ACTA SCIENTIFIC MEDICAL SCIENCES

Volume 3 Issue 3 March 2019

Research Article

An AI and IoT Multi-Tiers Architecture for Real Time Integration and Analyse of Data Sensor

Francesco Rago*

Megatris Comp. LLC, USA

***Corresponding Author:** Francesco Rago, Megatris Comp. LLC, USA.

Received: December 06, 2018; **Published:** February 04, 2019

Abstract

We have realized a new architecture for AI allowing an increase in capabilities and innovation in the Internet of Things (IoT). IoT applications are a major source of data and Big Data is a consequence. This challenge call for more intelligent computing models (Digital Agent, Neural Network, Deep learning, …) that enable rapid innovation for applications and service delivery. The scope of our work is to realize an Internet of Things (IoT) architecture to delivers fast-moving data from sensors and devices around the world making sense of all that data using artificial intelligence (AI). Digital Agents are used as a framework for modeling, understanding and reasoning about data. They operate for multisensory integration to adapt behavior for a perceptual integration of data in the context. The architecture is triggered by events and internal processes provide deep analysis on data using an R machine. Each Digital Agent has multiple cognitive/perceptual layers with increasing degrees of abstractions. WE design the architecture with top-down and bottom-up mechanisms working together to connect the cognitive representations to the perceptual data. We require a Cognitive Layer (CL) and a Perceptual Layer (PL) with their own autonomous structures.

A PL has a Receptive field as a particular region of the sensory space in which a stimulus (event) will modify the firing of the input neuron. In our model the receptive field is a set of sensorial point in a context. Receptive fields' points to interpretative object called place cell. A place cell is a kind of a neuron in a SOM (Self-Organizing Map). Place cells are thought, collectively, to act as a cognitive representation of a specific location in CL space, known as a cognitive map. The structure of CL reflects the cognitive associations that we normally acquire through experience; and the structure of the perceptual layer reflects the historical series of input data. There is an autonomous memory for each layer where the associations or structures of the respective layers are stored.

Keywords: AI; AI Architecture; Digital Agent; Neural Network; Semantic Network; Rules; AI Architecture Measurement

Abbreviations

IoT: Internet of Things; IoE: Internet of Everything; SSB: Smart System Bus; DA: Digital Agents.

Introduction

The new architectures for AI realize a processing system that allows an increase in capabilities and innovation in the Internet of Things (IoT). In a world where almost anything can be connected to the Internet, the exponential increase in the volume of information and connected devices creates a dilemma. In these cases new intelligent technology solutions are an enabling opportunity for innovation in Internet of Things (IoT). These challenges call for more intelligent computing models (Digital Agent, Deep Learning, Semantic Networks,…) that permits rapid innovation for applications and service delivery.

IoT applications are considered to be a major source of big data and are supported through cloud architectures where data is stored and processed.

The critical challenge is using this data when it is still in motion, extracting valuable information from it. Organizations are scrambling to apply tools and analytics to these streams of data before the data is stored for post-event analysis because it is necessary to detect patterns and anomalies while they are occurring, in motion, in order to have a considerable impact on the event outcome.

Our Research Platform

For real-time decision making on data, we have developed an architecture triggered by events that process and provide deep analysis on data. Event processing uses the following techniques to manage, and make sense of streaming data:

1. Assessment, applying transformations and rules to determine if further processing needs to occur or the data (or event) can be quickly discarded.
2. Analysis, time series, analysis generated by an event stream processing, can be continuously processed to understand real-time trends.
3. Correlation, event stream processing allows to connect to multiple streams of data and identify that series of events occurred.
4. Digital Agents, (DA) Neural Networks (NN) and other AI Tools are vital for modeling, understanding and reasoning about data.

A Digital Agent can be considered as the universal primitives of digital computation [1] while NN are connectionist computing systems vaguely inspired by the biological neural networks. Such systems learn tasks by considering examples, generally without task-specific programming.

Finally, we have used software metrics to expand our standards with new ones to try to define a rationale for software process design. As well know, a software metric is a measure of a degree to which a software system or process possesses some property [2-4]. The property we had essentially in mind was performance to improve algorithms and to create recipe to use the different AI tools. Therefore, AI measurements are highly demanded, but such work is still in its infancy in industrial environments.

Research Contribution

This research has explored the use of maps onto maps inspired by neuroscience. As is well known nervous system is a set of modules of neurons that is able to ignite a many layers cognition. Even if we used supervised neural network we were able to create a full circuit that from the analysis of IoT data generate to an answer through cognitive steps.

Related works on cognitive architectures

A cognitive architecture is a generic computational model to study systems behavior and cognition. It provides agents with decision-making mechanisms.

Among the most known cognitive architectures we have: SOAR [5,6]; CLARION [7] and ACT-R [8].

SOAR is the most known and includes working and long-term memory, and learning mechanisms (chunking, reinforced knowledge, etc.).

John E. Laird proposed a standard cognitive architecture to provide the appropriate computational abstraction for defining a standard model, although the standard model is not itself such an architecture [9-13]. The standard model spans key aspects of structure, processing, memory, learning and perception and motor.

We started our work from Laird one, even if we have modified the standard depending on neuroscience approaches developed these years.

The purpose of architectural processing is to support bounded rationality, not optimality. System behavior is driven by sequential action selection via a cognitive cycle.

We think that complex behavior arises from a sequence of independent cognitive cycles that operate in their local context, but separate architectural modules for global optimization and planning are necessary creating more independent layers in a logic of maps of maps as in natural brains.

