Usage-Based Learning in Human Interaction With an Adaptive Virtual Assistant

Clément Delgrange, Jean-Michel Dussoux, and Peter Ford Dominey

***Abstract*—Today users can interact with popular virtual assistants such as Siri to accomplish their tasks on a digital environment. In these systems, links between natural language requests and their concrete realizations are specified at the conception phase. A more adaptive approach would be to allow the user to provide natural language instructions or demonstrations when a task is unknown by the assistant. An adaptive solution should allow the virtual assistant to operate a much larger digital environment composed of multiple application domains and providers and better match user needs. We have previously developed robotic systems, inspired by human language developmental studies, that provide such a usage-based adaptive capacity. Here, we extend this approach to human interaction with a virtual assistant that can first learn the mapping between verbal commands and basic action semantics of a specific domain. Then, it can learn higher level mapping by combining previously learned procedural knowledge in interaction with the user. The flexibility of the system is demonstrated as the virtual assistant can learn actions in new domains (e-mail, Wikipedia, etc.), and then can learn how e-mail and Wikipedia basic procedures can be combined to form hybrid procedural knowledge.**

***Index Terms*—Human-system interfaces, intelligent assistant, language, usage-based learning.**

I. Introduction

NATURAL language is a concise and effective way to interact with software and cloud services. Natural-language interfaces have demonstrated their usefulness in areas, such as information extraction from databases [1], robot control [2], or in the execution of daily tasks through companions like Google Now, Siri, or Cortana [3]. A limitation of these systems is the inability to benefit from user's knowledge when dealing with unexpected situations. Such situations could be a speech input requesting an action unknown to the system, an unknown lexical unit or the inability to generate a valid course of actions. To overcome such limitations, a system should have the opportunity to learn from its own experience and from external sources by the means of natural language instructions and perceptual demonstrations. We determined that this objective can be achieved by taking advantage of concepts from usage-based learning in human language acquisition.

Our objective is to provide a method that allows the user maximum flexibility in teaching new behaviors to the assistant using natural language by minimizing the need of specific knowledge engineering of task domains or languages.

We first review related works in the domain of virtual assistants that exhibit the need to design systems that are adaptive to the evolution of the functionalities of a digital environment, in term of application domain and tasks, and to the changing user needs.

We then present robotic works inspired by developmental studies on which this paper is grounded. Here, we do not attempt to duplicate usage-based learning and generalization of grammatical constructions as in the human, but rather, we exploit three particular aspects of usage-based learning of grammatical constructions, and then we apply this to an intelligent assistant. First, we adopt the concept that language is a collection of form-to-meaning mappings [4] where in our case form is the natural language command from the user and meaning is the demonstrated execution. Second, we adopt the notion of abstraction over arguments, where a given construction can be reused over a set of different arguments. Finally, perhaps most importantly, we adopt the generative capacity for compositionality, where new constructions can be composed of existing ones, inheriting their argument structure.

In order to evaluate our proof-of-concept system, we provide a set of use cases transcriptions and an analysis of the number of interactions needed to complete a set of tasks.

We finally discuss how state-of-the-art machine learning techniques could improve the generalization of natural language user utterances to compete with actual virtual assistants and then lead to a realistic user study.

A. Domain Adaptive Assistant

User tasks generally require the use of multiple applications, for example, to plan an evening out with friends that require to interact with a messaging service, a calendar, and a search engine. One task the user would want to achieve is to

Manuscript received September 25, 2018; revised January 29, 2019 and May 24, 2019; accepted July 4, 2019. Date of publication July 9, 2019; date of current version March 11, 2020. This work was supported by the Jarvis and Intrinsec/Cloud-Temple under Grant ANRT/CIFRE 151575A10. *(Corresponding author: Peter Ford Dominey.)*

C. Delgrange is with INSERM UMR1093-CAPS, Université Bourgogne Franche-Comté, UFR des Sciences du Sport, 21000 Dijon, France, also with INSERM, CNRS, Robot Cognition Laboratory, Institut Marey, Université Bourgogne Franche-Comté, 21000 Dijon, France, and also with the Department of Innovation, Cloud Temple, 92000 Nanterre, France.

J.-M. Dussoux is with the Department of Innovation, Cloud Temple, 92000 Nanterre, France.

P. F. Dominey is with INSERM UMR1093-CAPS, Université Bourgogne Franche-Comté, UFR des Sciences du Sport, 21000 Dijon, France, and also with INSERM, CNRS, Robot Cognition Laboratory, Institut Marey, Université Bourgogne Franche-Comté, 21000 Dijon, France (e-mail: peter.dominey@inserm.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2019.2927399

send the list of the nearest pubs to his friend and the virtual assistant should so be able to interact with multiple application providers and of different domains.

Sun *et al.* [5] investigated how a future agent could organize activity across multiple applications (e.g., Yelp, OpenTable, and Messenger) in order to achieve a high level goal like "help me plan an evening out with my friends." Here, the association between user intentions and the use of specific applications are learned thanks to user traces on a smartphone. The value of such an agent is that it can allow the user to build their own virtual assistant based on their favorite applications without the need for each provider to share a common semantic. The long term goal of such agents is to operate a larger digital environment composed of a larger number of possible tasks.

B. User-Task Adaptive Assistant

Another aspect of adaptation is to allow the user to define the interpretation of a natural language order. This can be done by building instructable agents where a user can teach to the assistant how to accomplish a task by demonstrations or instructions at runtime and relies on one shot learning techniques.

1) PLOW: The PLOW system, developed by Allen *et al.* [6], allows the user to teach new tasks to an assistant based on GUIs demonstration. The range of problems covered by PLOW is broader than procedural knowledge acquisition and includes a set of modules allowing for example to emulate deliberative dialogues. Language understanding is performed by converting user natural language sentences into logical forms [7] and relies on the TRIPS system for dialogue management and task planning [8].

While the PLOW system can learn new task with the user, a number of domain specific competencies must be engineered for the system to work. This makes the system difficult to maintain as the number of applications increases.

2) LIA: LIA is another example of an instructable agent that users can teach to perform new action sequences to achieve new commands using solely natural language [9]. LIA is composed of two parts: 1) a semantic parser which assigns executable semantics to each learned natural language command and 2) a back-end which executes these commands. The back-end contains a set of primitive functions for a particular domain along with their corresponding logical form. LIA use a combinatory categorial grammar (CCG) to parse natural language utterances. The parser must first be trained to associate natural language lexicon with the set of primitive functions. The system can then derive more complex logical forms from user inputs.

While the LIA system has a lighter infrastructure than PLOW, it requires the use of predefined high-level primitives, like *sendEmail*. This prevents the user from using the same kinds of one shot learning technique to accommodate the virtual assistant to new domains or new applications (extending the back-end in LIA).

II. Usage-Based Language Understanding

Here, we consider how these limitations can be addressed by taking inspiration from usage-based learning.

Usage-based learning is a social-pragmatic approach to language acquisition in which children learn linguistic structures through understanding intentions and finding patterns in their discourse interactions with others [10]. From the outset of life, children and adults interact in feeding, changing diapers, getting ready for bed, etc., in repeating rituals that are accompanied by language from the adults [11]. This provides a rich shared space where language can enrich meaning at multiple levels [12]. In the usage-based concept of language acquisition, the child's first words allow reference to people and objects that are of central prominence in everyday life [10], [11]. After words, the first grammatical constructions are fixed around specific verbs, and specific actions that tend to be repeated and ritualized in the infants' social environment [10], [13], [14]. Constructions then become more abstract and generalized as the child's world becomes enriched [4], [10].

This learning requires access to perceptual and motor meaning via mechanisms that have been characterized in the context of developmental robotics [15]. The essential idea is that primitive innate perceptual and motor capabilities can be composed into more complex percepts or behaviors, and that these structured representations become associated with language [16]. Lallée *et al.* [17]–[21] have previously applied this methodology of structuring perceptual-motor primitives around language in the context of human–robot cooperative interaction. These works concentrate on three major aspects.

1) The existence of symbolic primitives grounded in the perceptual system.
2) Grounding the language in these primitives.
3) Learning the use of natural language by engaging the robot into cooperative activities with a human.

Our objective is to extend this paper into the domain of cooperative interaction with a virtual assistant. Part of the motivation is to allow the assistant to acquire human expertise through cooperative interaction with humans.

A. Symbolic Primitives

1) Developmental Theory: In the context of language development in human, a fundamental question is how the symbolic structure of language is grounded into the perceptual representation system.

Mandler established the existence of symbolic primitive structures [22], called schema. Schemas are described as a simplification of the information contained in the perceptual system. They are learned early by the infant through a process, called perceptual meaning analysis (PMA), that redescribes perceptual representation with a set of early developed structures, such as PATH, MOTION, or CONTAINMENT [23]. Inputs to the PMA are temporal aspects of objects moving along paths, "into or out of containers, objects going into or out of sight, and contingencies between paths, such as one object chasing another" [24]. The PMA generates representations that are the interpretation or construal of the events that are being observed. These image schemas can then be further enriched by language, which can add additional structure that the PMA cannot supply [24].

The use of perceptual primitives, such as MOTION and CONTACT do not assume any form of innate predicate structure, but instead, they are the base which allow a symbolic grounding of language or images.

2) Application in Robotic: Robots, by their physical attributes, share the same space as humans and one task is to convert information extracted from a camera into a meaning structure linked to natural language.

Perceptual robotics is a research domain in which robots are equipped with perceptual systems that can produce internal representations of the physical world, sensory systems provide a recoding or representation of the world that can provide the meaning component for this grounding. Thus, in Dominey and Boucher [25], a computer vision system recognized colored objects and provided outputs that could be used to ground the meaning of words in sentences. That is, the words could be learned to refer to the outputs produced by the visual system. Interestingly, this grounding applied to nouns and verbs. Nouns, like "block," "moon," and "cylinder" were grounded in the objects identified by the visual system. The visual system also extracted dynamic properties of these objects, including the direction and speed of motion (related to Mandler's MOTION), and whether objects were in contact (related to Mandler's CONTACT). Interestingly Dominey and Boucher determined that actions like "push" and "give" could be defined in terms of sequences of perceptual primitives related to contact. We, thus, developed an event parser which could detect GIVE(*A, B ,C*), meaning "A gives B to C" as the following sequence of primitive events: CONTACT(*A, B*), MOVE(*A*), MOVE(*B*), CONTACT(*B, C*), MOVE(*A*), and END-CONTACT(*A, B*).

