CHAPTER

2

Convology: an ontology for conversational agents in digital health

Mauro Dragoni[1], *Giuseppe Rizzo*[2] *and Matteo A. Senese*[2]

[1]**Fondazione Bruno Kessler, Trento, Italy** [2]**LINKS Foundation, Torino, Italy**

2.1 Introduction

The conversation paradigm has been implemented for the realization of conversational agents overwhelmingly in the last years. Natural and seamless interactions with automated systems introduce a shift from using well-designed and sometimes complicated interfaces made of buttons and paged procedures to textual or vocal dialogs. Asking questions naturally has many advantages with respect to traditional app interactions. The main one is that the user does not need to know how the specific application works, everyone knows how to communicate, and in this case, the system is coming toward the user to make the interaction more natural. This paradigm has been integrated into mobile applications for supporting users from different perspectives and into more well-known systems built by big tech players like Google Assistant and Amazon Alexa. These kinds of systems dramatically reduce the users' effort for asking and communicating information to systems that, by applying natural language understanding (NLU) algorithms, are able to decode which are the actual users' intentions and to reply properly. However, by performing a deeper analysis of these systems, we can observe a strong limitation of their usage into complex scenarios. The interactions among users and bots are often limited to a single-turn communication where one of the actor sends an information request (e.g., a question like "How is the weather today in London?" or a command like "Play the We Are The Champions song") and the other actor provides an answer containing the required information or performs the requested action (e.g., "Today the weather in London is cloudy." or the execution of the requested song).

Web Semantics
DOI: https://doi.org/10.1016/B978-0-12-822468-7.00004-3

While this is true for most of the current conversational agents, the one made by Google seems to be more aware of the possibility of multiturn conversation. In fact, in some particular situations, it is capable of carry a context between one user question and the following ones. An example could be asking "Who is the current US president?" and then "Where he lives?;" in this particular case, the agent resolves the "he" pronoun carrying the context of the previous step. Anyway this behavior is not general and is exploited only in some common situations and for a limited amount of steps. An evidence of this is the limit of the *DialogFlow* platform (a rapid prototyping platform for creating conversational agents based on the Google Assistant intelligence) to maintain context from one step to another (the maximum number of context it can carry is 5). While this mechanism could appear among sentences belonging to the same conversation, it is not true among different conversations, what we noticed is that each conversation is for sure independent from the previous ones. Hence, the agent does not own a story of the entire dialog. Additionally, the assistant does not seem to be conscious about the actual status of the conversation; this marks the impossibility for it to be an effective tool to achieve a complex goal (differently from single interactions like "turning on the light").

This situation strongly limits the capability of these systems of being employed into more complex scenarios where it is necessary to address the following challenges: (1) to manage long conversations possibly having a high number of interactions, (2) to keep track of users' status in order to send proper requests or feedback based on the whole context, (3) to exploit background knowledge in order to have at any time all information about the domain in which the conversational agent has been deployed, and (4) to plan dialogs able to dynamically evolve based on the information that have been already acquired and on the long-term goals associated with users. To address these challenges it is necessary to sustain NLU strategies with knowledge-based solutions able to reason over the information provided by users in order to understand her status at any time and to interact with her properly. Conversational agents integrating this knowledge-based paradigm go one step beyond state-of-the-art systems that limit their interactions with users to a single-turn mode.

In this chapter, we present Convology (CONVersational ontOLOGY), a top-level ontology aiming to model the conversation scenario for supporting the development of conversational knowledge-based systems. Convology defines concepts enabling the description of dialog flows, users' information, dialogs and users events, and the real-time statuses of both dialogs and users. Hence, systems integrating Convology are able to manage multiturn conversations. We present the TBox, and we show how it can be instantiated into a real-world scenario.

The chapter is structured as follows. In Section 2.2, we discuss the main types of conversation tools by highlighting how none of them is equipped with facilities for managing multiturn conversations. Then, in Sections 2.3 and 2.4, we present the methodology used for creating Convology and we explain the meaning of the concepts defined. Section 2.5 shows how to get and to reuse the ontology, whereas Section 2.6 presents an application integrating Convology together with examples of future projects that will integrate it. Section 2.7 discusses the sustainability and maintenance aspects, and, finally, Section 2.8 concludes the chapter.

2.2 Background

Conversational agents, in their larger definition, are software agents with which it is possible to carry a conversation. Researchers discussed largely on structuring the terminology around conversational agents. In this chapter, we decide to adhere to Franklin and Graesser (1997) that segments conversational agents according to both learned and indexed content and approaches for understanding and establishing a dialog. The evolution of conversational agents proposed three different software types: generic chit-chat (i.e., tools for maintaining a general conversation with the user), goal-oriented tools that usually rely on a large amount of prebuilt answers (i.e., tools that provide language interfaces for digging into a specific domain), and the recently investigated knowledge-based agents that aim to reason over a semantic representation of a dataset to extend the intent classification capabilities of goal-oriented agents.