Declarative and procedural long-term memories are contained in neural networks and not in symbols structures and associated quantitative metadata.

Global control is provided by a cognitive map realized by a neural network and procedural long-term memory is composed of rule-like conditions and actions. Rules are necessary to produce control actions for IoT environment.

Learning updates cognitive maps for specific contexts and it occurs online and incrementally, as a side effect.

Perception and Motor Control is the key of a working IoT architecture. We can have many different such perception modules, each with input from a different modality and its own buffer. This is prepared “ad hoc” for each smart object or mobile connected device. Motor control converts internal states into external actions.

Neuroscience suggests connectionist models as basis for cognitive architecture. It considers processing as the dynamic and graded evolution of activities in a neural net module. Each unit's activation depends on the connection strengths and activity of its neighbors, according to the activation function.

The consequence of all these assumptions is a hybrid connectionist architecture. Elements of classical symbolic processing are included in neural nets, Wermter and Sun [14]. We realized a collection of neural net modules that share data coded in activation patterns following Miikkulainen [15]. We used the best of the two worlds as the hybrid models combines both symbolic and sub-symbolic approaches with the rising paradigm represented by connessionistic machine learning methods, such as deep learning, which have found enormous practical success in limited domains, Kotseruba [15].

AI architecture for IoT

We are going to describe now our DA based architecture. It is devoted to manage complex environment and it is structure to give a cognitive taste to old fashioned Command, Control and Communication systems.

Complex Environment management

AI systems permit the transition from Command, Control and Communication systems to Mission Management Systems where there is a requirement for always more unmanned management.

We have the following transitions:

- The Human System Interface is no more a cockpit Aide but it becomes a decision aide permitting operator empowerment;
- Sensing has an evolution from events collection to a Dynamical Threshold Management with a choice of phenomena of interest;
- Monitoring and Diagnosis process uses Models leaving event-action schema for a more complete semantic of events.
- Decisions are simulation and model based; statistics support Proactive Decisions.

A summary of the software architecture

We give a rapid sketch of the architecture flow.

Smart objects and mobiles communicate with cloud servers using micro-services. Mobiles facilitate humans with an interface totally based onto Natural Language. The app user experience and user interface (UX/UI) is made to facilitate vocal and written communication.

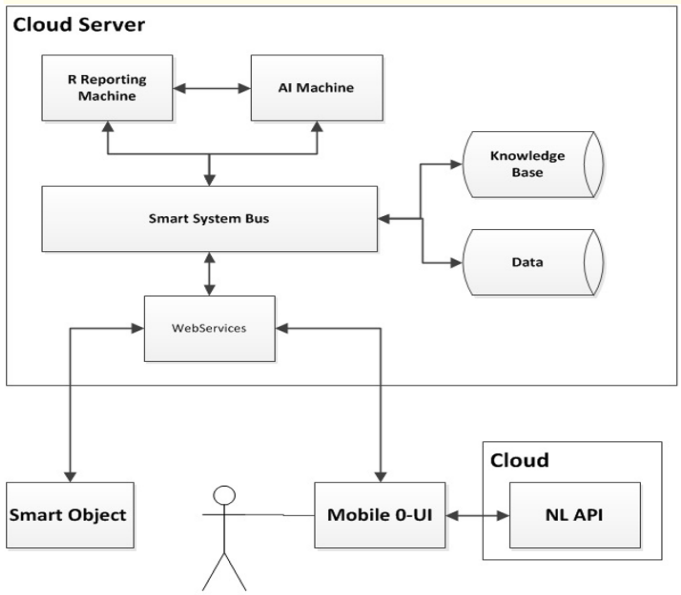

Figure 1: Architecture flow.

Web services actuate requests to SSB (Smart System Bus). SSB drives the requests to the appropriate software cloud machines.

Modifying the Natural Language content is easy to associate cloud applications to a mobile or to any IoT device.

Smart System Bus (SSB): Scope and Functions

SSB is a software system born in the new wave of IT Technologies related to Cloud Computing. It collects events from sensors and it is able to collect, cluster and support decisions, sometime in unmanned way.

The main functions of the SSB system are:

- Events management
- Communication management
- Actions management depending on Decision Models.

SSB has architecture with one or more servers and controllers. Controller can be installed everywhere. Controllers can be any type of distributed hardware mobiles enclosed.

SSB can send feedback controls to the environment. It aggregates large numbers of sensors with a fusion of heterogeneous in-

formation from many sources. Many instrument resources have a continuous technological evolution and this characteristic causes continual re-invention of driver software. SSB middleware has as primary design goal to facilitate integration of instruments into current computing to leverage Cloud-based services.

AI Machine

An AI machine is based on Digital Agent as the universal primitives of digital computation [1]. All physically possible computation can be directly implemented using Digital Agents.

Message passing using types is the foundation of system communication. When a Digital Agent receives a message, it can concurrently: sends/receives messages to/from other Digital Agents and creates new Digital Agents to solve a problem.

Digital Agent Model can be used as a framework for modeling, understanding, and reasoning about, a wide range of concurrent systems.

Information integration needs to make use of the following information system principles:

- **Persistence:** Information is collected and indexed.
- **Concurrency:** Work proceeds interactively and concurrently, overlapping in time.
- **Pluralism:** Information is heterogeneous, overlapping and often inconsistent. There is no central arbiter of truth.
- **Provenance:** The provenance of information is carefully tracked and recorded.