In the same manner, Lallée *et al.* [26] developed a system that learned to recognize actions, including COVER, UNCOVER, GIVE, and TAKE, based on the perceptual primitives VISIBLE, MOVE and CONTACT. These were encoded as <*enabling state, action, resulting state*> triplets. This research thus provides a demonstration of how basic meaning representation and higher level actions can be extracted from the perceptual stream, based on a initial set of simple perceptual primitives.

B. Grammatical Construction: Grounding Language in Symbolic Primitives

1) Developmental Theory: In a constructionist approach, very limited initial perceptual and motor abilities form the primitives that are assembled and structured as part of the process of usage-based language learning. In this context, language is considered to consist of a structured inventory of mappings between the surface forms of utterances and meanings, referred to as grammatical constructions [4]. These mappings vary along a continuum of complexity. At one end are single words and fixed "holophrases" such as "Gimme that" that are processed as unparsed "holistic" items [10]. At the other extreme are complex abstract argument constructions that allow the use of sentences like this one. In between are the workhorses of everyday language, abstract argument constructions that allow the expression of spatiotemporal events that are basic to human experience, including active transitive (e.g., John took the car) and ditransitive (e.g., Mary gave my mom a new recipe) constructions [4].

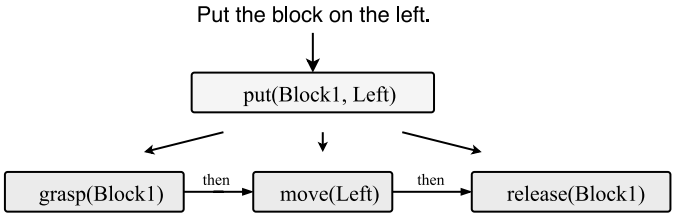

Fig. 1. Grammatical construction grounding: the sentence "put the block on the left" is mapped on a predicate-argument structure, which in turn is grounded on perceptual primitive schemas.

The notion of constructions becoming more abstract is a prevalent notion in language acquisition in the construction grammar framework. Tomasello extensively develops this point [10, Ch. 5]. The essential argument is that basic argument structure constructions are directly associated with dynamic, experientially grounded scene gestalts [27]. Then, through processes of abstraction, and pattern finding, analogy and distributional analysis, constructions become progressively more abstract. Gaspers *et al.* [28] addressed how abstraction can take place in a usage-based computational model of language acquisition, that is, based on bootstrapping mechanisms that relies on discovered regularities to segment speech into word-like units. In the top-down aspects, based on this initial segmentation, the model induces a construction grammar that in turn acts as a top-down prior that guides the segmentation of new sentences into words. Dominey proposed a model that first learned idiomatic holophrases that mapped directly onto meaning representations, then certain arguments became liberated in verb island constructions, and finally all open class elements become liberated in abstract argument constructions [29].

Fig. 1 illustrates the nature of simple constructions as mappings from sentence form to a predicate-argument meaning representation grounded into primitive schemas, they are the form of construction mainly used in robotics. We then present more advanced tools for abstracting language.

2) Application in Robotics: A central issue in language learning is grounding: establishing the link between a word and its referent.

Dominey and Boucher initially studied these form-to-meaning mappings in the domain of perceptual robotics [30] using the perceptual capabilities described above in Section II-A. There, a perceptual system learned to map perceived event representations, such as *push(Moon1, Cylinder1)* onto sentences, such as "The moon pushed the cylinder," or "The cylinder was pushed by the moon." They then extended this approach to include both the description of perceived events, and the execution of spoken commands in the cooperative human–robot interaction system (CHRIS) architecture [19].

For new actions (that have not yet been defined in the knowledge base) the system uses the set of observed primitives from temporal segmentation to generate a generic pattern

of primitives to define the action. Thus, a new predicate *cover*(*x*, *y*) is learned as a pattern of primitives: <*moving*(*x*), *contact*(*x, y*), *!visible*(*y*), *!moving*(*x*)> which is generated by the perceptual system. This then is the pattern onto which "Cover *y* with *x*" can be associated [26]. These actions were also associated with states. Thus, for example, after the action *cover*(*x, y*), the resulting state is *contact*(*x, y*), *visible*(*x*), *on*(*x, y*). By this same mechanism, Lallée *et al.* [26] demonstrated 4 other complex predicate-argument actions, including uncover, take, give, and put. These actions could then be assembled into higher level constructs that we called shared plans, allowing the human and robot together to achieve a desired state. This research thus provides a framework for demonstration of how predicate-argument structure for high level actions like *cover*, *put*, etc., can emerge.

In this context of grounding and language learning, Cangelosi and Riga [31], Stramandinoli *et al.* [32], and Tikhanoff *et al.* [33] have developed a number of simulation and robotics experiments that demonstrate how word meaning can be grounded in perception for language learning. Their more recent work begins to look at grammatical structure in the case of verb–noun pairs that link to object-oriented action [32]. Indeed, it is likely that meaning is more pertinent in the context of goal oriented action schemas or frames [34]. The learning of grammatical constructions has been approached both from a language evolution perspective [35], and from a developmental or learning perspective [25], [30], [36]. An important common point is that they rely on some form of organization of meaning in terms of pragmatic frames, or repetitive goal-oriented action schemas.

3) Models and Tools: Powerful algorithms for grammar induction combine statistical processing for pattern extraction, and rule-based structural generalization based on these discovered patterns [37]. This results in unprecedented capabilities for extracting grammatical structure, without addressing the problem of how such structure maps onto meaning. In this context of form to meaning mappings, Dominey *et al.* [38] initially investigated neural networks that learned to associate the pattern of closed class elements in a sentence with the ordered mapping of open class (semantic) words onto their respective thematic roles in the meaning. Hinaut and Dominey [39] subsequently showed that with larger networks and training corpora, such systems could actually generalize to new constructions that were not used in the training data.

Gaspers and Cimiano [40] exploited cross situational statistics that link words to their referents across multiple scenes, in order to then build up progressively more elaborated grammatical constructions. Gaspers *et al.* [28] then employed a powerful method where bottom up and top down learning mechanisms mutually reinforce each other in learning form to meaning mappings. An initial lexicon is formed bottom up from phonemic sequences that appear frequently enough to establish initial form-meaning mappings. These initial words can then be mapped into perceived sentences in order to define alignments between words in the sentence and their roles in predicate argument representations of meanings, which allow identification of grammatical constructions. Once an initial set of constructions is learned, they can be used in a top down manner to identify the meanings of new words in sentences that correspond to these learned constructions. This allows for a synergistic interaction between bottom-up and top-down learning. Dominey and Boucher [25], [30] and Dominey [41] exploited these two processes in where semantic bootstrapping provides initial word meaning which allows the learning of the mapping between lexical elements in the sentence and their corresponding role in the predicate-argument structure of the meaning. This results in learning of grammatical constructions that can be exploited in top-down syntactic bootstrapping, where novel words are more easily learned by using known constructions that directly map them to their referent in the meaning representation.

In this paper, we exploit a direct mapping between the user's top down spoken language input, which forms a pattern from the bottom-up demonstration to map onto. This is in line with a recent perspective proposed by McCauley and Christiansen [42]. They argue that in addition to distributional information, computational models of language learning should take more account of meaning, in terms of the agents, objects, and actions that the language is referring to.

C. Learning and Using Language in Cooperative Plans

1) Developmental Theory: A crucial aspect in the development of human cooperation is the ability for the child and adult to engage in a triadic interaction where the two agents interact around the third element—the object of their cooperative interaction. By 12–14 months of age, the child engages in these triadic interactions with the adult, and demonstrates that they are motivated to share experience with others, and importantly, to help them toward their goals [43]. Language allows the developing child to align shared plans with the adult, and to create new language-based representations that persist over time. In development, language is thus a tool for interacting minds, which allows the expansion of new spaces for interaction and cooperation [44].

Following Tomasello, we consider cooperation as the case when two individuals establish a common goal, and develop a shared plan to achieve that goal [43]. In such cooperative scenarios, it is typically an adult "leader" that establishes the shared plan as a turn-taking sequence of actions by the two participants in order to achieve the shared goal.

2) Application in Robotics: The robotic CHRIS system demonstrates how to integrate the core features that consist of the ability to extract meaning from perception, to learn how to compose new actions from primitive and learned actions, and to cooperatively learn and execute shared plans or procedures with a human via a spoken language interaction [19]. As described above, extracting meaning from perception is performed using a set of primitives, including MOTION and CONTACT, which allow recognition of actions like TAKE and GIVE in the perception module of the system. Then, in the grounding of a sentence like "put the block1 on the block2" in meaning, there is the transition from the sentence to a schematized representation of the meaning *put(block1, block2)* in the learned grammatical construction. This representation of the put action is then decomposed into primitive motor commands

like GRASP, MOVE, RELEASE and into perceptual primitives like OBJECT1, OBJECT2 that are encoded in the knowledge base. The language processing and plan learning capabilities are provided by a planning and supervision module and allow a Human, either by natural language descriptions or by physical demonstrations, to specify at runtime how to link a linguistic unit such as "block1" to a perceptual element like OBJECT1 as well as the composition of *put(Object1, Location1)* into the sequence GRASP*(Object1)*, MOVE*(Location1)*, RELEASE.

Lallée *et al.* [19] and then Petit *et al.* [2] have used this mapping of sentences to robot action in order to allow the human user to specify shared plans in cooperative human–robot interaction. One of their experiment was the execution of a shared plan to put a toy into a box. A human specified the shared plan as "I reach the toy, then I reach the trash box then you open the trash box then I put the toy in the trash box then you close the trash box." Then, the shared plan is executed and each successive step is spread between the human and the robot. In that work, the user is also able to teach the robot how to do new actions, like opening and closing the trash box.

