Plan and Ontology-Based Dialogue Policies for Healthcare

Milene Santos Teixeira[1], Mauro Dragoni[2]

[1]*Ulm University, Germany*
[2]*Fondazione Bruno Kessler, Trento, Italy*

Abstract
Dialogue systems for healthcare have faced slower progress when compared to traditional conversational domains. This delay can be blamed on healthcare constraints that require these systems to be interpretable and avoid risks to their target users. In this work, we discuss our approach for automating the generation of the dialogue manager, the component in charge of tracking the dialogue state and deciding the agent's next move. By integrating automated planning and a conversational ontology, we enable the generation (and update) of dialogue policies that address healthcare needs. Here, we put our emphasis on the aspects that demonstrate the benefit of dynamically planning such policies. While a study with experts shows their appreciation for a resulting agent, a synthetic evaluation reveals the scalability of the approach for real-sized healthcare dialogues and its suitability for real-time plan generation.

Keywords
automated planning, ontology, dialogue manager, healthcare

1. Introduction

A dialogue manager is the component of a goal-oriented dialogue system that is in charge of (i) tracking the current state of the dialogue and (ii) deciding the next action to be taken by the agent [1]. Within the healthcare domain, this decision is not trivial since, among other health-related needs, a wrong choice might lead the agent to waste interactions and delay the detection of possible emergencies. In fact, dialogue systems that address healthcare are required not only to be fast in their reasoning but also to be interpretable. Interpretability is vital not only to avoid risks to the target users of the system, but also to gain their trust [2]. As such, it is common for these systems to still rely on a hand-crafted dialogue manager, enabling the dialogue author to maintain control of the policy. As a consequence, health-related dialogue systems are limited and lag behind traditional conversational domains [3].

While techniques such as reinforcement learning can be highly efficient at learning policies from training data, they lead to a loss of some of the control over the outcome of the dialogue [2]. Automated planning is therefore a good alternative for managing predictable dialogue policies. Although planning is still overlooked when it comes to dialogue management, it has been reported as a powerful technique, capable of handling efficient policies [1]. Given the domain-knowledge specification, planning can automate the generation of huge policies that would be prone to errors or even unfeasible for a human actor to generate. However, specifying the

Planning and Ontology Workshop (PLATO), ICAPS 2023, July 09-10, 2023, Prague, CZ
milene.tsi@gmail.com (M. S. Teixeira); dragoni@fbk.eu (M. Dragoni)

components of the dialogue as a planning problem is a challenging task. Knowledge of dialogue modeling and automated planning is required. As the latter is uncommon among dialogue authors, it limits the adoption of plan-based approaches. Hence, to build efficient policies, the integration of AI planning with further mechanisms to formalize the knowledge of a specific domain is expected to bridge this gap.

This work presents an approach that integrates AI planning and a conversational ontology for the automated generation (and update) of dialogue managers for the health domain. The focus of this paper is given mainly to the plan-based aspects of the approach since we aim at demonstrating the feasibility of employing planning for real-world (and real-sized) health dialogues that require fast responses. In this approach, ontology has as its main roles aiding (i) the specification of the planning problem and (ii) the management of dialogue interactions. The proposed approach covers *information-seeking* dialogues [4], a type of dialogue that focuses on retrieving information with the purpose of revealing a position (classification). However, it is important to note that the layers that interact with the end-user, i.e. the natural language understanding (NLU) and generation (NLG) components are out of our scope and we refer the reader to [5] and [6] for surveys on those topics, respectively. Therefore, when referring to the information acquisition process, we refer to the dialogue management component, which is in charge of tracking the dialogue state and selecting the system's next action.

In this work, action selection is further aided by the agent-based framework introduced in [7]. The framework identifies the most useful piece of (missing) information to be retrieved at each step of the dialogue interaction. Here, the contribution of the framework is twofold: (i) it supports the agent in the action selection task by combining components that allow the complex reasoning on health-related factors (e.g. class' priority, slot's weight, co-occurrences) and (ii) it provides a strategy for classification, i.e. it identifies when the information retrieved is enough to output a conclusion, according to domain-specific guidelines. These two tasks deploy some of the planning actions specified in this work and they are exploited by the dialogue agent at execution time.