Digital Agents

Digital Agents are used as universal primitives of concurrent digital computation, have beliefs, desires and intentions. They have states which can be nested, so that, (for example), one agent is able to have beliefs about another agent's intentions. Agents communicate using asynchronous SSB standard messages architecture.

We can define formally a Digital Agent or Computing Agents as models of processes, which are essentially sets of traces representing possible complete sequences of actions performed by an agent and its environment. Such a model provides a firm basis for reasoning about phenomena.

Formally, Digital Agent or Computing Agents are modeled as processes, which are essentially sets of traces representing possible complete sequences of actions performed by an agent and its environment.

Such a model provides a firm basis for reasoning about phenomena.

Digital agents are objects which may perform various actions, thus exhibiting some discrete behavior, and this behavior may be influenced by the actions of other agents. An action performed by the computing agent itself is an output action, while an action performed by some exterior agent is an input action.

We shall use traces to represent sequences of actions. A trace is a sequence of symbols taken from some alphabet, which is just a set of symbols. The set of all finite traces formed from alphabet A is written A*. The catenation of two traces t and u is written as the juxtaposition of the two, i.e. combine (t, u). In our internal representation we assume that the continuous interaction of an agent and its environment is summarized by a discrete concatenation of view-action sequence of the form: v_0, a_0, v_1, a_1, an, v_n. A view represents a sensory description associated with a context state.

Agent declarations and initialization

Agents are declared via the agent keyword, followed by the agent's name (which must be unique), and the body of the agent. At startup, agents are invoked in the order in which they are declared; an agent terminates when it reaches the end of its code body or if terminated. Agents can communicate the results of their actions.

Beliefs, desires and intentions

Agents have explicit data structures corresponding to beliefs, desires, and intentions. These states can be nested, so that (for example) an agent can have beliefs about other agent's intentions.

Agent's beliefs are the information it has about contexts and about itself; these beliefs may be incorrect. An agent's intentions activate a course of action (s) as one's purpose or objective plan.

An agent's desires means a strong feeling of wanting to have something or wishing for something to happen but still it is not in an intention state.

The following is thus a legal modal expression:

believe AGE = 50;

intention (week = 1, heartrate = 80, … week = 12, heartrate = 75);

Context

Context is a structure composed by entities and relationships. Such data can be of two types: unchangeable and changeable.

An agent is part of a context and can represent and analyze it as a dynamical system. In general, the state of a dynamical system is a trace that summarizes all the information about the past behavior of the system.

Beliefs related to a context overload agent belief, for example:

On (table, floor); unchangeable

John (age, 15); changeable

d) Communication

A DA provides built-in communication primitives for sending and receiving messages. Their messages can be: pushed to mobile. They send orders and data, receive orders and data. Messages are event following SSB standard.

The effect of communication is to change the state of the recipient of the message or to activate actions. Note that message delivery is guaranteed, but it is asynchronous.

Functions

A DA can activate functions as precompiled or interpreted SSB routine. All functions have global scope and can be invoked either by agents or by other functions.

Learning

A DA can learn from different sources: Smart Objects, Mobiles and internet. Data is appropriately used to extract meaning using neural networks or mapping data to a cognitive map. Neural network are the key tool to classify events a.

Neural networks can be applied to numerous situations where time series prediction is required. We can turn a temporal problem into a simple input output mapping by taking the time series data x(t) at k time-slices t, t–1, t–2, …, t–k+1 as the inputs, and the output is the prediction for x(t+1).

Agent's Cognitive Architecture

Central to AI Machine is an extended semantic network. This is a knowledge base that contains a large number of entities of different type (class) and relationships between entities.

Each knowledge base is related to a context, an aggregate of entities, relationships and rules.

AI machine can be thought of as an architecture where there is no single algorithm that is responsible for intelligence. Rather, a large number of different specialized algorithms can be actives and these works closely together in cognitive synergy.

An Agent is immersed in a context (and he is swimming in the IoE interacting with Agents, processes, people). This implies that its learning and adaptation activities presuppose social learning.

In an agent it is very important to attribute importance to the whole of the cognitive and social processes involved in the learning process, integrating them with the associations between stimuli and reinforcements that follow a certain behavior, as behaviorism had always done.

An agent is able to put into play his private processes, such as attention or thought, to learn.

There are three elements that interact with one another in reference to the learning process: the agent, the context and behavior. It is the so-called mutual determinism or triadic reciprocity, whereby the environment influences the subject and his behavior, the subject influences the environment with his behavior and behavior influences the subject himself. We learn by observing the external context and the environment that surrounds us. We learn not only through reinforcements and penalties because mere observation produces certain learning effects without the need for direct reinforcement.

Computational Modelling for Digital Agent

After the description of the basic characteristic of a Digital Agent, we will describe the internal structure of their architecture.

First of all each DA has multiple cognitive/perceptual layers with increasing degrees of abstractions.

We require that there be top-down and bottom-up mechanisms working together to connect the Cognitive representations to the perceptual data.

We require a Cognitive Layer (CL) and a Perceptual Layer (PL) with their own autonomous structures. The structure of CL reflects the Cognitive associations that we normally acquire through experience; and the structure of the perceptual layer reflects the historical series of input data. There is an autonomous memory for each layer where the associations or structures of the respective layers are stored.

Phenomenological Layer

Phenomenological layer groups IoT data. The objective of PL is to organize data and to map to CL describing situations. During every system cycle new data are collected in PL.