III. TRANSFER TO THE DIGITAL UNIVERSE

As the CHRIS architecture demonstrates, a robot can learn how to speak about a perceptual entity like a human agent picking up an object [26] and how to participate in a cooperative game like uncovering a toy [20]. In the same way, a virtual assistant could learn how to recognize and speak about an e-mail and how to cooperatively send an e-mail.

In the following, we show how to adapt the techniques employed in the robotic domain to build a virtual assistant.

1) *Primitives:* We adapt the set of symbolic primitives to a digital world so that the system can observes the human actions on the GUI. We thus develop and exploit perceptual primitives that correspond to filling fields, clicking, highlighting, copying, and pasting.
2) *Grounding:* We show how the language is linked to these primitives to compose new actions thanks to user demonstrations.
3) *Language and Cooperation:* We show how a user can teach the system to interpret compositional sentences using natural language instructions that refer to learned construction and how the system can generalize a procedure over a set of arguments.

A. Overall Functioning

The virtual assistant architecture, shown in Fig. 2, is made up of the following components.

1) The *Environment* which renders the digital services in a model that the assistant can perceive and act upon. This includes a chat window.
2) The *SensorProcess* which converts percepts to meaningful representation thanks to the *ConstructionMemory*.
3) The *InterpreterProcess* which handles the interactions with a user in order to cooperatively execute procedures and to learn them. This uses the *ContextMemory* that is a working memory for the current tasks and the *EpisodicMemory* which stores event traces that occur within the platform. This is used in the one-shot learning to encode new constructions.
4) The *MotorProcess* which converts meaningful representations to executable actions in the environment.

In order to illustrate the information flow we first provide a general description of a usage scenario. This will then be explained in detail in the following sections.

> A user requests to send an e-mail in natural language, "send an e-mail to peter.dominey@inserm.fr about the meeting and say that I will be early." If the virtual assistant does not know this action, a learning behavior is activated. Then, the user can describe the steps that compose this action using either (a) natural language (using language constructions that have been previously learned), such as "create a draft," "set the subject," "write the content," etc., or (b) by performing a demonstration on a graphical user interface that the virtual assistant can observe. The learning behavior stores all perceptual events coming from the *SensorProcess* including user utterances in the *EpisodicMemory*. At the end of the learning, an algorithm consolidates this knowledge from the *EpisodicMemory* and stores the resulting new construction in the *ConstructionMemory*. The new construction is a map from the language form to the execution meaning. When the user again requests the execution of this action "send an e-mail to clement.delgrange@inserm.fr about the software and say that it functions perfectly," the virtual assistant will match the utterance with the learned construction, extract the arguments contained in the user utterance and instantiate the corresponding procedure. The procedural behavior is then activated. It leads the interaction with the user, monitors the *SensorProcess* and executes primitive actions thanks to the *MotorProcess* in order to complete the procedure. We describe in more details the *SensorProcess* and the *MotorProcess* models, the meaning extraction algorithm and the one-shot procedural learning algorithm in the next sections.

B. Shared Environment and Primitives

1) Principle: In robotics, the human and the robot share a commons space where the grounding of meaning in the perceptual and motor modalities relies on the physics of objects in space. The perceptual model represents objects in the world with attributes, such as a position and an orientation while the motor model defines the most primitive commands that the effectors of a robot can implement, such as to point, grasp, move, or release [19]. These primitive commands have regular consequences in the perceptual model in term of position and orientation, helping for plan recognition, procedural learning and finally, meaning extraction.

For the virtual assistant, the perceptual model should share the same characteristics as in robotics, that is, it must be domain independent, there must be a common space for the user and the agent where this latter should be able to learn by

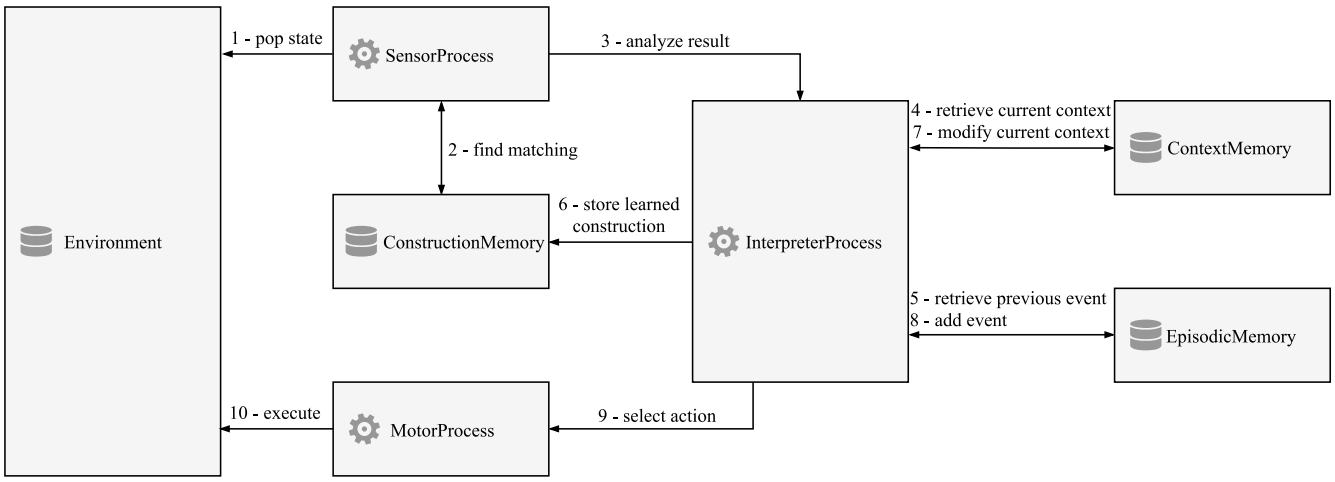

Fig. 2. Virtual assistant architecture overview: During an interaction, the *SensorProcess* pops the state (e.g., user request send an e-mail), and then searches for a match of this pattern in the *ConstructionMemory*. If a match is found, the *InterpreterProcess* begins to select actions from the retrieved sequence of actions, to execute via the *MotorProcess*. If no match is found for the user request, the system asks for a demonstration. Demonstrated events are stored in the *EpisodicMemory*, and then when the demonstration is finished, the sequences of events is consolidated via one-shot learning, and stored as construction that pairs the request (form) with the demonstration (meaning).

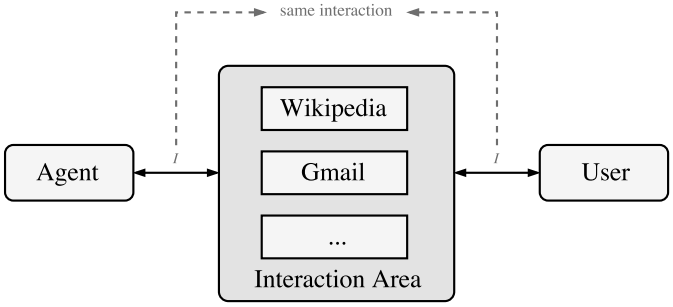

Fig. 3. Shared environment: The interaction area allows the agent to perform actions that can be observed by the agent during learning by demonstration, and that can be performed by the agent and observed by the user.

demonstration by observing the user actions, e.g., filling the "subject" field in an e-mail, and easily transfer its observations to its own motor commands. Fig. 3 illustrates the shared environment between a virtual assistant and a user as a shared environment representation and a shared observation area.

The shared environment representation is an abstraction of a GUI-based system interaction. We model the most primitive interactions that a user performs with a GUI, which are the ability to read and write textual information in a field, and to trigger a system transition (i.e., to enter text into a field, and click in a button). From the shared space, the virtual assistant is able to monitor and act on this representation independently of any natural language skills. We define the perceptual primitive structures as being frames, data, fields, and actions.

Frame: A frame is used to structure the information provided by the system and gather related information. It is composed of a label and a set of the other elements: frame, field, data, and action. It could be viewed as a tree structure of information and thus, each component has a parent reference. For example, all the paragraphs in a *Wikipedia* article.

Data: A data gives the current state of a resource. It is composed of a label, a list of values which stands for the current state and a parent frame. For example, the content of a paragraph in a *Wikipedia* article.

Field: A field describes and holds textual information provided by the user. It is composed of a label, a list of values which hold the current information and a parent frame. For example, the search input box on the *Wikipedia* website.

Action: An action defines a possible transition from the current state. It is composed of a label and a parent frame. For example, clicking on the search button on the *Wikipedia* website.

In complement to these perceptual structures, we define a set of primitive motor commands.

1) *Fill:* Used to fill in a field with a natural language text in the environment.
2) *Execute:* Used to trigger an action in the environment (clicking on a button).
3) *Speak:* Used to send utterance via the chat.

The shared observation area allows the virtual assistant to observe the user actions. We developed a synchronization tool which maintains the consistence between the *Environment* module of the virtual assistant and a local or remote system interface. In this manner, the virtual assistant can read and act similarly on its own digital services and those that a user connects to it. Our synchronization tool has been adapted on top of a rich client application (RCA), Web-based applications, and a Rest[1] API.

Thus, we have a shared space where observations come from and the perceptual and motor primitive structures that correspond to a kind of Mandler's primitive schemas on which language is grounded. We will now define how the effective observations are converted to these structures.

2) Interactions Between the Environment and the SensorProcess Modules: Concrete observations diverge depending if the virtual assistant is interacting with an RCA, a Web application, or a Rest API. In this paper, we employ a

[1]Representational state transfer.

direct mapping between these concrete types of observation and our primitive structures model. We respectively, relied on a specific graphical library, HTML parsing methods and Json resources description for these conversions. The *Environment* module acts as an abstraction of these different types of conversion and is represented as a queue of events which will be popped by the *SensorProcess*. The events are the following.