The first chit-chat tool, named ELIZA (Weizenbaum, 1966), was built in 1966. It was created mainly to demonstrate the superficiality of communications and the illusion to be understood by a system that is simply applying a set of pattern-matching rules and a substitution methodology. ELIZA simulates a psychotherapist and, thanks to the trick of presenting again to the interlocutor some contents that have been previously mentioned, it keeps the conversation without having an understanding of what really is said. At the time when ELIZA came out, some people even attributed human-like feelings to the agent. A lot of other computer programs have been inspired by ELIZA and AIML—markup language for artificial intelligence—has been created to express the rules that drive the conversation. So far, this was an attempt to encode knowledge for handling a full conversation in a set of predefined linguistic rules.

Domain-specific tools were designed to allow an individual to search conversationally into a restricted domain, for instance simulating the interaction with a customer service of a given company. A further generalization of this typology was introduced by knowledge-based tools able to index a generic (wider) knowledge base and provides answers pertaining a given topic. These two are the largest utilized types of conversational agents (Ramesh et al., 2017). The understanding of the interactions is usually performed using machine learning, in fact recent approaches have abandoned handcrafted rules utilized in ELIZA toward an automatic learning from a dialog corpus. In other words, the understanding task is related to turning natural language sentences into something that can be understood by a machine: its output is translated into an intent and a set of entities. The response generation can be fully governed by handcrafted rules (e.g., if a set of conditions apply, say that) or decide the template response from a finite set using statistical approaches [using some distance measures like TF-IDF, Word2Vec, Skip-Thoughts (Kiros et al., 2015)]. In this chapter, we focus on the understanding part of the conversation.

While machine learning offers statistical support to infer the relationship between sentences and classes, one pillar of these approaches is the knowledge about the classes of these requests. In fact, popular devices such as Amazon Echo and Google Home require, whether configured, to list the intents of the discussion. However, those devices hardly cope with a full dialog, multiturn, as the intents are either considered in isolation or contextualized within strict boundaries. Previous research attempts investigated the

multiturn aspect with neural networks (Mensio et al., 2018). The conversation was fully understood statistically, that is, through statistical inference of intents sequentially, without a proper reasoning about the topics and actors of the conversation. Other research attempts exploited the concept of ontology for modeling a dialog stating that a semantic ontology for dialog needs to provide the following: first, a theory of events/situations; second, a theory of abstract entities, including an explication of what propositions and questions are; and third, an account of Grounding/Clarification (Ginzburg, 2012). An ontology is thus utilized to also order questions maximizing coherence (Milward, 2004). Despite the research findings on this theme and the trajectory that shows a neat interaction between statistical inference approaches and ontologies for modeling the entire dialog (Flycht-Eriksson and Jönsson, 2003), there is a lack of a shared ontology. In this chapter, we aim to fill this gap by presenting Convology.

2.3 The construction of convology

The development of Convology followed the need of providing a metamodel able not only to provide a representation of the conversational domain but also to support the development of smart applications enabling the access to knowledge bases through a conversational paradigm. Such applications aim to reduce users' effort in obtaining required information. For this reason, the proposed ontology has been modeled by taking into account how it can be extended for being integrated into real-world applications.

The process for building Convology followed the METHONTOLOGY (Fernández-López et al., 1997) methodology. This approach is composed by seven stages: Specification, Knowledge Acquisition, Conceptualization, Integration, Implementation, Evaluation, and Documentation. For brevity, we report only the first five steps since they are the most relevant ones concerning the design and development of the ontology. The overall process involved four knowledge engineers and two domain experts from the Trentino Healthcare Department. More precisely, three knowledge engineers and one domain experts participated to the ontology modeling stages (hereafter, the modeling team). While, the remaining knowledge engineer and domain expert were in charge of evaluating the ontology (hereafter, the evaluators). The role of the domain experts was to supervise the psychological perspective of the ontology concerning the definition of proper concepts and relationships supporting the definition of empathetic dialogs.

The choice of METHONTOLOGY was driven by the necessity of adopting a life-cycle split in well-defined steps. The development of Convology requires the involvement of the experts in situ. Thus the adoption of a methodology having a clear definition of the tasks to perform was preferred. Other methodologies, like DILIGENT (Pinto et al., 2004) and NeOn (Suárez-Figueroa, 2012), were considered before starting the construction of the Convology ontology. However, the characteristics of such methodologies, like the emphasis on the decentralized engineering, did not fit our scenario well.

2.3.1 Specification

The purpose of Convology is twofold. On the one hand, we want to provide a metamodel fully describing the conversation domain from the conversational agent perspective. On the other hand, we want to support the development of smart applications for supporting users in accessing content of knowledge bases by means of a conversational paradigm. As mentioned in Section 2.1, Convology supports the modeling of a full dialog between users and systems.