Interface with the context

A sentient has a Receptive field as a particular region of the sensory space in which a stimulus (event) will modify the firing of the input neuron. In our model the receptive field is a set of sensorial point of a context.

Receptive fields' points to interpretative object called place cell. A place cell is a kind of a neuron in a SOM (Self-Organizing Map) called "place field". Place cells are thought, collectively, to act as a cognitive representation of a specific location in space, known as a cognitive map.

Multisensory integration

Multisensory integration permits to manage information from the different sensory modalities may be integrated by the Phenomenological system.

A coherent representation of objects combining modalities enables us to have meaningful perceptual experiences. Indeed, multisensory integration is central to adaptive behavior because it allows to perceive coherent perceptual entities in the context.

Bayesian integration

The theory of Bayesian integration is based on the fact that the agent must deal with a number of inputs, which vary in reliability. In dealing with these inputs, it must construct a coherent representation of the world that corresponds to reality. The Bayesian integration view is that the agent uses a form of Bayesian inference [16]. This view has been backed up by computational modeling of such a Bayesian inference from signals to coherent representation, which shows similar characteristics to integration in the brain.

With the assumption of independence between various sources, traditional cue combination model is successful in modality integration. However, depending on the discrepancies between modalities, there might be different forms of stimuli fusion: integration, partial integration, and segregation. To fully understand the other two types, we have to use causal inference model without the assumption as cue combination model. This freedom gives us general combination of any numbers of signals and modalities by using Bayes' rule to make causal inference of sensory signals [16].

Development of multisensory operations

An agent equipped with multiple sensory systems, utilize them in an integrative manner to achieve action and perception [17].

The agent's experiences in the context

We assume that the continuous interaction of an agent and its environment is summarized by a discrete view-action-view sequence of the form:

$v_0, a_0, v_1, a_1,...,a_n, v_n$. (1)

A view represents a sensory description associated with a context state.

An action denotes a sequence of one or more control laws (Zhang Y.) that take the agent from one state to the next. Distinctive states are the result of following control trajectory. The basin of attraction of the hill-climbing control laws absorbs accumulated error from each trajectory every time each action happens.

The sequence (1) is transformed into a set of schemas of the form (vi, si), ai, (vi+1, si+1), where si is the state name associated with the environment state where view vi is observed. A schema represents a particular action execution of the agent in the environment. An action execution is characterized in terms of the distinctive states the agent was at before and after the action was performed.

If the states of the system to control obey to a SISO nonlinear system:

$ds(t)/dt = f(s(t)) + Bu(t), y(t) = Cx(t)$ (2)

where s(t) is the state vector, y(t) is the system output e u(t) is the system input, we can achieve the desired control with a neural network. Using a recurrent neural network, we can construct a neural network system model:

$$ds(t)/dt = As(t) + WS(s(t)) + Bu(t),\ y(t) = Cs(t) \quad (3)$$

where W is the connection matrix and S is the activation function.

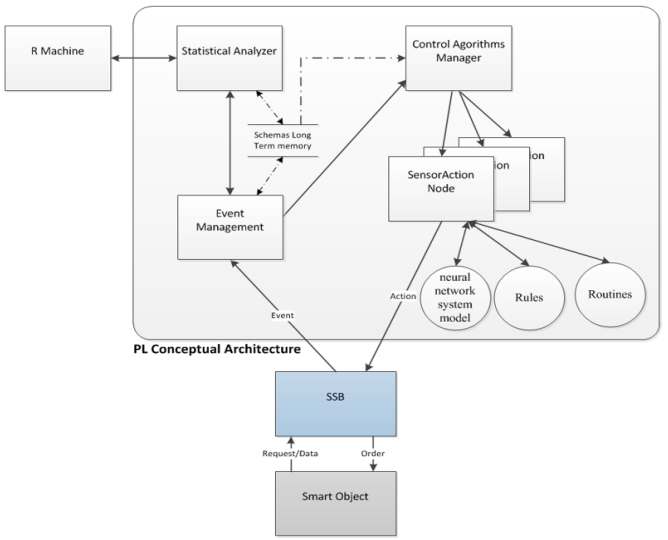

Figure 2: Phenomenological Layer.

Cognitive Layer

Phenomenological states are subject to a transformation from Phenomenological Entities into Cognitive Entities and Relationships. The CL has as main components a cognitive map, which serves to acquire, code, store and recall about the relative locations and attributes of phenomena in their everyday knowledge environment. The term refers to a kind of neural network representing the agent knowledge. A Cognitive map is a map, indicating routes and paths and environmental relationships, which finally determines what responses the system will release.

Knowledge is updated whenever a relationship is found. The cognitive map updated by a neural network is a topological map, i.e., it represents only the connectivity between entities. Associations between entities can be represented by the variable Vi, j, an association stored in the modifiable synapses of the neural network. Whereas a positive Vi, j association means that knowledge entity j can be accessed from knowledge i. 0 means no relationship exists.

The prediction of neighboring place j, pj, is permits to calculate activation aj by aj = pi Vi,j, and this activity indicates whether entity j is accessible from entity i.

Cognitive Layer Basic Flow

During every system cycle new data is collected and transformed into Phenomenological Entities, if appropriate. Data are contained in view and schemas structured depending on a template.