The remainder of this paper is organized as follows: the next section discusses related work that addresses ontologies and planning for dialogue management. Next, we summarize Convology, the ontology exploited in our dialogue manager. The following section details our approach, by keeping the focus on its plan-based aspects. Additional details on the handling of state tracking, mixed-initiative, and replanning are then provided. Next, we describe an early-stage user study and a synthetic evaluation. Finally, the paper is concluded, highlighting future contributions.

2. Related Work

In [3], a review of conversational agents for healthcare is presented. The authors reveal the slow progress of such systems. Recent literature on plan-based dialogue management was surveyed in [1]. In this survey, the authors identify the main contributions as well as open challenges to the use of planning as a dialogue manager. Next, we highlight a few works that approximate our contribution.

By also exploiting fully observable non-deterministic (FOND) planning, the work of [8] tackled the model acquisition problem concerning plan-based dialogues. By using an interface for declarative dialogue design, dialogue authors are in control of the behavior of the agent, being able to edit it as desired. However, this approach revealed significant confusion among non-experienced dialogue authors. Therefore, it was later extended in [9] to integrate explainable AI planning techniques with the aim of improving explainability during the model acquisition process. We simplify the model acquisition process by mapping the knowledge within the conversational ontology into the planning specification.

In [10], the authors rely on a domain ontology to provide information on the topic of the dialogue. While a task manager specifies a list of domain-specific tasks, a planner determines the execution ordering of these tasks. Lee et al. [11] also rely on a domain ontology for the generation of a dialog support system. However, this ontology is used by the planner to learn which is the goal state of a detected situation and how the components modeled in the ontology affect the environment so an appropriate plan can be generated to change this situation.

The work of Behnke et al. [12] integrates the user through a dialogue on a plan generation for a companion system. By relying on an ontology as a common knowledge source, separated models are automatically generated (and also extended) for the plan and the dialogue. In this process, ontology and planning aspects are translated through a mapping defined by the authors. A similar strategy is adopted by [13], which also addresses companion systems. This work exploits a conceptual model to provide domain-specific knowledge and a planning model to address procedural knowledge. The planning model is automatically generated by exploiting the most generic concepts within the ontology.

The ontologies used in these works represent the knowledge of the domain of the dialogue, but they do not formalize or represent the dialogue interactions themselves. In these works, dialogue interactions have been modeled and handled by additional components. An exception is the work of Bickmore et al. [14], which automates the generation of dialogue systems that focus on promoting health behavior change. In addition to domain knowledge on behavioral medicine, the ontology includes the description of a few components of the dialogue (e.g. *DialogueAction*), and a task model exploits the knowledge from the ontology to describe what steps should be taken by the system to achieve a goal. Although the authors have addressed reusability within their model, the focus of this approach is very restricted to the behavioral change domain. Our work, instead, combines an ontology for representing the conversational domain with planning techniques to support the automated generation of dialogue managers for different healthcare-wise domains.

3. The Conversational Ontology

Our work relies on Convology[1], a top-level ontology that models the goal-oriented conversational scenario (Figure 1). Its aim is twofold: (i) to semantically model and describe the concepts of the dialogue and (ii) to aid the understanding and management of the entire dialogue workflow. Convology can aid both the development of conversational agents and real-time dialogue interactions between a user and an agent. An application of Convology addressing an early version of our approach was described in [15].

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

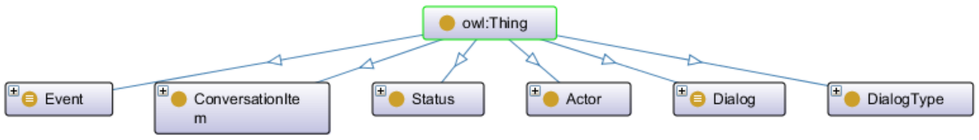


Figure 1: Overview of Convology.