From the granularity perspective, Convology is modeled with a *low* granularity level. As we discuss in Section 2.4, Convology contains only top-level concepts representing the main entities involved in describing a conversation and that can be used for storing information about user-based events that can be exploited for reasoning purposes. The rationale behind this choice is to avoid changes in the TBox when Convology is instantiated into a new domain. Thus, when a new application is developed, the experts in charge of defining all entities involved in the conversation supported by the application will work only on the ABox.

2.3.2 Knowledge acquisition

The acquisition of the knowledge necessary for building Convology was split in two phases: (1) the definition of the TBox and (2) the definition of the ABox. The TBox has been modeled by the modeling team having also competences in NLU. The modeling activity started by analyzing the requirements for realizing a classic (i.e., single-turn) conversational agents and by defining which kind of information are necessary for supporting the multiturn paradigm.

At this point, the modeling team defined the set of entities playing an important role during the reasoning process. In particular, three concepts have been defined: *UserEvent*, *UserStatus*, and *DialogStatus*. The first one defines events of interest associated with users. Such events are the basic information used at reasoning time. The second one allows to model the status of interest in which a *User* can be and it can be activated at reasoning time in case a specific set of *UserEvent* is verified. Finally, the third one represents a snapshot of a conversation between a *User* and a *Agent* and works as trigger for the system to perform specific actions. In Section 2.4, we will explain each concept and the interactions among them in more detail.

Differently, knowledge defined within the ABox is acquired through the collaborative work with domain experts. Indeed, when Convology is instantiated into a new application, it is necessary to define which are the relevant information (i.e., questions, answers, intents, etc.) used by the conversational agent for managing dialogs. Such information can be provided only by domain experts. Let us consider the sample scenario we reported in Section 2.6 about the asthma domain. There, pulmonologists have been involved for providing all the knowledge necessary for managing a conversation with users in order to collect information needed for supporting a real-time reasoning of their healthy status.

2.3.3 Conceptualization

The conceptualization of Convology was split into two steps. The first one was covered by the knowledge acquisition stage, where most of the terminology is collected and directly modeled into the ontology. While the second step consisted in deciding how to represent, as classes or as individuals, the information we collected from unstructured resources. Then, we modeled the properties used for supporting all the requirements.

During this stage, we relied on several ontology design patterns (Hitzler et al., 2016). However, in some cases, we renamed some properties upon the request of domain experts. In particular, we exploit the logical patterns *Tree* and *N-Ary Relation*, the alignment pattern *Class Equivalence*, and the content patterns *Parameter*, *Time Interval*, *Action*, and *Classification*.

2.3.4 Integration

The integration of Convology has two objectives: (1) to align it with a foundational ontology and (2) to link it with the Linked Open Data (LOD) cloud. The first objective was satisfied by aligning the main concepts of Convology with ones defined within the DOLCE (Gangemi et al., 2002) top-level ontology. Concerning the second objective, although it is not addressed by the TBox of Convology, it can be satisfied when Convology is integrated into specific application and some of the intents can be aligned with concepts defined in other ontologies. As example, if Convology is integrated into a chat-bot supporting people about diet and physical activity, instances of the *Intent* concept can be aligned with concepts defined within the AGROVOC[1] vocabulary. Similarly, the integration of Convology, proposed in Section 2.6, into a conversational agent supporting people affected by asthma opens the possibility of aligning instances of the *Intent* concept with concepts defined into an external medical knowledge base like UMLS[2].

This way, individuals defined within the ABox of Convology may work as a bridge between Convology and the LOD cloud.

2.4 Inside convology

The ontology contains five top-level concepts: *Actor*, *ConversationItem*, *Dialog*, *Event*, and *Status*. Among these, the *Dialog* concepts does not subsume any other concept. However, it works as collector of other concepts for representing a whole dialog instance. Fig. 2.1 shows a general overview of the ontology with the hierarchical organization of the concepts.

Below, by starting from each top-level concept, we detail each branch of Convology by providing the semantic meaning of the most important entities.

[1] http://aims.fao.org/vest-registry/vocabularies/agrovoc.

[2] https://www.nlm.nih.gov/research/umls/.