NewUtterance: This event is created each time a new natural language message is transmitted to one of the actors. It is composed of the message content and its source, either the agent or the user. For example "user: search for Boston."

ExecuteEvent: This event is generated each time an *Action* element is triggered in the environment model. It is composed of the element and its source, either the agent or the user. For example, when the user triggers the "search" button in *Wikipedia*.

FillEvent: This event is generated each time a *Field* element is updated in the environment model. It is composed of the element, its updated value and its source, either the agent or the user. For example, when the user fills the query field in *Wikipedia*.

FocusEvent: This event is generated each time an element is focused in the environment model. It is composed of the focused element and its source, either the agent or the user. We developed a special focus function that allows the user to indicate a particular item of interest in a GUI, e.g., a paragraph on a *Wikipedia* page. This will be of use when the user needs to demonstrate to the assistant to do something with the result of an action (e.g., to paste the result of a *Wikipedia* search into an e-mail).

AddEvent and DeleteEvent: These events are generated each time the structure of the environment model changes. As previously described, the environment model can be represented as a tree structure, those events are composed, respectively of the added leaf nodes or deleted leaf nodes (*Action*, *Data*, *Field*) and a reference which is a label path from the root node. Fig. 4 shows an example of the environment model structure when a *Wikipedia* page is loaded in a browser.

The *Environment* module is made up of the digital services and utterances produced by the user and the virtual assistant. When the *SensorProcess* pops events, those coming from the digital services will remain unchanged as they are already represented in term of our primitive structures however utterances will have to be mapped on those structures as explained in the next section.

C. Language Grounding

1) Principle: As stated in Section II-B, language understanding is defined as a process of mapping elements from a form space (e.g., natural language expressions) into a meaning space (e.g., a sequence of actions). Three kinds of representation must be taken into account, one to model elements in the form space (user utterances), one to model elements in the meaning space (primitives functions) and one to model data used by the process itself linking the form and meaning, called

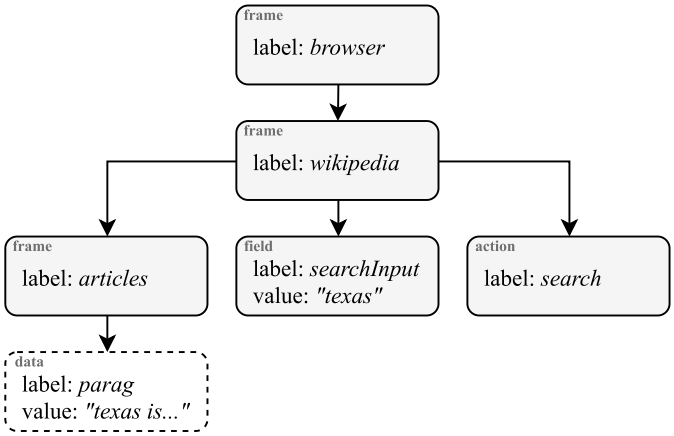

Fig. 4. Example of the environment model that illustrates a *Wikipedia* page loaded in a browser. The hierarchical structure forms label paths from the root node to the leaf nodes. In the current case, when an article will be fully loaded, an *AddEvent* will be dynamically generated and will contain a new *DataNode* with the label path set to *browser/Wikipedia/articles/parag*.

constructions. This is based on the construction grammar framework as characterized by Goldberg [4].

Meaning is extracted from goal directed utterances that command the agent to perform an action, such as "send an e-mail to Peter." The main learning capability of the virtual assistant is to learn how to map such utterances to a sequence of primitive functions that execute the intended goal. The structured sequence of primitives that results from this interaction with the human in a goal directed activity is thus associated with the surface form of a grammatical construction. This is similar to the pragmatic frame characterization of language learning developed by Rohlfing *et al.* [34]. This sequence corresponds to a procedure in our meaning representation.

Mandler sought to define a level of representation for the PMA that corresponds to the sensory capabilities of the developing child. In this paper, we make an analogy between the developing child in the physical world, and an adaptive agent in a digital world of GUIs. In that world, we identified a set of sensory-motor actions that we refer to as primitives because they are the lowest level actions that a user can perform on a GIU and they are independent of the particular application. Thus, in a similar way, we define six primitive functions that compose a procedure. Three of these primitive functions are the primitive motor commands defined in Section III-B1 (fill, execute, and speak) and we define three additional functions of a different nature.

1) *Focus:* Used to highlight an element in the environment.
2) *Retain:* Used to map an highlighted element with a natural language expression.
3) *End:* Used to trigger the one-shot learning algorithm.

These three later functions have been introduced for the virtual assistant functioning only and further described in the next sections. The *Focus* is used to direct attention, the *Retain* is used to tell the assistant to memorize something, together they can be used as a form of copy-past for passing information between operations. The *End* is used to indicate the end of a learning stage.

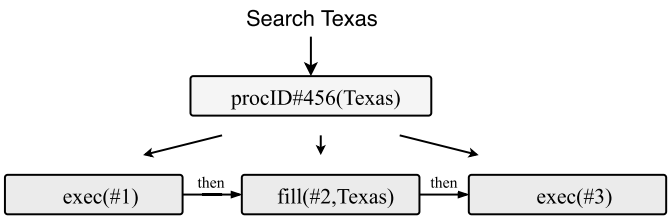

Fig. 5. Example of a learned grammatical construction grounded in the primitive functions.

The first step is to ground language directly on these six primitives. As illustrated by Fig. 5, where "Search Texas" is mapped to a learned predicate-argument structure (a procedure in our meaning representation) which in turn is grounded to the primitives.

Then, Fig. 6 illustrates how a complex procedures can be composed from existing procedures in a hierarchical structure that is finally grounded in the six primitive functions. Procedures are always grounded in these six functions, but when a new procedure is being learned, it can be defined in terms of previously learned procedures and the primitives.

A procedure is thus an ordered sequence of primitives functions with variables which are learned when a new construction is learned. In that respect, variables are created at the time a construction is learned and instantiated when a construction is used. For example, in the learned construction of Fig. 5, Search Texas is an instance of the form component "Search VAR" where VAR is a variable set to "Texas," and the procedure, or the sequence of fill-exec actions, is the meaning component. This procedure has one variable (VAR) which is bound the *Fill* primitive function.

Variables are thus created when a construction is learned. For example, the user says "Look up the definition of Boston," and the system determines that there is no known construction for this, and asks the user to demonstrate. During the demonstration, the user says "Search Boston," which corresponds to a known construction. During processing, the system matches "Boston" in this known construction, with Boston in the initial construction that is being learned. This match of the argument Boston across the two situations allows the system to create a variable in the new learned construction "Look up the definition of VAR." This learning takes place with a single example, in the one-shot learning mechanism that is described in more detail below. In the subsequent executions, the value of VAR will be passed to the previously learned procedure Search VAR.

Then, variables are instantiated by extracting the arguments from a user input utterance. For example, if a user says "Look up the definition of Paris," VAR will be instantiated with "Paris." This process is explained in the next section.

2) Interactions Between the SensorProcess and the ConstructionMemory Modules: In a construction such as "Send an e-mail to peter@gmail.com about the meeting and say that I will be ready as planned," the system learns a form of complex item-based construction "Send an e-mail to VARX about VARY and say that VARZ." In this case VARX, VARY, and VARZ correspond to the recipient, the subject and the message content of an e-mail.

The *ConstructionMemory* stores the constructions as form-to-meaning mappings. It is a map of user utterance event patterns to procedures. As just illustrated, the patterns contain place holders for variables which are bound to the procedure variables when the procedure is instantiated or invoked. This notion of construction is borrowed from the domain of human linguistics, where grammatical constructions are mappings from sentence forms to meanings [4], [27]. In this context, we have exploited the distinction between closed class words (grammatical function words) used in the form characterization of constructions, and open class words (nouns, verbs, adjectives, etc.) that instantiate variables in the constructions in learning grammatical constructions [38], [39].

The *SensorProcess* uses this form to meaning mapping to find a match with the current user utterance event and the set of mappings in the *ConstructionMemory*. For example, when the user utterance is "send an e-mail to clement@gmail.com about this paper and say finish it," the *SensorProcess* compares this with the set of patterns in the *ConstructionMemory*. When a match is found, the pattern variables are instantiated and propagated to the procedure variables. The current utterance, the instantiated pattern variables and the decontextualized procedure constitutes the meaning which is forwarded to the *InterpreterProcess*.

As in the action learning system of Lallée *et al.* [26], there is no predefined predicate representation, they are learned in interaction with the user. This process is explained in the next section.

D. Using and Learning Language in Cooperative Procedure

The *InterpreterProcess* is responsible for maintaining the state of the virtual assistant coherent with the scenario it is engaged in with the user. This includes scenarios in which the user can cooperatively execute a plan of actions with the virtual assistant, and scenarios in which the virtual assistant learns a new plan of actions. For this, the *InterpreterProcess* has access to all the modules on which it can execute primitive actions and retrieve the current state. This includes the *ConstructionMemory*, the *ContextMemory*, the *EpisodicMemory*, and the *MotorProcess*.

1) Interaction With Other Modules:

a) ContextMemory: The *InterpreterProcess* (see Fig. 2) works as a state transition system. States are characterized by two variables in the *ContextMemory*. One holds the current procedure instance being executed, and the other holds the current phase of the virtual assistant which can be the following.

1) *Available:* Indicates that the virtual assistant is not engaged in a learning or executing scenario.
2) *Learning:* Indicates that the virtual assistant is learning a new plan of actions.
3) *Executing:* Indicates that the virtual assistant is executing a plan of actions.

The content of the *ContextMemory* and the meaning result in inputs will determine the next transition to perform.