The terminological part or TBox of the ontology addresses concepts that are common to any goal-oriented dialogue. Next, we briefly list the concepts that are most interesting for this work.

The concept *Dialog* represents the multi-turn interaction and connects *Subdialog* instances. These instances represent smaller interactions that are identified according to a *DialogType*. This work focuses on *InformationSeeking* dialogues, as previously mentioned. *Event* describes a single event that can be performed by an *Actor* (i.e. *User* or *Agent*) during a conversation, i.e. every action performed by either the *Agent* or the *User* is instantiated as an *Event*.

Within the concept *ConversationItem*, that identifies relevant knowledge involved in each interaction, we have: (i) *Slot*, representing pieces of information to be filled during the session (e.g. *symptom_x*); (ii) *RelevantInformation*, representing the value given to a *Slot* (e.g. *YesValue*, *NoValue*); (iii) *DialogAction*, representing actions performed by the *Agent*, i.e. the next message sent to a *User* (e.g. *request_symptom_x*). This class subsumes the concepts: *Feedback*, *ClosedQuestion*, and *OpenQuestion*, which represent the type of actions that can be addressed by the *Agent*; (iv) *Intent*, representing a piece of relevant information that is recognized by the NLU module from a natural language input provided by a *User* (e.g. *inform_symptom_x*).

The *Status* concept subsumes (i) the *UserStatus* concept, whose instances represent the possible states of a *User* that can be discovered by means of a conversation (i.e. the classification achieved) and (ii) the *DialogStatus* concept, whose instances provide a snapshot of a specific *Subdialog* at a certain time. The *Status* concept can be associated with the *Dialog* instance to indicate the possible dialogue goals.

The assertional part of the ontology (ABox) is reserved for the domain-specific knowledge that must be provided when the ontology is integrated into a conversational application. While some instances must be provided by the dialogue author (e.g. *Slot, UserStatus*), others can be generated as a result of the dialogue interactions, depending on the domain requirements (e.g. *DialogStatus*).

4. An Ontology and Plan-Based Dialogue Manager

Our approach (Figure 2) for the automated generation (and update) of dialogue managers is composed of (i) a model acquisition component, (ii) a planning module, and (iii) an execution module. As the focus of this paper is on the plan-related aspects of the approach (i.e. the planning module), we only briefly summarize the role of the other two modules. The integration with the ontology is discussed accordingly.

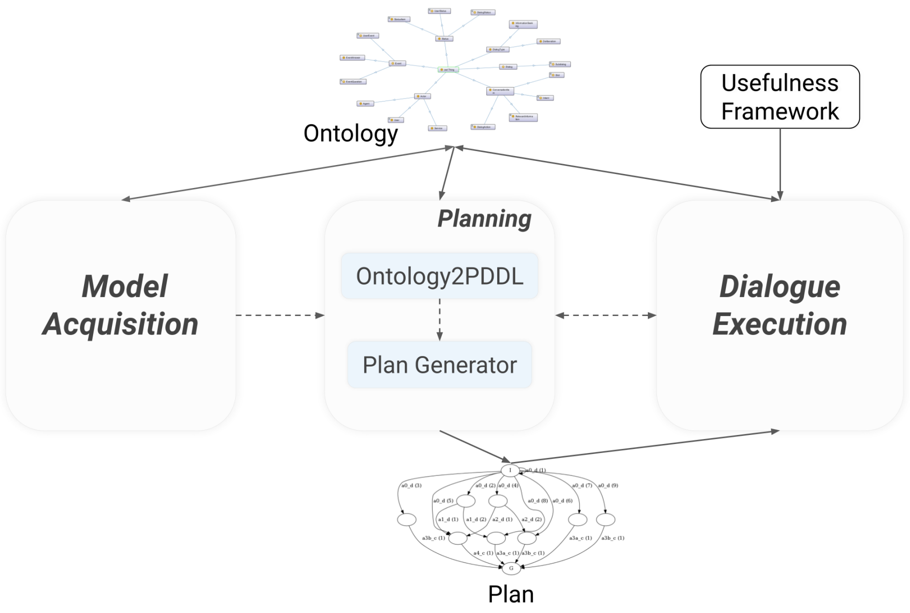


Figure 2: Overall picture of the modules composing the plan and ontology-based dialogue manager.