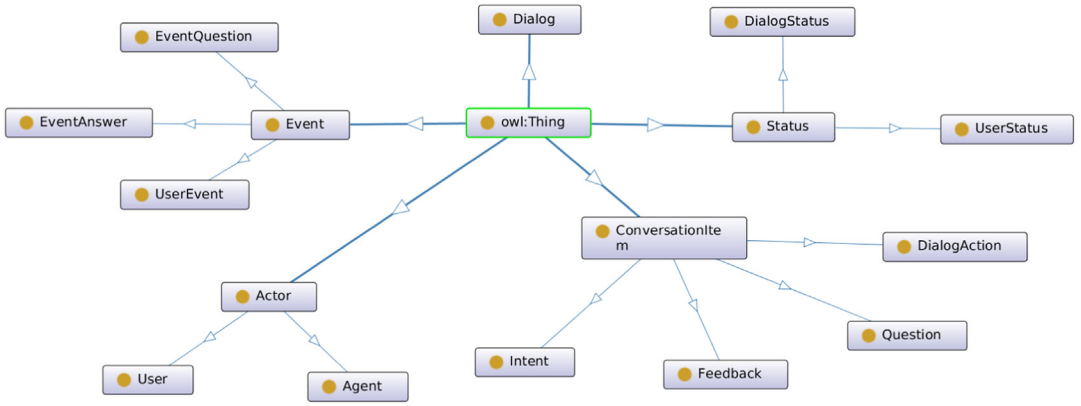

FIGURE 2.1 Overview of Convology.

2.4.1 Dialog

The *Dialog* concept represents a multiturn interaction between a *User* and one or more *Agent*. A new instance of the *Dialog* concept is created when a user starts a conversation with one of the agents available within a specific application. The *hasId* datatype property associated with the *Dialog* instance works as tracker for all the interactions made during a single conversation between a *User* and the involved *Agent*. Furthermore, the value of this property is used at reasoning time for extracting from the knowledge repository only the data related to a single conversation in order to maintain the efficiency of the reasoner suitable for a real-time environment.

2.4.2 Actor

The *Actor* concept defines the different roles that can take part into a conversation. Within Convology, we foresee two main roles represented by the concepts *Agent* and *User*. Instances of the *Agent* concept are conversational agents that interact with users. When Convology is deployed into an application, instances of *Agent* concept represents the different agents involved into the conversations with the users adopting the application. Within the same application (e.g., the conversational agent implemented for the asthma scenario described in Section 2.6), Convology will have a different instance of the *Agent* concept for each *User* even if the application is the same. The rationale behind is that different active conversations may be in different statuses. Hence, for favoring the efficiency of reasoning activity, different instances are created into the ontology. Finally, different instances of the *Agent* concept are associated with different instances of the *Dialog* concept.

The second concept defined in this branch is *User*. Instances of the *User* concept represents the actual users that are dialoguing with the conversational agent. A new instance of the *User* concept is created when a new user starts a conversation within a specific application (e.g., a new user installs the application for monitoring her asthma conditions). An instance of the *User* concept can be associated with different instances of the *Dialog* and

Agent concepts. The reasons for which this does not happen for the *Agent* concept (i.e., an *Agent* instance can be associated with one and only one instance of *User*) is because the focus of Convology is to track and support the conversations from the user perspective. Thus the ontology maintains a single instance of *User* for each deployment of Convology due to the necessity of tracing the whole history of users.

For debug purposes (e.g., to analyze the behavior of the conversational agents for evaluating its effectiveness), it is anyway possible to collect all instances of the *Agent* concept.

2.4.3 ConversationItem

A *ConversationItem* is an entity taking part into a conversation and that allows to represent relevant knowledge for supporting each interaction. Within Convology, we defined four subclasses of the *ConversationItem* concept: *Question*, *Intent*, *Feedback*, and *DialogAction*.

An individual of type *Question* represents a possible question that an instance of type *Agent* can send to a *User*. Instances of *Question* are defined by domain experts together with all the *Intent* individuals that are associated with each *Question* through the *hasRelevantIntent* object property. A *Question* can be associated with also a specific *UserEvent* through the *hasTriggerQuestion* object property.

An *Intent* represents a relevant information, detected within a natural language answer provided by a *User*, that the NLU module is able to recognize and that the reasoner is able to process. Concerning the mention to the NLU module, it is important to clarify that the detection of an *Intent* within a user's answer requires the integration of a NLU module able to classify the content of users' answers with respect to each *Intent* associated with the *Question* sent to a *User*. Hence, one of the prerequisites for deploying Convology into a real-world system is the availability of a module that maps the content of users' answers with the instances of the *Intent* concept defined into the ontology. The possible strategies that can be implemented for supporting such a mapping operation are out of scope of this chapter. An *Intent* is then associated with a *StatusItem* through the *activated* object property: once a specific *Intent* is recognized, a *StatusItem* instance is created into the knowledge repository for supporting the inference of the user's status.

Differently from a *Question* where it is expected that a *User* performs a new interaction and a set of relevant *Intent* are associated with them, a *Feedback* represents a simple sentence that an *Agent* can send to users and for which it does not expect any reply. Feedback are used for closing a conversation as result of the reasoning process or simply for sending single messages to users without requiring any further interaction.