If we indicate EAVi = [elementi, attributei, valuei] and RELk = (elementi, relation_namek, elementj) as elements of the views; we can execute rules:

{EAVi} AND {RELi} => {actionj} AND {EAVj} AND {RELj}

and new states prepared using statistical tools of R machine, if appropriate.

There is a time cycle to transform Phenomenological Entities into Cognitive Entities and Relationships. Neural networks recognize data status and define a possible update/evolution of CL.

The relation between schemas and contextual cognitive maps is recognized by a type of nonlinear mapping. The neural network model can be applied to many types of non-linear maps, if the pertinent variables were adjusted properly. This is a non-linear mapping between views whose data updates the specific contextual cognitive map while schemas are the input of cognitive maps to generate an output that will operate on PL.

If the CL is very dissimilar or juxtaposed after a predefined number of cycles (dissimilarity is measured on using a standard algorithm) this means a new concept is in the CL and problem solving results are evaluated using a fitness algorithm: more a logic element is able to fix problems, more it is fit. If it is not fit, new CL is changed with the old one.

Method

This research uses a software development methodology specifically suited to the AI and IoT paradigm. The software production process can be successful if an appropriate methodology interacts with all the different layers of the organization: processes, human and non-human resources management. AI computing promotes designing and developing applications in terms of autonomous software entities (agents). Agents are situated in contexts and they achieve their goals by interacting with others agents in terms of high-level protocols.

Learning is the most natural of activities [18]. A machine learns from data it receives by identifying patterns and relationships

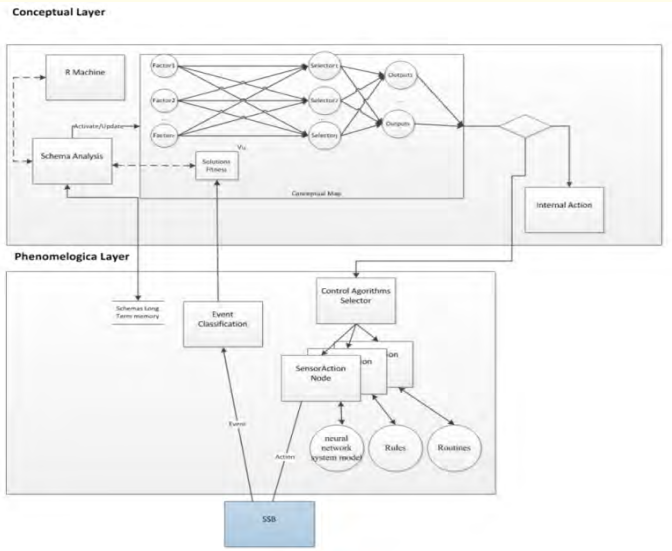

Figure 3: CL Layer.

within the data itself. Once operational, it becomes an automated process with minimal intervention (though substantial influence) from its human counterparts.

Machines can continuously analyze substantial amounts of information and transform what they have read into intelligent insights at a tremendous operational scale.

Overview of methodology

The methodology organizes all the activities in four typical macro-phases: (Plan, Design, Build, and Operate), classifying them into Process, Organization, Resources, depending on how many different layers of the organization exist. The phases of are repeated recursively for each agent that autonomously aligns itself with the organization's goals in a never-ending process. We will focus our attention only on Design Phase.

Design

In the Design phase, you go to the macro-level definition tasks of the process model. In this way it is possible to carry out the gap analysis, which determines the guidelines for subsequent activities. The decomposition of processes in sub-processes and tasks is carried out to the level of detail determined by the specific design requirements. In any case, decomposition levels are defined as process policies, goals and interfaces, leaving the agent creating his experience and inducing adaptive activities. If it is applicable, it can be based on standard pattern logic where process patterns are the set of activities, actions, work tasks or work products and similar related behavior followed by a software development life cycle:

- Develop a high-level business description,
- Develop a Solution Overview Diagram,
- Identify Process Patterns,
- Identify Integration Patterns,
- Identify Composite Patterns,
- Identify Application Patterns,
- Identify Run-Time Patterns.

Identify run-time and product mappings

In each step the use of Machine Learning requires considerable data and computational power. Because Machine Learning applies analytics to such large amounts of data, and runs sophisticated algorithms, it typically requires high levels of computer performance and advanced data management capabilities.

In each Pattern the designer has to value the applicability of Machine Learning algorithms in the terms of:

- Function Approximation
- Probability Estimation
- Pattern Recognition
- Clustering
- Prediction

For each step the designer must define Data Collection and Preprocessing, the Architecture of networks, the appropriate training and validation procedure.

Architecture of networks use the paradigm "maps of maps" to integrate different Receptive Field processing and Machine Learning algorithms. Pattern based Adaptive Modules manage Relationship Module to integrate Sensors Data and networks.

Some Basic Principles of Architectural and IoT Design

We introduce some empirical rules that can lead to the emergence of super-additive phenomena:

1. Information and intelligence must be distributed among a large number of entities like individuals, AI agents and smart objects;

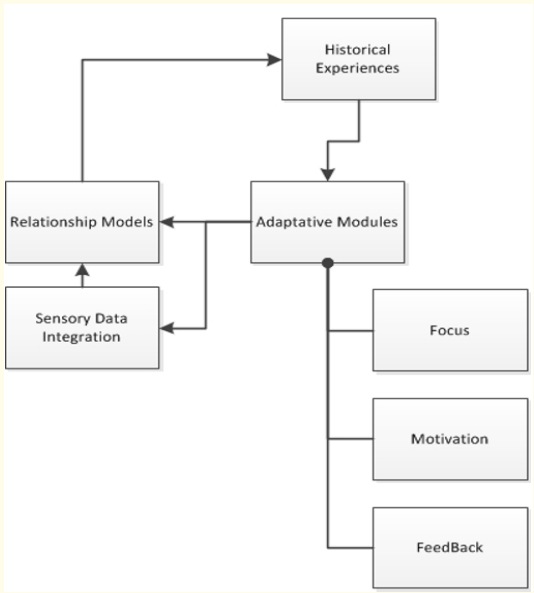

Figure 4: Adaptive and Relationship Modules.