4.1. Model Acquisition Module

The first unit is dedicated to the acquisition of knowledge on the topic of the dialogue system. At this stage, all domain-specific knowledge that is relevant to the dialogue session must be populated in the ontology ABox. Instances for concepts such as *Subdialog*, *UserStatus*, and *Slot* must be provided by a dialogue author. For example, for *subdialog_1* the slots *slot1*, *slot2*, and *slot3* must be acquired to understand whether *class1* can be classified. From *Slot* instances, additional instances are inferred such as *Intent* instances, that represent information that can be interpreted from *user actions* (e.g. *inform_slot1*) and *DialogAction* instances, representing *system actions* (e.g. *request_slot1*).

4.2. Planning Module

The role of the *planning* module is to specify the components of the dialogue in a way that it can be compiled by a state-of-the-art planner, with the aim of automatically generating the dialogue policy (i.e. plan) to be integrated into the dialogue system. In this work, the planning specification, which is given in the Planning Domain Definition Language (PDDL), is derived from the knowledge within the conversational ontology. This specification relies on correspondences that can be observed among both techniques, for example, *DialogAction* in the ontology can be represented as planning actions. This way, to keep an interface between both, the specification of the planning problem replicates the vocabulary used in the ontology. However, while some aspects within the planning specification are mapped directly from concepts or instances in the ontology, others correspond to modeling decisions that are necessary to obtain the expected plan (e.g. handling deadends). Besides obtaining a structured policy that selects the most appropriate actions towards the dialogue goal, the mapping to PDDL will easily reflect in the plan any possible update made in the domain-specific knowledge represented in the ontology ABox (e.g. add or delete slots).

FOND planning is the type adopted in this work. In FOND problems, operators may have a set of different possible effects, indicating that one of these effects will be true at execution time. Non-determinism is suitable to the kind of problem addressed in this work since, although a set of possibilities exists, it is not possible to know the exact outcome (effects) of most dialogue actions before their execution. By representing dialogue actions as non-deterministic, the plan can reflect the various paths that the dialogue might take. Meanwhile, as state tracking in this work does not require high complexity (if compared to a spoken dialogue system), *full observability* is obtained by adopting a threshold to the confidence score that is attached to the *Intents* received from the NLU output. Finally, within FOND planning, we address *strong-cyclic planning* (SCP) [16]. Through this kind of solution, it is possible to guarantee that a path to reach the dialogue goal (task completion) exists, repeating an action when necessary information is still missing (e.g. clarification questions can be made).

4.2.1. Planning Domain Specification

In this work, planning actions represent dialogue actions and they are classified into two types: DIALOGUE and SERVICE, which were borrowed from [17]. Actions of type DIALOGUE originated from *DialogAction* instances in the ontology. These actions result in interactions with the end-user, such as questions or feedback that are converted to a natural language output. SERVICE actions, on the other hand, are specified to manage aspects of the dialogue session that occur in the background (e.g. setting a constraint to avoid a dead-end).

In order to address the needs of the health domain in the information retrieval process, the specification of the main actions within the planning domain integrates the usefulness framework previously described. Since the usefulness value of a slot varies by state according to the information retrieved, the integration of this framework into the planning domain relies on abstraction. That is, the planning domain includes abstract actions that instruct the agent to *search* for and *request* the most useful slot, anticipating the paths for the different alternatives. Since goal achievement also depends on the information retrieved, the corresponding actions (named *report*) also rely on abstraction. This way, it is only during execution time that the agent deploys the usefulness framework and determines the most useful slot or whether a classification was reached. This strategy makes it possible to separate domain knowledge reasoning from dialogue management, supporting the flexibility and portability of the approach.