Instances of the *DialogAction* concept describes the next action that an *Agent* individual has to perform. *DialogAction* individuals can be defined by domain experts as consequences of the detection of specific intents or can be generated as result of reasoning activities and associated with a *DialogStatus* instance. Individuals of type *DialogAction* are associated with a *Question* or a *Feedback* individual representing the next message sent to a *User*. Moreover, a *DialogAction* might have the datatype property *waitTime* set, that is, the amount of seconds that the system must wait before sending the *Question* or the *Feedback* to the *User*.

2.4.4 Event

The *Event* concept describes a single event that can occur during a conversation. Within Convology, we identified three kinds of events: *EventQuestion*, *EventAnswer*, and *UserEvent*. Instances of these concepts enable the storage of information within the knowledge repository, trigger the execution of the reasoning process, and allow the retrieval of information for both analysis and debugging purposes.

An *EventQuestion* represents the fact that a *Question* has been submitted to an *Actor*. Here, we do not make distinctions between the actors because, from a general perspective, the model supports scenarios where questions are sent from a *User* to an *Agent*. Instances of this concept are associated with knowledge allowing to identify the timing (*hasTimestamp* datatype property), the *Actor* instance that sent the question (*sentQuestion* object property), and the *Actor* instance that received the question (*receivedQuestion* object property).

On the contrary, the *EventAnswer* concept represents an *Answer* provided by an *Actor*. The timing information associated with individuals of this concept is defined through the *hasTimestamp* datatype property, whereas the sender and the receiver are defined by the *sentAnswer* and *receivedAnswer* object properties.

The last concept of this branch is the *UserEvent* one. A *UserEvent* represents an *Event* associated with a specific user. The purpose of having a specific *UserEvent* concept instead of inferring *UserEvent* objects from the *EventQuestion* and *EventAnswer* individuals is that a *UserEvent* does not refer only to questions and answers but also to other events that can occur. Examples are the presence of one or more *Intent* within users' answer (this kind of knowledge cannot be associated with *EventAnswer* individuals because *Agent* instances do not provide *Intent* within an answer) or information about users' action that are not directly connected with the conversation (the storage of these information is important in case of it is of interest to analyze users' behaviors). The relationship between a *UserEvent* and an *Intent* is instantiated through the *hasRecognizedIntent* object property. Finally, instances of *UserEvent* can trigger the activation of a specific *UserStatus* (explained below) as result of the reasoning process. Triggering events are instantiated through the *hasTriggerQuestion* and *triggers* object properties. The former allows to put in relationship a *UserEvent* with a *Question*. The latter associates a *UserEvent* with a specific *UserStatus*. Both relationships are defined as result of the reasoning process.

2.4.5 Status

The last branch of Convology has the *Status* concept as top-level entity. This branch contains concepts describing the possible statuses of users, through the *UserStatus* and *StatusItem* concepts, or of dialogs, through the *DialogStatus* concept.

Instances of the *UserStatus* concept are defined by the domain experts, and they represent which are the relevant statuses of a *User* that the conversational agent should discover during the execution of a *Dialog*. Let us consider the asthma scenario described in Section 2.6, the aim of the conversational agent is to understand which is the health status of the user. Within this application, the domain experts defined four *UserStatus* based on the gravity of the symptoms that are recognized during the conversation. A *UserStatus* is associated with a set of *UserEvent* that, in turn, are associated with *Intent* individuals.

This path describes which is the list of *Intent* enabling the classification of a *User* with respect to a specific *UserStatus*. This operation is performed by a SPARQL-based reasoner.

A *UserStatus* individual is associated with a set of *StatusItem* individuals representing atomic conditions under which a *UserStatus* can be activated. Generally, not all *StatusItem* has to be activated for inferring, in turn, a *UserStatus*. Different strategies can be applied at reasoning time, but they are out of scope of this chapter.

The third subsumed concept is the *DialogStatus* one. A *DialogStatus* individual provides a snapshot of a specific *Dialog* at a certain time. Entities associated with a *DialogStatus* individual are the *Dialog* which the status refers to, the identifiers of the *User* and of the one or more *Agent* involved into the conversation, and the *DialogAction* that has to be performed as next step. Individuals of type *DialogStatus* are created at reasoning time after the processing of the *Intent* recognized by the system.

2.5 Availability and reusability

Convology is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0[3], and it can be downloaded from the Convology website[4]. The rational behind the CC BY-NC-SA 4.0 is that the Trentino Healthcare Department, that funds the project in which Convology has been developed, was not in favor of releasing this ontology for business purposes. Hence, they force the adoption of this type of license for releasing the ontology. Convology can be downloaded in two different modalities: (1) the conceptual model only, where the user can download a light version of the ontology that does not contain any individual, or (2) the full package, where the ontology is populated with all the individuals we have already modeled for the asthma domain. Convology is constantly updated due to the project activities using the ontology as core component.

The ontology is available also as web service. Detailed instructions are provided on the ontology website. Briefly, the service exposes a set of *informative* methods enabling the access to a JSON representation of the individuals included into the ontology.