2. Control should not be superimposed, but must emerge as policies and result in interaction between peer-to-peer agents;
3. The system must be autocatalytic; must, therefore, be characterized by a positive feedback, which guarantees increasing returns;
4. The only way to create a truly complex system that works correctly is to assemble it incrementally starting from simple modules that can operate independently;
5. The system must be as heterogeneous as possible as diversity accelerates adaptation and strengthens the system;
6. It is necessary to minimize the errors made during the system testing phases;
7. There is no need to pursue the optimization of a particular function; you need to have multiple goals;
8. In the system testing phase, a state of continuous imbalance must be pursued because equilibrium means death of the whole system.
9. The laws governing large complex systems come from the bottom. Such laws are subject to a process of continuous adaptation, as a result of the interactions between the agents at the lowest levels of such systems.

A system built following previous rules doesn't behave like mechanical systems as can adapt to a vast set of stimuli (even not predetermined). It may evolve and its performance is not sensitive to component faults as a result of redundancy of system elements.

On the other hand, such system is not optimal relying on an emerging control mechanism (such as a price system in a free market economy) and inefficient, due to redundancy in the elements used. It is difficult to control. The system may evolve towards unwanted situations, and is non-linear and this means that, in the face of similar but not equal stimuli, different responses can be obtained. Consequently, it is difficult to predict the behavior of the system.

The complexity of the system, and the high number of interrelationships between the agents, makes the operation of the system at least obscure. In view of these considerations it is necessary to keep the system in close control.

Metrics

Effective management of any process requires quantification, measurement, and modeling. Software metrics provide a quantitative basis for the development and validation of models of the software development process. Metrics can be used to improve software productivity and quality. This module introduces the most commonly used software metrics in AI Architecture.

Run-time Architecture Evaluation

Following [19] a symbol system supports the acquisition, representation, storage, and manipulation of symbolic structures. Architecture is analogous to the hardware of a standard computer, while the symbols (which encode knowledge) correspond to software. The role of a general symbolic architecture is to support the encoding and use of diverse types of knowledge that are applicable to various goals and actions.

The basic functions performed by anarchitecture usually consist of the following [20]:

- The fetch-execute cycle
- Assemble the operator and operands
- Apply the operator to the operands using architectural primitives
- Store results for later use
- Support access structures
- Input and output

Viewpoint	Architectural Metric (AM)	Type (Unit of Measurement)
Scenario Viewpoint	Number of patterns	Counter
Logical Viewpoint	Number of external interfaces: number of Receptive field in the context map, number of interface invocations with other layer and agents. Number of nets and adaptive module per net	Counter Counter
Process Viewpoint	Process Counter Process Coordination: Interprocess Communication (IPC), Remote Call Counter.	Counter Counters
Physical Viewpoint	Tier Counter	Counter
Architectural Decision Viewpoint	Number of architecture design problems solved Number of options considered per problem	Counter Fuzzy Counter
Information Viewpoint	Data model size and structure (e.g., number of entities and entity relationships) Transaction management profile, e.g. number of system transactions and their size/duration	Counter Counter
Patterns Metrics	number of layers EIP integration flows	Counter Counter

Table a

Viewpoint	Architectural Metric (AM)	PL	CL
Scenario Viewpoint	Number of pattern	18	10
Logical Viewpoint	Number of external interfaces (number of Receptive field in the context map) and number of interface invocations with other layer and agents Number of nets and adaptive module per net	5	2
Process Viewpoint	Process Counter Process Coordination Means Interprocess Communication (IPC) and Remote Call Counter	6 65	6 2
Physical Viewpoint	Tier Counter	2	2
Architectural Decision Viewpoint	Number of architecture design problems solved Number of options considered per problem	5 NA	2
Information Viewpoint	Data model size and structure (e.g., number of entities and entity relationships) Transaction management profile, e.g. number of system transactions and their size/duration	7	7
Patterns Metrics	Number of patterns, EIP integration flows	18 3	

Table b

Activity	Unit duration (secs)	#	Ave Duration (sec)
Execute cycle			14
Apply Operator	0,24	20	4,8
Rules	0,32	24	7,68
I/O			1,52

Table c

Results

Architectures are distinguished by their implementation and specific set of primitive operations supported.

We experimented the described architecture with the following metrics.

We processed weather forecasts for 65 simulated smart objects with 65 DA executions on a ACPIx64 server using as database a MySql engine. We have the following average results.

A Digital Agent uses in average 14 seconds to terminate the execution. Operators containing neural networks activations and R routines use less time than the logical algorithm of rules. We have to balance rules that are easier to implement but slower in execution with the other operand that uses neural network.

Discussion

This research has explored the use of maps onto maps on two layers. We trained supervised neural network because our assumption was that this is the basic structure of a brain as experimented by neuroscience.