In the remainder of this section, we focus on the description of these main actions [2].

search_most_useful_slot A SERVICE action named *search_most_useful_slot* is specified to support the information retrieval process. This action states that a search for the most useful slot is the next step to be conducted by the agent. The precondition for this action is the predicate *search_most_useful*. This action has a non-deterministic effect, which implies that any of the listed slots could be the most useful one. However, the actual value can be determined only upon action execution. A simplified example of this action, including three slots, is given below:

[2]Full samples of planning specifications are available at https://osf.io/t48qm/?view_only=98403c3d7ce34f71bacdf6d180764c96

```
:action search_most_useful_slot
:precondition search_most_useful
:effect (oneof
  (is_most_useful slot1)
  (is_most_useful slot2)
  (is_most_useful slot3) )
```

request All *ClosedQuestion* (subconcepts of *DialogAction*) instances related to a *Subdialog* instance are translated to a *request* action in the planning domain. Request actions are of type DIALOGUE as they will be translated to a question that is made to the end-user. *Request* actions specify two main preconditions: (i) the associated slot is empty and (ii) the slot is the most useful. The effect of a *request* action, on the other hand, is non-deterministic since its execution may result in (i) the identification that a value was received for the slot (i.e. it is no longer empty) and the possible classifications[3] that might have been achieved as a consequence of filling this information; or (ii) a *fallback_intent* is reported, meaning that the agent did not understand the user's answer. A simplified example of a *request* action is given below:

```
:action request_slot1
:precondition (and (is_most_useful slot1) (not(has_value slot1)))
:effect (oneof
  (and (has_value slot1) (possible_classification_us class1))
  (fallback_intent) )
```

report All *UserStatus* instances related to a *Subdialog* instance are translated to a *report* action in the planning domain. These actions are used for the classification of the *UserStatus* (goal achievement), after retrieving the expected information. Like the actions for information retrieval (i.e. *request*), the specification of *report* actions is based on the usefulness framework. Therefore, instead of a predefined set of *Slot* instances that classify a given *UserStatus*, different combinations of slot values might lead to a classification. To avoid specifying every possible combination as preconditions of report actions, the modeling strategy adopted in this work abstracts this information with the predicate *possible_classification_us*. This predicate works as a flag indicating that the agent must check if a classification is achieved for the corresponding *UserStatus*. Since every new information received by the agent might lead to a classification, this flag is raised as an effect of a *request* action, as previously described. This way, *report* actions have a non-deterministic effect and they can be considered a mix of SERVICE and DIALOGUE actions. The former is given when no classification is reached and, therefore, the flag *search_most_useful* enables the continuation of the information retrieval process. The latter is a consequence of a classification, which should result in output to the end-user.

A simplified example of a classification action for a *UserStatus* instance (*class1*) is shown next:

```
    :action report_classification_class1
    :precondition (possible_classification_us class1)
    :effect (oneof (search_most_useful) (GOAL_ACHIEVED) )
```

[3]The precondition for a *report* action is now satisfied.

4.2.2. Planning Problem Specification

The *problem* instance, which is based on the domain instance, is also specified according to the knowledge available in the ontology. This instance defines (i) the *initial* state, (ii) the *goal* state, and (iii) the set of *objects* of the modeled world.

The *initial state* specifies the information currently available; i.e. the information regarding the *Slot* instances. The *goal state* is abstracted with the predicate *GOAL_ACHIEVED*. Since different combinations of parameters might reach the goal, it is enough to specify this predicate as an effect of actions that lead to the goal. In some cases, when goal-achievement is a possibility but it is not guaranteed upon action execution, like *report* actions, this predicate is associated with a non-deterministic effect.

Finally, the finite list of *objects* represents the pieces of information that are available within the problem. This list is obtained from the instances of three classes that are mapped to planning types in the planning domain: *RelevantInformation*, *Slot*, and *UserStatus*. Consequently, each object is associated with its type. If replanning is required at some point in the dialogue, it is enough to update the initial state with updated values for these objects.