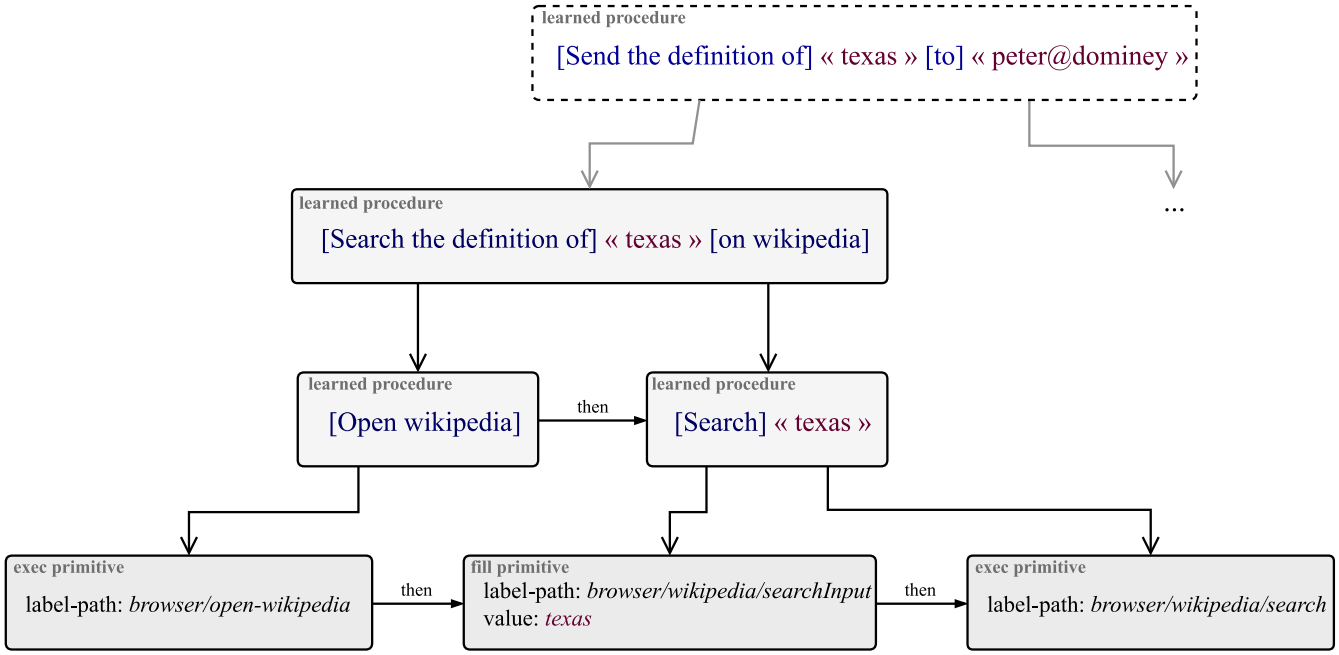

Fig. 6. Learned procedure illustrating compositional meaning representation. At the lowest level, the procedure is grounded in the six primitive functions—fill, execute, speak, focus, retain, and end. Here, we see two "exec" and one "fill" actions. At the highest level, a procedure is defined purely in terms of previously learned procedures, which finally ground out in the six primitives. This illustrates the possibility for compositional structure in defining new procedures.

b) EpisodicMemory: As a new procedure is being learned, the *InterpreterProcess* stores each executed primitive action in the *EpisodicMemory*, as well as each transformation of the environment model and the result of the *focus* and the *retain* primitive actions. This information is used by the learning algorithm to replay all the executed actions during the learning phase, so that these actions can then constitute the meaning component of the new learned construction. Note that during learning, the user can invoke primitive functions, and can also invoke learned procedures that are grounded in primitive functions, as illustrated in Fig. 6.

c) ConstructionMemory: The *ConstructionMemory* is updated when the learning algorithm produces a new construction. This employs a "one-shot" learning method, where the demonstration that is encoded in the *EpisodicMemory* is transformed into a procedure with variables that corresponds to the meaning component of the form-to-meaning mapping. Then the *ConstructionMemory* is also used by the *SensorProcess* to determine if a current utterance produced by the user matches with a learned construction. In this case, the learned construction is instantiated and executed.

d) MotorProcess: The *MotorProcess* is used to execute the primitive motor actions, during learning and execution of a learned construction.

2) InterpreterProcess Operation Modes: The virtual assistant can be engaged in two main modalities: one when it executes a procedure and one when it learns a new one.

a) Execution mode (executing a learned procedure): The simplest situation is when the virtual assistant executes a procedure and the user provides all the context in one utterance. For example: "search Texas on Wikipedia." In this situation, the *InterpreterProcess* will execute the learned procedure that was retrieved when the utterance was matched in the *ConstructionMemory*. At this stage, the virtual assistant will be in an "available" state. The system will instantiate the procedure in the retrieved meaning, and propagate the variables. That is, all the primitive actions will be grounded. The virtual assistant will execute the procedure and report when it is finished. Execution of the procedure will correspond to the following events.

1) Fill("browser/Wikipedia/searchInput," Texas).
2) Execute("browser/Wikipedia/search").
3) Speak("Ok, It is done") [*At this stage, the GUI is showing the article*].

Executing a Learned Procedure With Missing Information: A second situation is when the user does not provide all the context in the utterance, such as "Search the definition of a term." The result meaning will be retrieved as a procedure for searching on *Wikipedia*, similar to the previous example, except that the term of the search is missing. This will leave the variable of the *Fill* primitive unbound in the retrieved procedure. In such a case, where an unbound variable is encountered, the *InterpreterProcess* will generate a *Speak* primitive action in order to request the missing variable such as: "What is the query?" The next user utterance will interpreted as the value of the missing variable. This leads to the third situation, when the virtual assistant does know yet how to request the missing variable.

b) Learning mode (learning how to request information): In order to generate the utterance, the virtual assistant must learn the mapping between the environment model element and the natural language pattern. For this, the *InterpreterProcess* will first ask the user to explain an element of its interface either by natural language such as "What is that: /browser/Wikipedia/searchbox/searchInput?" or by GUI pointing. The response from the user will be bound to the

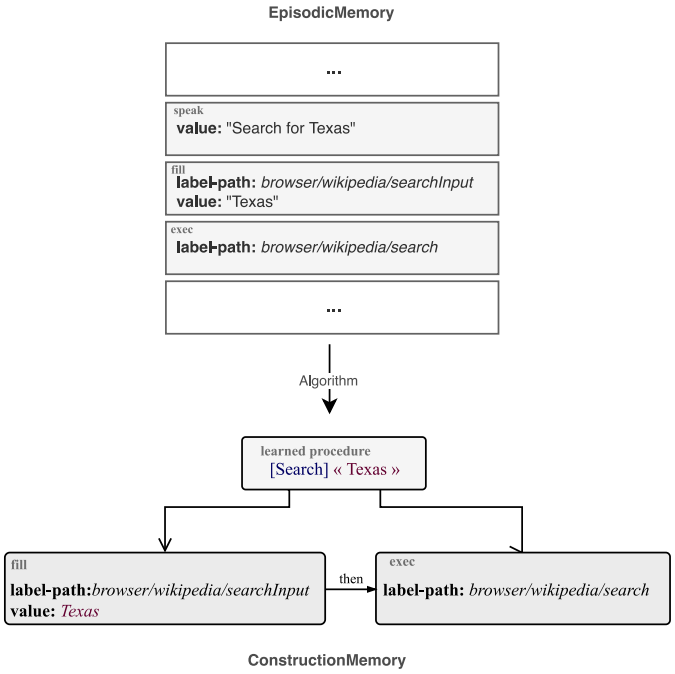

Fig. 7. One-shot learning algorithm applies on the example Search Texas.

element reference in the *ConstructionMemory* and will be reused for a next iteration as in the second scenario just illustrated in Section III-D2a.

Learning a New Procedure With a One-Shot Algorithm: Learning scenarios are triggered when the meaning result does not contain a procedural meaning representation and the current state is available. In such case, if the user produces an utterance like Search Texas on Wikipedia, the system will invite the user to either explain what the system has to do or to provide a GUI demonstration and mark in the *EpisodicMemory* the beginning of the learning phase.

During learning, the meaning representations are of the same format for procedures evoked by natural language utterances and for environment model events demonstrated on the GUI. Thus, the *InterpreterProcess* will handle these representation in the same way, except that for procedures evoked by language, the virtual assistant will execute the primitive action, whereas for GUI user demonstrations it is the user. Note that the use of existing procedures to define new procedures is a powerful mechanism for compositionality [45]. The virtual assistant logs all events occurring in the environment in the *EpisodicMemory*, including those from executing learned procedures during the teaching.

When the procedure is completed, the user must trigger the one-shot learning algorithm with a sentence indicating the end of the learning step. This sentence will be bound to the primitive function *End*. As just described, during learning, the virtual agent stores all events corresponding to the execution of primitive motor functions and perceptual updates. At the end of the learning, the algorithm creates the new construction as illustrated in Fig. 7. The algorithm will generalize by creating variables in the construction. During the learning of a new procedure, when the user is explaining, if he/she repeats a string that is present in the original request, then that string is

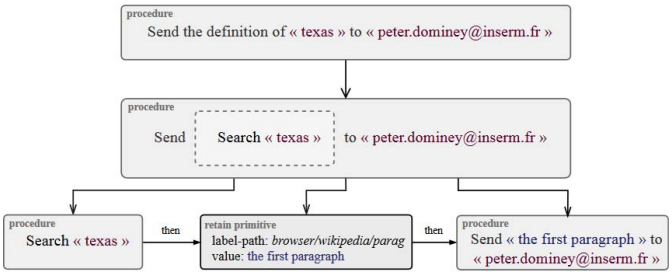

Fig. 8. Procedure illustrating the compositional meaning of two distinct sub procedures.

interpreted as a variable. For example, if the user says Search Texas, the system asks "What do you mean?" and the user shows that you open Wikipedia, and then type Texas in to the search window, Texas is identified as a string that is actually a variable in the learned construction "Search VARX."

3) Motor Process: The role of the *MotorProcess* is to transform primitive motor function representations into execution units in the environment. As for the sensory event representation, these representations are considered to be innate primitives, and are specified at the conception phase and must be kept stable along the life-cycle of the virtual agent, as all the learned procedures rely on them. The transformations are simple algorithms that apply the function representation into the environment. This will in turn generate observable events that will be processed by the *SensorProcess*.