Metrics where used to value specific rules and operators performance. Next steps will be to create higher layer to permit a DA stream to improve cognitive levels of the systems.

An organizational notation: Knowledge is produced from big data in the form of a variety of trained different neural networks, but a higher competence will be required to people. Such competence development can no longer be considered a marginal activity but must be considered as a contribution to any organization creation. A strong methodology that integrates processes improvement with AI software development will fluid and adaptive knowledge. If knowledge is constantly updated together with technological evolution, this allows better prospects for the future [21-66].

Conclusion

This research has explored the use of maps onto maps on two layers. We trained supervised neural network because our assumption was that in nature the basic structure of a brain are coded in DNA. This is equivalent to create a structure using commands. Next steps will be to create higher layer to permit a DA stream to improve cognitive levels of the systems.

Bibliography

1. Carl H., *et al*. "A Universal Modular Actor Formalism for Artificial Intelligence". *IJCAI* 1973.
2. Kitchenham B. "Metrics and Measurement". Software Engineer's Reference Book, edited by John McDermid, Butterworth.
3. Kitchenham B. "Software Metrics: Measurement for Software Process Improvement". Blackwell Publishers, Inc. Cambridge, MA, USA © (1996).
4. Abran A., *et al*. "Guide to the software engineering body of knowledge". IEEE Computer Society Press (2004).
5. Sun R. "Desiderata for Cognitive Architectures". *Philosophical Psychology* 17 (2004): 341-373.
6. Sun R. "Cognition and Multiagent Interaction, From Cognitive Modeling to Social Simulation". In: Sun, R. (ed.) Rensselaer Polytechnic Institute, Cambridge U. Press, Cambridge (2005).
7. Anderson J., *et al*. "An Integrated Theory of the Mind". *Psychological Review* 111 (2004): 1036-1060.
8. Anderson JR and Lebiere C. "The Atomic Components of Thought, Lawrence Erlbaum Associates"
9. Laird JE. "Research in human-level AI using computer games". *Communications of the ACM* 45 (2002): 32-35.
10. Laird JE. "Preface for special section on integrated cognitive architectures". *SIGART Bulletin* 2 (1991): 12-123.
11. Laird JE. "Using a computer game to develop advanced AI". *Computer* 34 (2001): 70-75.
12. Laird JE., *et al*. "Chunking in Soar: The anatomy of a general learning mechanism". *Machine Learning* 1 (1986): 11-46.
13. Laird JE., *et al*. "Soar: An architecture for general intelligence". *Artificial Intelligence* 33 (1987): 1-64.
14. Miikkulainen R and Dyer M. "Natural Language Processing with Modular PDP Networks and Distributed Lexicon". *Cognitive Science* 15 (1991): 343-399.
15. Kotseruba I and J Tsotsos. "40 Years of Cognitive Architectures Core Cognitive Abilities and Practical Applications".
16. Vilares and Kording K. "Bayesian models: the structure of the world, uncertainty, behavior, and the brain". *Annals of the New York Academy of Sciences* 1224 (2011): 22-39.
17. Stein B., *et al*. "The merging of the senses". Cambridge, Mass: MIT Press (1993).
18. Garvin DA. "Learning in Action: A Guide to Putting the Learning Organization to Work, Harvard (2000)". Business School Press, Boston, MA.
19. Cabri G. ''Engineering mobile agent applications via context dependent coordination". *IEEE* 28 (2002): 1034-1051.
20. Newell A. "Unified Theories of Cognition". Harvard Press, Boston, MA. Shapiro, D., and Langley, P. (1990).
21. Gat E., "Integrating planning and reacting in a heterogeneous asynchronous architecture for mobile robots". In Proceedings Tenth National Conference on Artificial Intelligence (1992): 809-815.
22. Laird JE., *et al*. "A Standard Model for the Mind: Toward a Common Computational Framework across" (2017).
23. Langley PJ and Laird E. "Cognitive Architectures: Research Issues and Challenges, (Technical Report)". Institute for the Study of Learning and Expertise, Palo Alto, CA (2017).
24. Lee J., *et al*. "Reactive-system approaches to agent architectures". in N.R. Jennings and Y. Lesp, Intelligent Agents VI, Proceedings of the Sixth International Workshop on Agent Theories, Architectures, and Languages ATAL-99.
25. Lehman J., *et al*. "A Gentle Introduction to SOAR, An Architecture for Human Cognition: 2006 Update". University of Michigan (2006).
26. Lespérance NR. "Intelligent Agents VI. Agent Theories, Architectures, and Languages. ATAL 1999". *Lecture Notes in Computer Science* (1999): 1757.
27. Newell A. "Unified Theories of Cognition". Harvard University Press, Cambridge, MA (1990): 11.

28. Milner R. "Processes: A Mathematical Model of Computing Agents in Logic Colloquium 1973 Artificial Intelligence, Cognitive Science, Neuroscience, and Robotics, AI Magazine 38 (4).

29. Pollack ME and M Ringuette. "Introducing the tile world: Experimentally evaluating agent architectures". In Proceedings of the Eighth National Conference on Artificial Intelligence 1 (1990): 183-189.

30. Tabuada P. "Symbolic Control of Linear Systems Based on-Symbolic Subsystems". IEEE Trans. On Automatic Control 51.6 (2006): 1003.

