ingested by the system. Upon examining this chain, the operator may either accept these conclusions (which is fed back into the system in order to increase its confidence values) or correct the intermediate results or final assertions. Once these corrections are entered into the system, the system will re-evaluate any other documents that might be affected by the changes.

The architecture and implementation details described in the following sections summarize the system that we have designed to cover all the human intervention scenarios explained above.

IV. ARCHITECTURE

A. Hadoop

Apache™ Hadoop® [9] is a software framework that enables the developer to analyze and transform very large data sets using the MapReduce programming model [10]. What makes Hadoop's architecture and paradigm relevant in big data analysis and decision-making is the partitioning and computation of the data itself across many (potentially thousands) of hosts, while executing application computations in parallel close to their data. Because of this fundamental characteristic, a Hadoop cluster is easily able to scale its computation capacity, storage capacity, and IO bandwidth by simply adding commodity servers. Within the proposed pipeline-inspired architecture, Hadoop MapReduce is being fed with data consisting of unstructured or semi-structured text documents, sensors or other formatted data, and human operator entry. The architecture provides for the data to be streamed into the Hadoop cloud and partitioned to the applicable processing resource(s) in batch mode. Operators have the opportunity to interrogate individual data points during this processing in an interactive way. We will provide more details about the individual tasks of this modality in the implementation section.

B. HDFS

HDFS [11] is the distributed file system component of Hadoop. HDFS is part of the family of other popular distributed file systems (PVFS, Lustre and GFS) [12], specifically designed to store metadata and application data separately. Each node in a Hadoop instance typically has a single node called "name node". Name nodes are responsible for handling the metadata that describes the data; all the other nodes of the cluster are responsible for the actual data to be processed and are generally referred to as "data nodes"; a cluster of data nodes form the HDFS cluster. The architecture of an HDFS cluster provides for these data nodes to serve up blocks of data over the network using HDFS's block protocol. This is done taking advantage of the TCP/IP network layer as a transport for communication between the nodes.

HDFS was specifically designed to be able to store very large files (typically in the range of gigabytes to terabytes of data) across multiple machines. This fits the requirement of highly modular architecture that keeps the dataset as consistent as possible (for a distributed file system) within a given unit of time. One of the other main requirements of our architectural approach is for the human to be able to intervene on the assertions while the algorithm of the MapReduce job is still processing the remaining documents of the dataset. The batch mode of HDFS and Hadoop MapReduce allows for live scans of the data, giving the human operator the opportunity to look across the data in the HDFS prior to committing it to a consolidated data store.

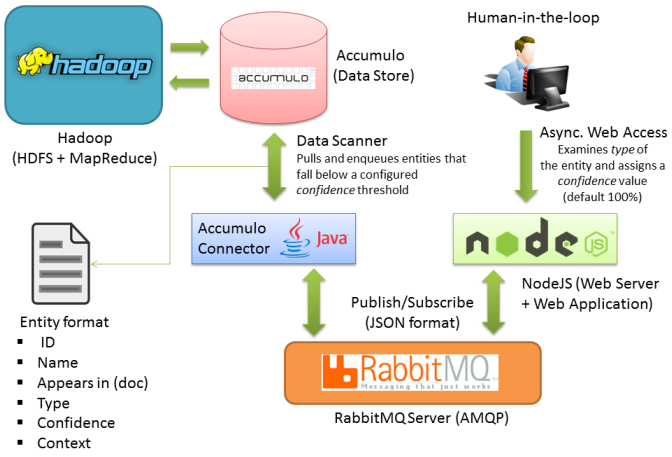

Fig. 1. Global view of the system's architecture

C. Accumulo

Apache™ Accumulo [13] is a data storage and retrieval system that provides a sorted, distributed key/value store based on the BigTable technology from Google [6]. BigTable is a distributed storage system that is designed to scale up to petabytes of data across thousands of commodity servers. Its architecture makes use of a tabular key–value store, in which each key is a pair of strings corresponding to a row and column identifier. These records are lexicographically sorted by row key, and rows are distributed across multiple database servers. Efficient record ingestion and read are ensured through this mechanism of sorted records, even for a small range of rows, independently of the quantity of data stored.

Accumulo and the others BigTable-like distributed databases provide a storage solution for data-intensive applications, making trade-offs between performance, scalability, and data consistency. It is known that traditional Relational Database Management System (RDBMS) based on

Structured Query Language (SQL) aim to provide atomic transactions and data consistency, a fundamental requirement for many applications. On the other hand, BigTable uses a relatively new "NoSQL" [14] approach that relaxes these transaction requirements, guaranteeing only eventual consistency while tolerating old or approximate data within the update process. It is worth mentioning though, that since new NoSQL databases lack the mature code base and rich availability of features of established RDBMS solutions, the process of designing and optimizing performance-sensitive queries must be taken care of by the application developer.