The reusability aspect of Convology can be seen from two main perspectives. First, Convology describes a metamodel that can be instantiated from conversational agents into different domains. This opens the possibility of building an ecosystem of knowledge resources describing conversational interactions within many scenarios. Second, Convology enables the construction of innovative smart applications combining both natural language processing and knowledge management capabilities as presented in Section 2.6. Such applications represent innovative solutions within the conversational agents field.

[3] https://creativecommons.org/licenses/by-nc-sa/4.0/.

[4] http://w3id.org/convology.

2.6 Convology in action

As introduced, a real practical scenario based on Convology was the development of *PuffBot*, a multiturn goal-oriented conversational agent supporting patients affected by asthma. The current version of *PuffBot* supports interactions in Italian, but we are in the process of extending it to both English and Chinese languages. In Fig. 2.2, we provide a sample conversation in Italian between *PuffBot* and a user. *PuffBot* is equipped with a NLU module able to classify intents contained within natural language text provided by users. The current list of intents available in *PuffBot* is relatively short (almost 40) and were defined with the collaboration of the domain experts of the Trentino Healthcare Department. Within this list, there are also defined 12 intents referring to the **OnBoarding** part consisting in a set of *Question* submitted for building a preliminary of the user profile (e.g., the name, the city where she lives, sports practiced, etc.) that is stored into the knowledge repository and used as contextual information at reasoning time. The main aim of *PuffBot* is to perform a real-time inference of *UserStatus* in order to monitor users' health conditions and to suggest the most effective action to take for solving undesired situations.

During the design phase, we decided to create a hierarchy of intents, each single intent belongs to a set of related intents. For instance, we have defined different types of intents related to the cough (e.g., cough frequency, last episode) and other intents related to the recently medical examinations done or breath situation. To handle different steps of

FIGURE 2.2 This figure illustrates an example of an entire conversation with *PuffBot*. The two screenshots on left show the **OnBoarding** phase where we delineate the user profile. The third one instead is the real conversation scenario where we want to infer the *UserStatus* through a series of questions. The last message contains the overall resume with the advice made by the reasoner.

conversations, all together we have exploited the possibility of creating several instances of *Dialog*, each one with its own *DialogStatus* identified by Convology with a unique identifier.

The conversation can be triggered both by the user and the agent. When the agent receives a trigger from the outside (e.g., a humidity changing in the air was detected), it can ask specific questions to the user in order to monitor his status. Anyway the user can start the conversation by saying something and so by triggering an *UserEvent* that has to be related to one of the defined intents.

Each time *PuffBot* recognizes a relevant intent (i.e., an intent modeled within the knowledge base), and it triggers the reasoner that is in charge of inferring the current user's status and to generate the next *DialogAction* to take. For instance, a possible *DialogAction* can be a further question needed for understanding the *UserStatus* with higher accuracy. Once the application classifies the *UserStatus* with a certain accuracy[5], the reasoner triggers the dispatch of an advice to the user containing a summary of the information that has been acquired and inferred through the use of Convology. Generally, this advice is an instance of the *Feedback* concept.

Fig. 2.3 presents an exemplification about how the reasoning process works. On the top-left part of the picture, we report a piece of the conversation between the user and *PuffBot*. Red circles highlight relevant messages provided by users that are transformed into *UserEvent* individuals (i.e., the blue blocks in Fig. 2.3). At this point the NLU module is invoked for analyzing the natural language text provided by the user and it returns the set of detected *Intent*. For each *Intent*, the *hasRelevantIntent* object property is instantiated (i.e., the green arrows in Fig. 2.3) in order to associate each *UserEvent* individual with the related *Intent* (i.e., the white block in Fig. 2.3). The right part of Fig. 2.3 shows three instances of the *UserStatus* concepts, namely *LowRisk*, *MediumRisk*, and *HighRisk*. These individuals are defined by domain experts and they represent the risk level of a *User* of having a strong asthma event in the short period. Each status is associated with several symptoms that are instances of the *StatusItem* concept. Within the knowledge base, the relationships between an *Intent* and a *StatusItem* are defined through the *activates* object property (i.e., the red arrows). Hence, the detection of specific *Intent* triggers the activation of specific *StatusItem* individuals. At this point, the SPARQL-based reasoner starts and try to infer which is the most probable status in which the user is and, in case of an undecided classification, it generated the proper individuals for triggering the continuation of the conversation (i.e., *DialogAction* individuals).

2.6.1 Other scenarios

Besides the description of the *PuffBot* application, Convology is going to be deployed in more complex scenarios. Below, we mention two of them always related to the healthcare domain, indeed, as explained in Section 2.7, currently, the sustainability of Convology is strictly connected with activities jointly done with the Trentino Healthcare Department. The first one concerns the promotion of adopting healthy lifestyle. Here, a conversational agent is used for acquiring information about consumed food and performed physical activities by means of natural language chats with users. With respect to the *PuffBot* application, the number of possible *Intent* and *UserStatus* dramatically increases due to the high number of relevant entity that the system has to recognize (i.e., one *Intent* for each recipe

[5] The strategies implemented for classifying users within different statuses are out of scope of this chapter.