31. Adaptive Neural Network Based Control of Noncanonical Nonlinear Systems. IEEE Transactions on Neural Nnetwork.

32. Wermter S and Sun R. "Hybrid Neural Symbolic Integration, Berlin, Springer Verlag" (2000).

33. Wallace SA and Laird JE. "Toward a Methodology for AI Architecture Evaluation: Comparing Soar and CLIPS".

34. Kluwer Albus J. *et al*. "RCS: A reference model for intelligent control". *IEEE Computer* 25 (1992): 56-79.

35. Ali SS and Shapiro SC. "Natural language processing using a propositional semantic network with structured variables". *Minds and Machines* 3 (1993): 421-451.

36. Anderson JR. "Cognitive architectures in a rational analysis". In K. VanLehn (Ed.), Architectures for intelligence (1991).

37. Mahwah NJ: "Lawrence Erlbaum". Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. An integrated theory of the mind. Psychological Review 111 (2004a):1036-1060.

38. Gat E., *et al*. "Experiences with an architecture for intelligent, reactive agents". *Journal of Experimental and Theoretical Artificial Intelligence* 9 (1997): 237-256.

39. Bratman ME. "Intentions, plans, and practical reason". Cambridge, MA: Harvard University Press (1987).

40. Carbonell JG., *et al*. "Prodigy: An integrated architecture for planning and learning". In K. Van Lehn (Ed.), Architectures for intelligence. Hillsdale, NJ (1990).

41. Haigh K and Veloso M. "Interleaving planning and robot execution for asynchronous user requests". Proceedings of the International Conference on Intelligent Robots and Systems Osaka, Japan: IEEE Press (1996): 148-155.

42. Hayes-Roth B., *et al*. "A domain specific software architecture for adaptive intelligent systems". *IEEE Transactions on Software Engineering* 21 (1995): 288-301.

43. Hendler J., *et al*. "AI planning: Systems and techniques". *AI Magazine* 11 (1990): 61-77.

44. Ingrand FF., *et al*. "An architecture for real-time reasoning and system control". *IEEE Expert* 7 (1992): 34-44.

45. Jones RM., *et al*. "Automated Intelligent Pilots for Combat Flight Simulation". *AI Magazine* 20 (1999): 27-42.

46. Konolige, K., *et al*. "The Safira architecture: A design for autonomy". *Journal of Experimental and Theoretical Artificial Intelligence* 9 (1997): 215-235.

47. Langley P. "Order effects in incremental learning". In P. Reimann and H. Spada (Eds.), Learning in humans and machines: Towards and interdisciplinary learning science. Oxford: Elsevier (1995).

48. Langley P. "Intelligent behavior in humans and machines (Technical Report)". Computational Learning Laboratory, CSLI, Stanford University, CA (2006).

49. Langley P and Choi D. "A unified cognitive architecture for physical agents". Proceedings of the Twenty-First National Conference on Artificial Intelligence. Boston: AAAI Press (2006a).

50. Langley P and Choi D. "Learning recursive control programs from problem solving". *Journal of Machine Learning Research* 7 (2006b): 493-518.

51. Langley P., *et al*. "Hierarchical skills and cognitive architectures". Proceedings of the Twenty-Sixth Annual Conference of the Cognitive Science Society (2004): 779-784.

52. Chicago IL., *et al*. "Experimental studies of integrated cognitive systems". Proceedings of the Performance Metrics for Intelligent Systems Workshop (2004).

53. Gaithersburg MD and Lewis RL. "An architecturally-based theory of sentence comprehension". Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society (1993): 108-113.

54. Boulder CO., *et al*. "Situations, actions and causal laws (Memo 2)". Artificial Intelligence Project, Stanford University, Stanford, CA (1963).

55. Meyer M and Kieras D. "A computational theory of executive control processes and human multiple-task performance: Part 1. Basic mechanisms". *Psychological Review* 104 (1997): 3-65.

56. Muscettola N., *et al*. "Remote Agent: To boldly go where no AI system has gone before". *Artificial Intelligence* 103 (1998): 5-48.

57. Nason S and Laird JE. "Soar-RL: Integrating reinforcement learning with Soar". Proceedings of the Sixth International Conference on Cognitive Modeling (2004): 220-225.

58. Newell, A. "Unified theories of cognition". Cambridge, MA: Harvard University Press (1990).

59. Nuxoll A and Laird J. "A cognitive model of episodic memory integrated with a general cognitive architecture". International Conference on Cognitive Modeling (2004): 220-225.

60. Pell B., *et al*. "An autonomous spacecraft agent prototype". Proceedings of the First International Conference on Autonomous Agents Marina del Rey, CA: ACM Press (1997): 253-261.

61. Remington R., *et al*. "Using Apex/CPM-GOMS to develop human-like software agents". Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (2003).

62. Sammut C. "Automatic construction of reactive control systems using symbolic machine learning". Knowledge Engineering Review 11 (1996): 27-42.

63. Shapiro D and Langley P. "Symposium on learning and motivation in cognitive architectures: Final report". Institute for the Study of Learning and Expertise, Palo Alto, CA (2004).

64. Sun R. "Cognition and multi-agent interaction: Extending cognitive modelling to social simulation". Cambridge University Press (2005).

65. Trafton JG., *et al*. "Enabling effective human-robot interaction using perspective-taking in robots". IEEE Transactions on Systems, Man and Cybernetics, 25 (2005): 460-470.

66. VanLehn K. "Architectures for intelligence". Hillsdale, NJ: Lawrence Erlbaum. Veloso, M. (1991).

Volume 3 Issue 3 March 2019