D. Node.js

Node.js™ [15] is a software platform designed to develop fast and scalable networking and server-side applications. The Node.js runtime executes JavaScript code inside a particularly efficient VM, the Google V8 JavaScript engine [16]. A large percentage of its basic modules are written in JavaScript. For these reasons and the easy portability of the runtime (Node.js applications can run on Windows, Mac OS X and Linux with no changes), its popularity in industry and research has been increasing steadily.

Furthermore, Node.js proposes an event-driven API that suits the development of network applications designed to maximize throughput and efficiency. The latter is achieved by design, using a non-blocking I/O model and asynchronous events. In fact, even though the underlying core uses multiple threads for file and network events, Node.js applications run single-threaded, hiding the complexity of multithreaded code and lock management from the network programmer. Due to these design choices and its asynchronous nature, Node.js is best suited for I/O bound and real-time applications, being able to scale up to millions of concurrent requests. Given its built-in support for asynchronous I/O, sockets, and HTTP communication, Node.js can also act as a traditional web server without the need of any additional modules. For all of these reasons, Node.js fits our needs perfectly to realize a web application that enhances document evaluation and decision-making by allowing human intervention. A brief analysis of the requirements shows that the web application needs to scale up to potentially thousands of injected documents per second, while the server-side application requires the handling of multiple concurrent requests by different operators.

E. RabbitMQ

RabbitMQ™ [17] is an open source message-oriented middleware (or message broker) implementing the Advanced Message Queuing Protocol (AMQP) [18] and providing a reliable, guaranteed and in-order message delivery. AMQP is an open standard application layer protocol for message-oriented middleware that supports message orientation, queuing and routing (including point-to-point and publish-and-subscribe). The architecture of any AMQP-compliant middleware like RabbitMQ consists of three main components: Publisher(s), Consumer(s) and Broker/Server(s) [19]. Each component can be replicated in number and situated on independent nodes. Publishers and Consumers communicate with each other through message queues bound to exchanges within the Brokers.

For the purpose of this paper, RabbitMQ was essential to guarantee a transparent delivery of data messages between the Accumulo data store and Node.js due to the heterogeneity of the two platforms in terms of programming languages (Java and JavaScript respectively).

V. IMPLEMENTATION DETAILS

The task of developing the architecture involves determining the data flow through the system, the components involved, and where the human intervention opportunities will be. As stated in the previous sections, data enters the system through unstructured or semi-structured text documents, sensors or other formatted data, and human operator entry. The development of this highly interconnected system involved all the components described in the architectural view sections, adapting the input/output of each component to fit this complex pipeline. On one extreme end, also identifiable as the source of the pipeline, the Hadoop MapReduce framework has being leveraged for the phase of document processing. The first part of the processing pipeline has been created wherein document sorting and information extraction - tokenizing, part-of-speech tagging, named entity recognition and disambiguation, etc. - are all performed in a highly modular and distributed format, through which the entire pipeline is scalable to large volumes of information. The developed architectures provides for all of the listed tasks to be performed across a large body of unprocessed text and entirely in parallel.

This high modularity requirement is achieved through a separation of the processing phases, in which each listed step is being performed by a different MapReduce job. This modular subdivision of the information extraction process allows for fine control; at each step in the MapReduce pipeline, the intermediate data are stored on the Hadoop Distributed File System (HDFS) and therefore present opportunities for human intervention. Through this continuous evaluation and integration process that involves the human, we aimed to improve the global accuracy while reducing the time necessary for the entity recognition task when processing huge amounts of data.

All assertions made about the documents within the processing pipeline are assigned confidence values at the time of extraction; every assertion below a defined certainty threshold gets ranked according to uncertainty and successively sent out to the following component of the pipeline. This is accomplished by adding an additional output to several of the MapReduce jobs, containing the machine-performed assertions ranked by confidence. The ranked assertions are periodically scanned and read from the Accumulo database through a Java connector implemented using the available public API. Thanks to this component, the assertions are then pushed to RabbitMQ that stores them in the appropriate queue.

Once the data is in the queue, the RabbitMQ process is triggered and its service (running in background all the time) takes care of pushing them to the instance of Node.js subscribed to the queue. This is done by taking advantage of the simple and performant publish/subscribe mechanism provided by RabbitMQ. The Node.js module is, by design, optimized to maximize throughput with its non-blocking I/O model. Data is received asynchronously from one pipe and sent

to another. The Node.js module implements an HTTP server that is responsible for serving this live data to a web client, through WebSockets via Socket.IO [20]. This end of the pipeline ultimately allows the human operator to examine their veracity and provide corrections, if necessary. The web interface for the human operates is itself built with HTML5, JavaScript and jQuery [21].