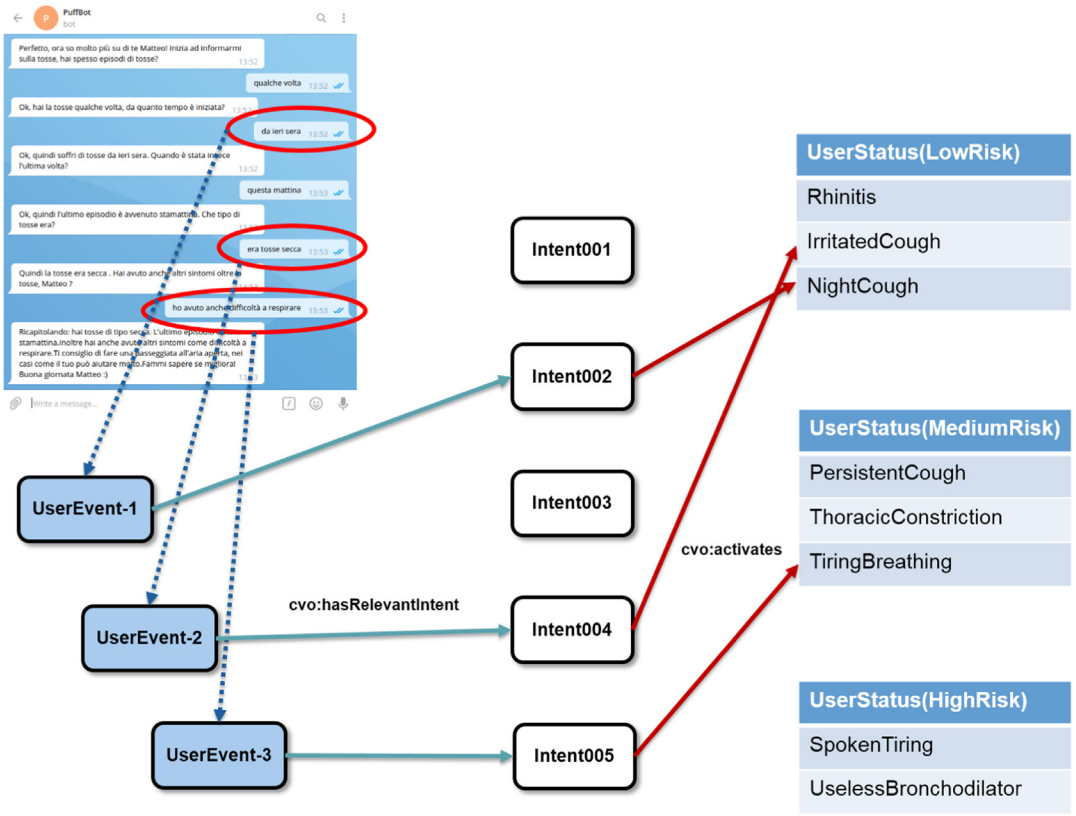

FIGURE 2.3 Exemplification of the reasoning process.

and physical activity). The second scenario relates to support users affected by diabetes concerning its self-management of the disease. One of the most common issue in self-managing chronic disease is given by psychological barriers avoiding users in performing self-monitoring actions (e.g., measuring glycemia value). Convology will be deployed into an application used for knowing which are the barriers affecting each user. With respect to the first scenario and to the *PuffBot* application, the main challenge that will be addressed by the domain experts is the definition of all relevant *Intent* associated with each barrier that has to be detected. This modeling task will require a strong interaction between psychologists and linguistics in order to identify all natural language expressions that can be linked with each barrier.

2.7 Resource sustainability and maintenance

As mentioned in the previous section, the presented ontology is the result of a collaborative work between several experts. While, on the one hand, this collaboration led to the

development of an effective and useful ontology; on the other hand, the sustainability and the maintenance of the produced artifact represent a criticality.

Concerning the sustainability, this ontology has been developed in the context of the *PuffBot* project. The goal of this research project is to provide the first prototype of conversational agent relying on the use of a knowledge base in order to support a multistep interaction with users. This project, recently started within FBK, is part of the "Trentino Salute 4.0" framework promoted by the Trentino's local government with the aim of providing smart applications (e.g., intelligent chat-bots) to citizens for supporting them under different perspectives (e.g., monitoring of chronic diseases, promoting healthy lifestyles, etc.). One of the goals of this framework is to promote the integration of artificial intelligence solutions into digital health platforms with the long-term goal of improving the life quality of citizens. The presented ontology is part of the core technologies used in this framework. The overall sustainability plan for the continuous update and expansion of the Convology ontology is granted by this framework and by the projects mentioned in Section 2.6.