For example, the *Execute* primitive function will be apply to an *Action* element in the environment model, this will generate an *ExecuteEvent* that will be processed by the virtual assistant as well as other event generated by the environment itself. Concretely, this corresponds, for example, to clicking on the search button after filling in the search target in *Wikipedia*.

E. Compositionality

The previous sections explained how each module of the architecture interact to learn new procedures. In this section, we show in more detail how the system can learn to interpret a compositional sentence.

As explained, the *InterpreterProcess* allows the combination of multiple learned procedure into a new one. In order to compose these sub procedures, the *Focus* and *Retain* primitive functions can be used to temporarily save the evaluation of one procedure into a variable and to reuse it in another sub procedure. This allows to learn how to interpret such utterances as: "Send the definition of Texas to peter.dominey@inserm.fr" where "the definition of Texas" must be evaluated and applied to "send x to peter.dominey@inserm.fr" as shown in Fig. 8. In this example, the user can combine the previously learned procedures Search Texas and "Send an e-mail to peter.dominey@inserm.fr."

In this case variables are created internally during learning of a new procedure that involves the *Retain* function. For example, after the user told the assistant to search the definition of Texas, the user will ask "retain the first paragraph." What this means to the system is that a variable called "the first

paragraph" is created, and it is associated with the function of the *Retain*. The *Retain* is a function that is associated with the *Focus*, where the user focuses attention on some object in the page with a green icon that can be moved on the page. In this case, the user focuses attention on the first paragraph of the resulting Wikipedia definition. The retain the first paragraph command thus associates the contents of the focus (the first paragraph of the Wikipedia page) with the variable called the first paragraph. Later in the scenario, when the assistant asks "What is the message?" and the user replies the first paragraph, the utterance will be evaluated as pointing to a variable and the contents of the variable named the first paragraph is pasted into the message frame of the mail. In this manner, the variable called the first paragraph will always be associated with the contents of the first paragraph of the Wikipedia page that is, opened during the execution of the "Send the definition of *X* to *Y*" construction.

IV. Current Performance

The main objective of this paper is the design of the representations, the algorithms, and the architecture for an adaptive assistant that exploits certain functionality of usage-based learning of constructions. To evaluate this objective, we concentrate on a functional/behavioral evaluation. We thus, explain in detail several use cases that illustrate the usage-based functionality of the system. We also provide a quantitative analysis of the potential savings in user intervention that idealized performance of such a system could provide.

A. Use Cases

In this section, we provide examples of typical relevant use case scenarios that a user can achieve with our implementation based on the *Wikipedia* and the *Gmail* website. We present 4 scenarios in chronological order. The scenarios begin when the user connects for the first time with its virtual assistant, and in the final scenario the system can interpret the utterance "Could you send an e-mail to user@domain.com and write him something about Texas."

a) Scenario 1: The user will teach two basic *Gmail* actions to its virtual assistant: how to create a new draft and how to set the subject. In this situation, the user is connected with its virtual assistant with an opened chat box and is already connected to its *Gmail* inbox.

User	Create a draft
Assistant	What do you mean?
User	The user creates a draft on the Gmail GUI. (The system perceives the event primitives, generated by the user's actions in the GUI, and stores them in the *EpisodicMemory*).
Assistant	Ok *(the virtual assistant has received events from the GUI)*
User	End *(the user indicates the end of the learning procedure)*

At this point, when the user says "End," a new construction is created in the *ConstructionMemory* with Create a draft as the form component, and the sequences of events stored in the *EpisodicMemory* as the meaning component.

User	Set the subject
Assistant	What do you mean?
User	*The user fills the subject field on the Gmail GUI*
Assistant	Ok
User	End

Again, when the user says End, a new construction is created in the *ConstructionMemory* with Set the subject as the form component, and the sequences of events stored in the *EpisodicMemory* as the meaning component. Similarly, the user can teach other basic actions on *Gmail*, such as opening its inbox, sending a draft, setting the recipients as well as for other services such as searching a term on *Wikipedia*. A video illustrating these examples is available online.[2]

b) Scenario 2: When the virtual assistant needs to know how to speak about an element in the environment it can ask for this element. In this scenario, the user teaches how to request the subject of a message so that the next time the user will not have to interact with its GUI to set the subject of a message. A full video of this example is available online.[3]

User	Set the subject *(this matches with a previously learned procedure, from scenario 1, where the user filled the subject field.)*
Assistant	What does "/browser/Inbox/Subject" mean? *(In the current execution, the subject field in the GUI is not yet set, and this leads to an unbound variable corresponding to the subject input box in the Gmail GUI that can also be highlighted)*
User	The subject of the message *(the user specifies that this field is the subject of the message)*
Assistant	What is the subject of the message? *(this is still an unbound variable, so the assistant asks for the content)*
User	Meeting schedule
Assistant	Ok, It is done.

c) Scenario 3: After some learning, the virtual assistant is able to perform basic actions on both the *Wikipedia* and the *Gmail* Web sites. In this scenario we show how a user can teach a new procedure that combines previously learned procedures on different task domain as illustrated in an online video.[4]

User	Send the definition of Boston to cl***ge@gmail.com
Assistant	What do you mean?
User	Go to Wikipedia
Assistant	Ok, It is done
User	Search for Boston
Assistant	Ok, It is done
User	retain the first paragraph

[2]Usage-based Learning Virtual Assistant [Part 1: Learning basic actions] https://www.youtube.com/watch?v=Lj_CW0OIU88&list=PLKnvrLyZkzbaNzpcz_ql3HS-mLvyOu1mB&index=2.

[3]Usage-based Learning Virtual Assistant [Part 2: Learning GUI elements] https://www.youtube.com/watch?v=TthKpUKlrck&list=PLKnvrLyZkzbaNzpcz_ql3HS-mLvyOu1mB&index=3.

[4]Usage-based Learning Virtual Assistant [Part 4: Learning complex tasks] https://www.youtube.com/watch?v=Kxeke7tQ7ik&index=5&list=PLKnvrLyZkzbaNzpcz_ql3HS-mLvyOu1mB.

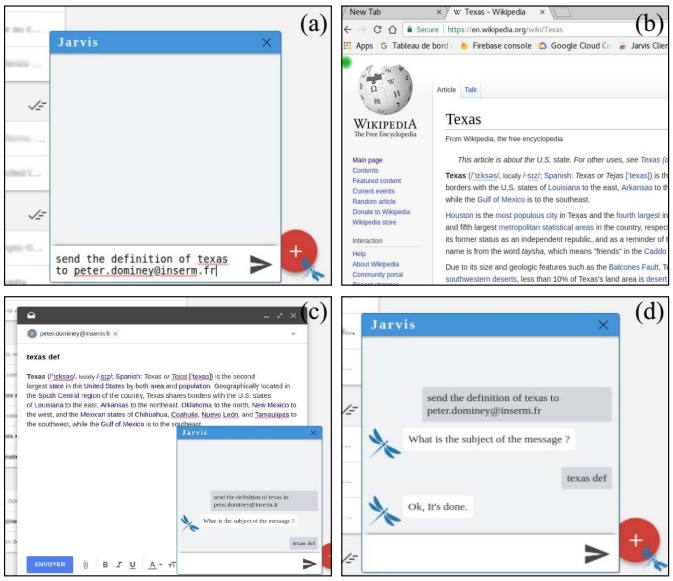

Fig. 9. (a) User makes a request in the chat window "send the definition of Texas to peter.dominey@inserm.fr." (b) System recognizes this construction, and executes the learned procedure: opens *Wikipedia*, writes Texas into the search input. (c) Once the definition is loaded, copy-pastes the first paragraph of the definition into the body of the e-mail. (d) Asks the user for the subject and then sends the message.

User	*The user uses the focus to point the element to retain in its GUI interface.*
Assistant	Ok, It is done *(The system binds the indicated paragraph to the variable the first paragraph)*
User	Close Wikipedia
Assistant	Ok, It is done
User	Send an e-mail to ce***ge@gmail.com
Assistant	What is the subject of the message? *(Note that here the assistant asks a question in order to establish the binding for this variable)*
User	Boston's definition
Assistant	What is the message?
User	The first paragraph
Assistant	Ok, It is done. (The contents of the variable the first paragraph is set as the contents of the message as described in Section III-E)
User	End

Fig. 9 provides a view of the GUI interface during the unfolding of this execution of the virtual assistant.

The result of this command generates a mail which effectively contains the Wikipedia definition of Boston. This learning scenario demonstrates how a new construction can be learned, with a rather complex sentence as the form component, and a structured set of primitive commands as the meaning component. Importantly, when the user explains what this sentence means, she can reuse existing learned procedures by evoking them with natural language, as well as using more primitive actions on the GUI. This illustrates the ability of the system to learn compositional structures. It is worth noting that the system immediately generalizes to use of the same construction with different arguments. Thus, the user can now say "Send the definition of Texas to john@gmail.com" and the system will execute the learned procedure appropriately.

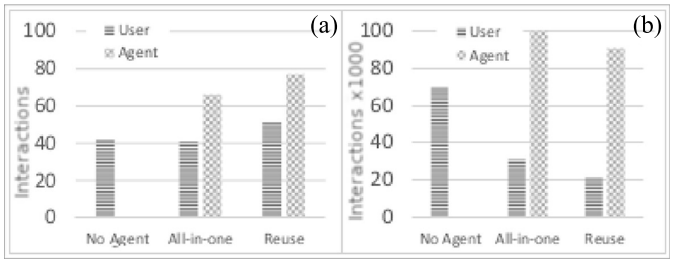

Fig. 10. System evaluation: number of actions taken by the user and the agent to execute learned actions. We compare three strategies, when the user do not use the agent, when he does not reuse learned procedures to learn new procedures and when he does reuse previously learned procedures. (a) Represents the number of user and agent interactions during the process where the user teaches the agent three tasks. (b) Represents the number of user and agent interactions after the activity in (a), when the user teaches the agent 100 new tasks based on reuse of previously learned tasks.

d) Scenario 4: The virtual assistant can learn how to interpret more elaborated sentence in the same manner it learns procedures. That is, we can say a different sentence that has the same meaning as a previously learned procedure, and then when the system asks "What do you mean" we state the meaning as in the learned formulation. This way a previously learned procedure can be reformulated. Here, we show the final result and an online video highlights this feature in a context of text only interactions on a smartphone[5]:

User	Could you send an e-mail to cl***ge@protonmail.com and write him something about Texas
Assistant	What is the subject of the message?
User	Texas definition
Assistant	Ok, It is done

These scenarios illustrate how a complex set of primitive commands can become associated with a high level invoking command, in the form of a grammatical construction encoded in the *ConstructionMemory*.