Assertions are presented to the user by an interface in order of reverse confidence; that is, the user is presented with low-certainty assertions first as more of the user's time is likely to be spent on those. When a change on one or more assertions is made, the human operator can decide to send the changes back to Accumulo in a one by one fashion or all at once. The format chosen for the data-interchange is JSON (JavaScript Object Notation) [22], allowing the exchange of data between the Node.js module and the Accumulo API. After a JSON data transformation, the Node.js module pushes the restricted set of assertions that the human operator modified back to Accumulo through the Java API connector.

All the changes made by the human operator are executed in parallel while the MapReduce pipeline is still operational on other documents and entities. At each human intervention opportunity, the data is not required to be examined; if a user never offers any sort of input, the processing of both the unexamined document and subsequent information will continue. If corrections are provided, they will be persisted both to the HDFS (where the document's representation will be updated and any necessary reprocessing will be performed), and Accumulo, the datastore where all salient assertions are ultimately persisted. Accumulo takes care of updating the dataset on which the MapReduce jobs are operating in an asynchronous and consistent way. Possible interventions made by the human operator will also influence the capacity of the MapReduce entity recognition task, increasing its precision and confidence during the disambiguation phase.

VI. EXPERIMENTAL RESULTS

A. Experimental Design

A testing corpus of 60 documents was selected from news sites. The gold data set was constructed by randomly sampling 20% of the corpus at the sentence level. This body of text was manually annotated with named entity tags for person entities. The corpus consisted of articles from online news sources, specifically Al Jazeera. Testing was performed on a single-node Hadoop Cluster running on a CentOS 6.3 virtual machine.

A series of experiments was then conducted using the gold data set. The following phrases are used to describe stages that occurred in each experiment.

Statistical Entity Extraction: Named entity extraction was performed by the statistical OpenNLP name finder, which uses a maximum entropy model.

Dictionary Tagging: Extracted entities are added to a dictionary, which is then used with the OpenNLP dictionary name finder to locate and tag missed names. This allowed some degree of name recognition to step in when entities appeared in less obvious contexts and statistical recognition failed. For instance, if the name "Bashar al-Assad" were found in the statistical step, all instances of the sequence "Bashar al-Assad" would be tagged as an entity in the dictionary step. The type of the dictionary-recognized entity was assigned to match the original dictionary entry.

Name Expansion: Extracted character sequences were divided into substrings so that the individual components of the name (i.e. first, middle, and last names) could be extracted. For instance, a person entity named "John Kerry" found in the first pass would result in the tagging of non-consecutive instances of both "John" and "Kerry" as person entities in the second pass.

Human Intervention: Prior to the dictionary tagging stage, a user was allowed to vet entities for accuracy. In this experiment, 199 person entities were reviewed and, if appropriate, corrected.

Experiment 1 (Baseline) : + Statistical Entity Extraction
Experiment 2: + Statistical Entity Extraction + Dictionary Tagging - Name expansion - Human Intervention
Experiment 3: + Statistical Entity Extraction + Dictionary Tagging + Name expansion - Human Intervention
Experiment 4: + Statistical Entity Extraction + Dictionary Tagging - Name expansion + Human Intervention
Experiment 5: + Statistical Entity Extraction + Dictionary Tagging + Name expansion + Human Intervention

B. Experimental Results

The results of the experiments are shown in Table 1. Experiment Results

.

Table 1. Experiment Results

Experiment	Precision %	Recall %	F-Measure %
Experiment 1	79.79	40.11	53.39
Experiment 2	50.64	42.24	46.06
Experiment 3	50.31	43.85	46.86
Experiment 4	80.61	42.24	55.39
Experiment 5	80.39	43.85	56.75

C. Discussion

The baseline, using the default OpenNLP named entity extraction models, showed reasonable precision and low recall. No research or effort went toward improving the baseline accuracy, as our goal is merely to show the effect our distributed pipeline and human vetting had on the results.

When found entities were tagged elsewhere in the corpus, as in Experiments 1-4, the recall increased as missed entities

(of the same name) were found. However, the precision dropped significantly. Random inspection of the results indicate that some common words had been tagged in the statistical extraction stage as entities, and these wrongfully tagged entities were subsequently perpetuated throughout the corpus by the dictionary tagging stage. When we added in the name expansion, allowing multi-word names to be annotated as parts, the recall improved yet again, but the precision went down slightly; this can be accounted for in the same way as the initial precision drop.

The human vetted results improved over the baseline. They yielded similar or greater increases to recall without the loss of precision. This is because the user removed the entities that had been wrongly tagged in the statistical extraction stage, preventing them from being wrongfully tagged throughout the remainder of the corpus. The gains to precision will be small, because there are few wrongfully tagged entities; these results show a system that is targeted at improving recall. At this stage, the user is merely certifying whether or not something is an entity, which allows the boost in recall without the hit to precision that would result in non-entities being tagged throughout the corpus. Further, there is an upper bound to recall gains, as our user interface does provide away to suggest or tag entities which were missed entirely.