The maintenance aspect is managed by the infrastructure available within FBK from both the hardware and software perspectives. In particular, we enable the remote collaboration between experts thanks to the use of the MoKi (Dragoni et al., 2014) tool (details about the tool are out of the scope of this chapter). Here, it is important only to remark that this tool implements the support for the collaborative editing of ontologies by providing different views based on the kind of experts (domain expert, language expert, ontology engineer, etc.) that has to carry out changes to the ontology.

The canonical citation for Convology is "Dragoni M., Rizzo G., Senese M.A., Convology: an Ontology For Conversational Agents (2019). http://w3id.org/convology".[6]

2.8 Conclusions and future work

In this chapter, we presented Convology: a top-level ontology for representing conversational scenarios with the aim of supporting the building of conversational agents able to provide effective interactions with users. The knowledge modeled within Convology derives from the analysis of knowledge engineers with competences in NLU, and it has been thought for providing a metamodel able to ease the development of smart applications. We described the process we followed to build the ontology and which information we included. Then, we presented how the ontology can be utilized and we introduced the projects and use cases that currently integrate and use Convology.

Future research activities will focus on the integration of our model within the projects we mentioned in Section 2.7 with the aim of verifying the correctness and completeness of Convology and to further improve the model. Furthermore, our intent is to analyze if also the Convology TBox can be opened to domain experts in order to provide a more flexible tool for describing specific domains. Finally, we aim to integrate Convology within mindfulness applications that, from the conversational perspective, are very complex to manage and it would be a stressful test-bed for the proposed model.

[6] DOI of the ontology file will be provided in case of acceptance in order to include possible refinements suggested by Reviewers.

References

Dragoni, M., Bosca, A., Casu, M., Rexha, A., 2014. Modeling, managing, exposing, and linking ontologies with a wiki-based tool. In: LREC, pp. 1668–1675.

Fernández-López, M., Gómez-Pórez, A., Juristo, N., 1997. Methontology: from ontological art towards ontological engineering. In: Proc. Symposium on Ontological Engineering of AAAI.

Flycht-Eriksson, A., Jönsson, A., 2003. Some empirical findings on dialogue management and domain ontologies in dialogue systems—implications from an evaluation of birdquest. In: Proceedings of the Fourth SIGdial Workshop on Discourse and Dialogue, 2003, pp. 158–167. <https://www.aclweb.org/anthology/W03-2113>.

Franklin, S., Graesser, A., 1997. Is it an agent, or just a program?: a taxonomy for autonomous agents. In: Müller, J.P., Wooldridge, M.J., Jennings, N.R. (Eds.), Intelligent Agents III Agent Theories, Architectures, and Languages. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 21–35.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L., 2002. Sweetening ontologies with dolce. In: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, Springer-Verlag, 2002, pp. 166–181. <http://dl.acm.org/citation.cfm?id=645362.650863>.

Ginzburg, J., 2012. A semantic ontology for dialogue. The Interactive Stance. Oxford University Press, Oxford.

Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A., Presutti, V. (Eds.), 2016. Ontology engineering with ontology design patterns—foundations and applications. *Studies on the Semantic* Web, IOS Press, Vol. 35.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Torralba, A., Urtasun, R., et al., 2015. Skip-thought vectors, CoRR abs/1506.06726. <http://arxiv.org/abs/1506.06726>. arXiv:1506.06726.

Mensio M., Rizzo, G., Morisio, M., (2018) Multi-turn qa: a rnn contextual approach to intent classification for goal-oriented systems. In: Companion Proceedings of the The Web Conference 2018, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, pp. 1075–1080. Available from: https://doi.org/10.1145/3184558.3191539.

Milward, D., 2004. Ontologies and the structure of dialogue. In: Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue (Catalog), 2004, 69–77.

Pinto H.S., Staab, S., Tempich, C., 2004. DILIGENT: towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In: de Mántaras, R.L., Saitta, L. (Eds.), Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22–27, 2004, IOS Press, pp. 393–397.

Ramesh, K., Ravishankaran, S., Joshi, A., Chandrasekaran, K., 2017. A survey of design techniques for conversational agents. In: Kaushik, S., Gupta, D., Kharb, L., Chahal, D. (Eds.), Information, Communication and Computing Technology, Springer Singapore, Singapore, pp. 336–350.

Suárez-Figueroa, M.C., 2012. NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse (PhD thesis), Technical University of Madrid, 2012. <http://d-nb.info/1029370028>.

Weizenbaum, J., 1966. Eliza—a computer program for the study of natural language communication between man and machine. Commun. ACM 9, 36–45. Available from: https://doi.org/10.1145/365153.365168.