B. Idealized Performance

We show in this section how the system can be effective by reducing the number of interventions of the user. Fig. 10 illustrates how learning, and reuse of learned procedures in new procedures, reduces the number of interventions by the user. We compare three strategies the user can take as follows.

No Agent	The *No Agent* strategy is when the user does not use the agent and has to perform all the actions on its GUI.
All-In-One	The *All-in-one* strategy is when the user minimizes natural language interactions during the learning, that is, learning by demonstration is always preferred and there is no reuse of previously learned procedures when learning a new compound task.
Reuse	The *Reuse* strategy is when the user maximizes natural language interactions during the learning, that is, learning by explanation is always

[5]Usage-based Learning Virtual Assistant [Part 6: Use learned skills on a smartphone] https://www.youtube.com/watch?v=2m0YUBIetKc&index=7&list=PLKnvrLyZkzbaNzpcz_ql3HS-mLvyOu1mB.

preferred and the user tries to maximize the reuse of the previously learned procedures by structuring the learning from the most fine grain tasks to the most compound tasks.

To evaluate these three strategies, we first count the number of interactions (acting on a GUI or speaking to the agent count both for one interaction) to complete a set of three tasks (search on Wikipedia, send an e-mail and send the definition of something to someone). Panel A represents the number of interactions needed to complete the three tasks three times which is the maximum number of times after which the agent has totally learned a procedure and where then no interaction linked to the learning step is needed. Panel B represents a projection of the number of interactions needed to complete the two first tasks 100 times and then 100 times 100 different versions of the third task. So the panel B simulates a user who extensively uses what he teaches to his agent and who extensively teaches new compound tasks.

When first teaching the agent how to do things, there is no advantage to the user (panel A) compared to doing the same actions without the agent. When learned actions are repeatedly used and when we simulate the learning of 100 additional tasks (panel B), then, we respectively see a reduction in the user actions with the agent and the benefit from reuse of previously learned procedures. This does not constitute a proper user study, but allows a quantitative view of the potential effectiveness of the system. A noteworthy point is that this paradigm of reuse of learned material in learning new material generates progression in efficiency that increases as learning more potentially reusable material proceeds.

V. Discussion

In this paper, we present a usage-based system where an end-user is able to teach to a virtual assistant how to operate a set of arbitrary digital services using natural language. The virtual assistant thus uses the inherent structure in the user's natural language commands to organize primitive perceptual elements into procedures. This allows the assistant to adapt to different services and to variability in users' choices on their use of language. The learning was initially employed to learn how to ground natural language utterances into domain independent perceptual and motor commands in order to execute basic service actions (e.g., to open *Gmail*, create a draft mail, set the subject, etc.). Based on such learned utterance-action constructions, the virtual assistant then learned how to perform more complex tasks, characterized by the composition of these previously learned constructions and the corresponding procedures, with argument passing between them (e.g., the search term from *Gmail* being passed to *Wikipedia*, and the resulting definition being passed back to *Gmail*). Thus, learned procedures are demonstrated to be used in a recursive or compositional definition of a new procedure like "send an e-mail to pfdominey@gmail.com about programming with the definition of python," that the user can explain in natural language by decomposing into the "send an e-mail" procedure, and another learned procedure that gets definitions from *Wikipedia*. The ability to ground commands in user demonstrated actions on the GUI, and then the ability to compose these learned commands purely by language illustrates how simple recursive mechanisms for creating labeled structures provides a powerful compositional mechanism for specifying complex interactions with the world [45].

We show that by taking a developmental perspective based on grammatical constructions as form-to-meaning mappings [10], [11], [27], while the developed infrastructure is lite, the virtual assistant is able to jointly learn natural language structures together with procedural semantics in realistic usage scenarios. The ability to learn these procedures and to label them is minimalist in terms of its simplicity but it is quite powerful. Indeed, this conception avoids the need for inherent language-specific and domain-specific knowledge engineering, while still providing a substantial learning capability. As we illustrated, in our system, this capability allows the user to progressively create successive levels of hierarchical structure, resulting in powerful, and compact linear strings that represent and re-enact a complex hierarchical structure. Additionally, we introduce the capability for variables and the passage of variable bindings across these different levels of the hierarchical structure. This provides a concrete implementation which illustrates the power of the merge and label capability in human cognition [45].

The ability for a user to teach an assistant with natural language brings several advantages: the user can adapt the assistant to its own needs by teaching new procedural knowledge, and the system is more resilient to faults such as unrecognized words in the lexicon. Perhaps most importantly, with this learning, extending the system in term of functionalities can be done at runtime by the end-user instead of preprogramming commands at the conception phase. At the same time, the end-user experience remains intuitive by keeping the teaching stage as transparent as possible regarding the user's demonstrations. This transparency is enhanced by eliminating the need for the user to learn prespecified language commands, and shielding users from the virtual assistant's internal organization so as not to distract them from their tasks.

By enabling a reduced set of primitives, we have provided a closed set of elements to ground language on. In our study, we thus defined a set of "motor" primitives (*Fill*, *Execute*, . . . ,) and "cognitive" primitives (*Focus*, *Retain*, *End*). While conceptually inspired from human studies, these primitives were primarily identified based on analysis of how users interact with GUIs (motor primitives) and how they manipulate information on GUIs (cognitive primitives), and so, they are tailored to our application case. Future research could address how such primitives could be identified in a more generic way and if they can help for the interoperability of cognitive modules or AI techniques. Well defined, they could be at the interface of many tools, such as image recognition, natural language parsing, a simulator, or a knowledge base.

The major limitation of this system is its rigidity in matching the fixed component of input sentence with learned constructions. Thus, if the system knows the meaning of "send an e-mail to *X* about *Y* and say that *Z*," and the user says "send a mail to *X* about *Y* and say that *Z*," the system will not match

the pattern, despite the similarity to the original sentence. We are currently developing a method to address this limitation, based on machine learning work in the area of semantic relatedness [46]. The system will compare the user's input with existing sentences in the *ConstructionMemory* in order to determine if the request is a variant of a known construction, or if it is a new construction to be learned. This paper will be a key point to make the virtual assistant more resilient to natural language variability, which includes the ability to generalize over synonymy, recurrent dialog parts of speech, different grammatical forms, and to allow user studies which are used to this level of language adaptability. Another related limitation is the rigidity in learning to ask questions. For example, when the agent needs to know how to request the subject of a message, it learns from the user's response "the subject" by prefixing it with the fixed pattern "What is," to generate the question "What is the subject." This has been introduced as a simplification.

Further improvements will be made. First, the capability to generate narrative constructions that allow the virtual assistant to learn how to understand and narrate a sequence of events based on the causal and dependency relations between the constituent events will be adapted from our related work in robotics [47]. Indeed, the ability of narrative to impose additional structure on representations of human experience is crucial in the human ability to represent a reality shared with others [48], [49], and will play an important role in the future development of adaptive assistants.

Another possible extension of the system is to learn the meanings of words based on their situation within a construction. In the construction "Send an e-mail to *X* about *Y* and say that *Z*," the system has access to the information that items which instantiate *X* are e-mail addresses. While we currently do not exploit this, such syntactic bootstrapping (i.e., inferring a referent meaning based on its grammatical configuration) can be used to accelerate learning as we demonstrated in Dominey [41].

VI. Conclusion

We set out to develop an intelligent assistant that can learn from human demonstration in novel Web contexts. A major challenge in this area is to allow the system to adapt to new Web tools without pre-engineering the interface. To address this we borrowed ideas from developmental psychology where it has been suggested that infants have perceptual primitives that can be used to understand actions and events in new contexts (Mandler). We thus, developed perceptual and motor primitives that correspond to filling fields and clicking, copying, and pasting in Web GUI interfaces. This allows the system to observe and repeat actions performed by the user. We borrowed a second concept from cognitive linguistics, the grammatical construction, as a mapping from sentence to meaning. Here, the meaning is characterized in our perceptual-motor primitives. Our learning mechanism thus allows grammatical constructions to be learned from demonstration, and then further constructions to be learned based on previously learned constructions in a recursive hierarchical manner. We illustrated how this compositional learning could ease the work of the teacher as the learned procedures become increasingly complex.

References

[1] M. Korpusik, Z. Collins, and J. Glass, "Semantic mapping of natural language input to database entries via convolutional neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Proces. (ICASSP)*, New Orleans, LA, USA, 2017, pp. 5685–5689.

[2] M. Petit *et al.*, "The coordinating role of language in real-time multimodal learning of cooperative tasks," *IEEE Trans. Auton. Mental Develop.*, vol. 5, no. 1, pp. 3–17, Mar. 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6249732

[3] N. G. Canbek and M. E. Mutlu, "On the track of artificial intelligence: Learning with intelligent personal assistants," *J. Human Sci.*, vol. 13, no. 1, pp. 592–601, 2016.

[4] A. E. Goldberg, *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago, IL, USA: Univ. Chicago Press, 1995.

[5] M. Sun, Y.-N. Chen, and A. I. Rudnicky, "An intelligent assistant for high-level task understanding," in *Proc. ACM 21st Int. Conf. Intell. User Interfaces*, Sonoma, CA, USA, 2016, pp. 169–174.