The overall testing showed a measurable improvement in F-measure and recall with human vetting over non-vetting with a single intervention point currently implemented. It allows for the recall gain of the automatic extraction system without lowering precision. The measured accuracy of our methods was highly dependent on corpus size. Large amounts of data are paramount to any statistical extraction system, and the small corpus made some of our measures unstable. Further, the efficacy of our system is directly proportional to the interrelatedness of the reports; our design relies on our entity reoccurrence throughout a corpus.

VII. CONCLUSIONS AND FUTURE WORK

The architecture proved to be effective in several of its design goals. It allowed for the effective insertion of a human operator into a massively distributed, high-volume data environment; the human's contributions had a noticeable effect on the overall accuracy of the system. The system made efficient use of the operator's input without being wholly dependent on human intervention – the system never waits. Finally, the architecture valued the human-in-the-loop over computational resources, but still limited the amount of time spent on redundant and unnecessary reprocessing.

Moreover, the results demonstrated a concept fundamental to the design of any multistage decision system: error propagation. Each assertion made automatically by the computer will be propagated by the system, creating a snowball effect of incorrect information. The experiments in which entity name expansion occurred without human vetting showed just how detrimental to a system's accuracy this can be. The minimal interaction of a human with the pipeline at each stage of processing can dramatically increase a system's accuracy by preventing error propagation.

ACKNOWLEDGMENT

This material is based upon work performed under Office of Naval Research SBIR Contract No. N00014-13-O-1178. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Office of Naval Research.

REFERENCES

[1] P. M. Fitts, "Human Engineering for an Effective Air Navigation and Traffic Control System," W Washington, DC: National Research Council, 1951.

[2] J. M. Bradshaw, M. Sierhuis, A. Acquisti, P. Feltovich, R. Hoffman, R. Jeffers, D. Prescott, N. Suri, A. Uszok, and R. Van Hoof, "Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications," in *Agent Autonomy,* edited by Henry Hexmoor, Rino Falcone and Cristiano Castelfranchi, 243-80. Kluwer, 2003.

[3] J. M. Bradshaw, P. J. Feltovich, H. Jung, S. Kulkarni, W. Taysom, and A. Uszok, "Dimensions of adjustable autonomy and mixed-initiative interaction," in Agents and Computational Autonomy: Potential, Risks, and Solutions, (M. Nickles, M. Rovatos, and G. Weiss, eds.), pp. 17–39, Berlin/Heidelberg: Springer, 2004.

[4] G. A. Dorais, R. P. Bonasso, D. Kortencamp, B. Pell, and D. Schreckenghost, "Adjustable autonomy for human-centered autonomous systems on Mars," in First International Conference of the Mars Society, 1998.

[5] J. M. Bradshaw, P. Feltovich, and M. Johnson, "Human-Agent Interaction," in Handbook of Human-Machine Interaction, edited by Guy Boy, in press. Ashgate, 2011, pp. 283–302.

[6] J. C. R. Licklider, "Man-computer symbiosis," *IRE Transactions in Electronics. New York: Institute of Radio Engineers.* (1960): 4-11.

[7] R. Hoffman, P. Feltovich, K. M. Ford, D. D. Woods, G. Klein, and A. Feltovich, "A rose by any other name… would probably be given an acronym," *IEEE Intelligent Systems*, July-August 2002, 72-80.

[8] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. van Riemsdijk, and M. Sierhuis, "Coactive design: Designing support for interdependence in joint activity". *Journal of Human-Robot Interaction*, Vol. 3, No. 1, 2014, pp. 43-69.

[9] Apache Hadoop, URL hadoop.apache.org

[10] MapReduce, Wikipedia URL en.wikipedia.org/wiki/MapReduce

[11] HDFS, URL hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[12] M. Varade, V. Jethani, "Distributed Metadata Management Scheme in HDFS," in International Journal of Scientific and Research Publications, Vol. 3, Issue 5, 2013.

[13] Apache Accumulo, URL accumulo.apache.org

[14] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," ACM Transactions on Computer Systems (TOCS), vol. 26, no. 2, p. 4, 2008.

[15] NoSQL, Wikipedia URL en.wikipedia.org/wiki/NoSQL

[16] Node.js, URL nodejs.org

[17] The V8 JavaScript engine, code.google.com/p/v8.

[18] RabbitMQ, URL rabbitmq.com

[19] AMQP, URL amqp.org

[20] SocketIO, URL socket.io

[21] jQuery, URL jquery.com

[22] JavaScript Object Notation, JSON, URL json.org