[6] J. Allen *et al.*, "PLOW: A collaborative task learning agent," in *Proc. Nat. Conf. Artif. Intell.*, vol. 22. Vancouver, BC, Canada, 2007, pp. 1514–1519. [Online]. Available: http://www.aaai.org/Papers/AAAI/2007/AAAI07–240.pdf

[7] C. H. Hwang and L. K. Schubert, "Episodic logic: A situational logic for natural language processing," in *Situation Theory and Its Applications*, vol. 3. Menlo Park, CA, USA: SRI Int., 1993. pp. 303–338. [Online]. Available: http://books.google.fr/books?hl=fr&lr=&id=wlfBUzFzJ8gC&oi=fnd&pg=PA303&dq=Episodic+Logic+expressiveness&ots=vKc3iXc8EI&sig=Hk3XN87GAJwFxeSNs6I_tYu8W8A

[8] G. Ferguson and J. F. Allen, "TRIPS: An integrated intelligent problem-solving assistant," in *Proc. AAAI/IAAI*, 1998, pp. 567–572. [Online]. Available: http://www.aaai.org/Papers/AAAI/1998/AAAI98–080.pdf

[9] A. Azaria, J. Krishnamurthy, and T. Mitchell, "Instructable intelligent personal agent," in *Proc. AAAI*, Phoenix, AZ, USA, 2016, pp. 2681–2689.

[10] M. Tomasello, *Constructing a Language: A Usage Based Theory of Language Acquisition*, vol. 8. Cambridge, MA, USA: Harvard Univ. Press, Jan. 2003.

[11] E. V. Clark, *First Language Acquisition*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[12] I. Nomikou, M. Schilling, V. Heller, and K. J. Rohlfing, "Language-at all times," *Interact. Stud.*, vol. 17, no. 1, pp. 120–145, 2016.

[13] M. Tomasello, "The item-based nature of children's early syntactic development," *Trends Cogn. Sci.*, vol. 4, no. 4, pp. 156–163, 2000.

[14] E. Lieven, H. Behrens, J. Speares, and M. Tomasello, "Early syntactic creativity: A usage-based approach," *J. Child Lan.*, vol. 30, no. 2, pp. 333–370, 2003.

[15] M. Asada *et al.*, "Cognitive developmental robotics: A survey," *IEEE Trans. Auton. Mental Develop.*, vol. 1, no. 1, pp. 12–34, May 2009.

[16] J. M. Mandler, "How to build a baby: III. Image schemas and the transition to verbal thought," in *From Perception to Meaning: Image Schemas in Cognitive Linguistics*. Berlin, Germany: Mouton de Gruyter, 2005, pp. 137–163.

[17] S. Lallée *et al.*, "Towards a platform-independent cooperative human-robot interaction system: I. Perception," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2010, pp. 4444–4451. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5652697

[18] S. Lallée *et al.*, "Towards a platform-independent cooperative human–robot interaction system: II. Perception, execution and imitation of goal directed actions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, San Francisco, CA, USA, 2011, pp. 2895–2902.

[19] S. Lallée *et al.*, "Towards a platform-independent cooperative human robot interaction system: III. An architecture for learning and executing actions and shared plans," *IEEE Trans. Auton. Mental Develop.*, vol. 4, no. 3, pp. 239–253, Sep. 2012.

[20] S. Lallée *et al.*, "Cooperative human robot interaction systems: IV. Communication of shared plans with Naïve humans using gaze and speech," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2013, pp. 129–136.

[21] S. Lallée, V. Vouloutsi, S. Wierenga, U. Pattacini, and P. Verschure, "EFAA: A companion emerges from integrating a layered cognitive architecture," in *Proc. ACM/IEEE Int. Conf. Human–Robot Interact.*, Bielefeld, Germany, 2014, p. 105.
[22] J. M. Mandler, "How to build a baby: On the development of an accessible representational system," *Cogn. Develop.*, vol. 3, no. 2, pp. 113–136, 1988.
[23] J. M. Mandler, "How to build a baby: II. Conceptual primitives," *Psychol. Rev.*, vol. 99, no. 4, pp. 587–604, 1992.
[24] J. M. Mandler, "On the spatial foundations of the conceptual system and its enrichment," *Cogn. Sci.*, vol. 36, no. 3, pp. 421–451, 2012.
[25] P. F. Dominey and J.-D. Boucher, "Learning to talk about events from narrated video in a construction grammar framework," *Artif. Intell.*, vol. 167, nos. 1–2, pp. 31–61, 2005.
[26] S. Lallée, C. Madden, M. Hoen, and P. F. Dominey, "Linking language with embodied and teleological representations of action for humanoid cognition," *Front. Neurorobot.*, vol. 4, p. 8, Jun. 2010. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2889716/
[27] A. E. Goldberg, "Constructions: A new theoretical approach to language," *Trends Cogn. Sci.*, vol. 7, no. 5, pp. 219–224, 2003.
[28] J. Gaspers, P. Cimiano, K. Rohlfing, and B. Wrede, "Constructing a language from scratch: Combining bottom–up and top–down learning processes in a computational model of language acquisition," *IEEE Tran. Cogn. Develop. Syst.*, vol. 9, no. 2, pp. 183–196, Jun. 2017.
[29] P. Dominey, "From holophrases to abstract grammatical constructions: Insights from simulation studies," in *Constructions in Acquisition*, E. Clark and B. Kelly, Eds. Stanford, CA, USA: CSLI, 2006, pp. 137–162.
[30] P. F. Dominey and J.-D. Boucher, "Developmental stages of perception and language acquisition in a perceptually grounded robot," *Cogn. Syst. Res.*, vol. 6, no. 3, pp. 243–259, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389041704000762
[31] A. Cangelosi and T. Riga, "An embodied model for sensorimotor grounding and grounding transfer: Experiments with epigenetic robots," *Cogn. Sci.*, vol. 30, no. 4, pp. 673–689, 2006.
[32] F. Stramandinoli, D. Marocco, and A. Cangelosi, "Making sense of words: A robotic model for language abstraction," *Auton. Robots*, vol. 41, no. 2, pp. 367–383, 2017.
[33] V. Tikhanoff, A. Cangelosi, and G. Metta, "Integration of speech and action in humanoid robots: iCub simulation experiments," *IEEE Trans. Auton. Mental Develop.*, vol. 3, no. 1, pp. 17–29, Mar. 2011.
[34] K. J. Rohlfing, B. Wrede, A.-L.Vollmer, and P.-Y. Oudeyer, "An alternative to mapping a word onto a concept in language acquisition: Pragmatic frames," *Front. Psychol.*, vol. 7, p. 470, Apr. 2016.
[35] L. Steels, "The emergence and evolution of linguistic structure: From lexical to grammatical communication systems," *Connection Sci.*, vol. 17, nos. 3–4, pp. 213–230, 2005.
[36] B. Bergen, N. Chang, and S. Narayan, "Simulated action in an embodied construction grammar," in *Proc. Annu. Meeting Cogn Sci. Soc.*, vol. 26, 2004, pp. 108–113.
[37] Z. Solan, D. Horn, E. Ruppin, and S. Edelman, "Unsupervised learning of natural languages," *Proc. Nat. Acad. Sci. USA*, vol. 102, no. 33, pp. 11629–11634, 2005.
[38] P. F. Dominey, M. Hoen, J.-M. Blanc, and T. Lelekov-Boissard, "Neurological basis of language and sequential cognition: Evidence from simulation, aphasia, and ERP studies," *Brain Lang.*, vol. 86, no. 2, pp. 207–225, 2003.
[39] X. Hinaut and P. F. Dominey, "Real-time parallel processing of grammatical structure in the fronto-striatal system: A recurrent network simulation study using reservoir computing," *PLoS ONE*, vol. 8, no. 2, 2013, Art. no. e52946.
[40] J. Gaspers and P. Cimiano, "A computational model for the item-based induction of construction networks," *Cogn. Sci.*, vol. 38, no. 3, pp. 439–488, 2014.
[41] P. F. Dominey, "Conceptual grounding in simulation studies of language acquisition," *Evol. Commun.*, vol. 4, no. 1, pp. 57–85, 2002. [Online]. Available: http://www.ingentaconnect.com/content/jbp/evco/2000/00000004/00000001/art00004
[42] S. M. McCauley and M. H. Christiansen, "Prospects for usage-based computational models of grammatical development: Argument structure and semantic roles," *Wiley Interdiscipl. Rev. Cogn. Sci.*, vol. 5, no. 4, pp. 489–499, 2014.
[43] M. Tomasello, M. Carpenter, J. Call, T. Behne, and H. Moll, "Understanding and sharing intentions: The origins of cultural cognition," *Behav. Brain Sci.*, vol. 28, no. 5, pp. 675–691, Oct. 2005. [Online]. Available: https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/understanding-and-sharing-intentions-the-origins-of-cultural-cognition/F9C40BF73A68B30B8EB713F2F947F7E2
[44] K. Tylèn, E. Weed, M. Wallentin, A. Roepstorff, and C. D. Frith, "Language as a tool for interacting minds," *Mind Lang.*, vol. 25, no. 1, pp. 3–29, 2010.
[45] T. Goucha, E. Zaccarella, and A. D. Friederici, "A revival of the Homo loquens as a builder of labeled structures: Neurocognitive considerations," *Neurosci. Biobehav. Rev.*, vol. 81, pp. 213–214, Mar. 2017.
[46] E. Agirre. (2018). *Semantic Textual Similarity*. [Online]. Available: http://ixa2.si.ehu.es/stswiki/index.php/Main_Page
[47] A.-L. Mealier, G. Pointeau, S. Mirliaz, K. Ogawa, M. Finlayson, and P. F. Dominey, "Narrative constructions for the organization of self experience: Proof of concept via embodied robotics," *Front. Psychol.*, vol. 8, p. 1331, Aug. 2017.
[48] J. S. Bruner, *Acts of Meaning*, vol. 3. Cambridge, MA, USA: Harvard Univ. Press, 1990.
[49] J. Bruner, "The narrative construction of reality," *Critical Inquiry*, vol. 18, no. 1, pp. 1–21, 1991.