4.2.3. Plan Generator

The resulting PDDL specifications are compiled by an automated planner. An efficient state-of-the-art planner for generating strong-cyclic FOND plans is the Planner for Relevant Policies (PRP)[4] [18]. In this work, PRP is exploited to generate plans that correspond to the dialogue policy to be integrated into the dialogue system.

Figure 3 illustrates a short graph of a resulting plan for the information retrieval process of a single slot. While each node represents a dialogue state, the edges represent actions applied to that state[5]. Note that the policy starts by searching for the most useful slot and, next, a request action follows. The request action leads either to (i) a fallback question that leads to the goal state, aiming to force a replanning operation, or (ii) to a state where the requested information was acquired. When the information is acquired, the policy checks for a classification (report action). At this point, the dialogue goal might be achieved. As this is a short example and no further information can be acquired, the other outcome of the report action is to report a default classification, i.e. the classification achieved when no other is possible. However, in a domain with further slots, the next action would correspond to a new search for the most useful slot. Note that this resulting policy can be analyzed by the dialogue author, who is in charge of judging whether this is the agent's expected behavior.

4.3. Execution Module

The execution module is part of the agent's online reasoning process. Its role is to interpret the policy generated in the previous stage, using it to manage the ongoing dialogue session. That is, based on the planned policy, the executor identifies the new dialogue state and learns which action should be executed next. The execution module integrates the natural language layers

[4]https://github.com/QuMuLab/planner-for-relevant-policies

[5]Note that in FOND planning a single action might result in different states (e.g. *request_slot1*).

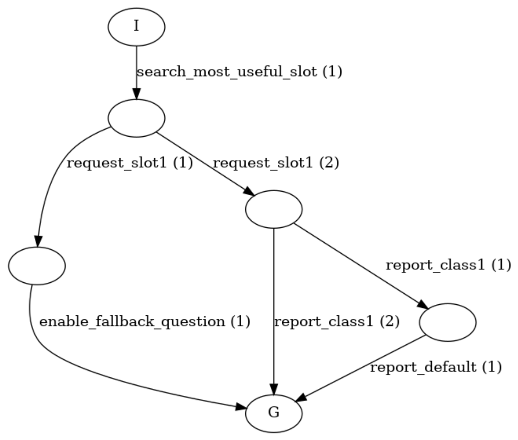


Figure 3: Sample of resulting policy to retrieve one slot.

that are responsible for interacting with the end-user by (i) interpreting the user's input and (ii) translating the system's response to a human-understandable output. This module integrates (i) Convology as a knowledge base for the execution algorithm and (ii) the usefulness framework to reason on the most useful information to be asked and support classification.

5. Further Aspects on Dialogue Management

In this work, the resulting plan corresponds to the dialogue policy that dictates the behavior of the dialogue agent. Every action executed by the agent must be informed by the dialogue policy, that is, it must be planned in advance. At the beginning of each sub-dialogue, an initial dialogue policy is generated by the planner according to the information available in the ontology at that stage. This policy is dynamically replanned during the dialogue session to adapt to updates in the available knowledge. Both replanning operations and sub-dialogue switches are transparent to the end user. The remainder of this section discusses how planning contributes to some additional aspects regarding the management of a dialogue.

Dialogue State Tracking: non-deterministic actions foresee a set of different states that can be reached after their execution. Therefore, the FOND plan anticipates all states that are possible within a sub-dialogue. However, it is only during dialogue execution that it is possible to determine what changed in the real world as a consequence of the execution of the last planned action. This change is what takes the dialogue to a new state. The definition of the possible effects of an action supports state recognition since the agent has a direction on which variables might change upon execution. For example, to recognize the new state after executing the planning action named '*search_most_useful_slot*', the agent can iterate through the action possible outcomes and query the corresponding instances in the ontology. Consequently, the agent identifies which *Slot* was set as the most useful at the current stage. Given this information, the planned action for this state can be executed. Through reasoning, the dialogue history available in the ontology can also support the interpretation of the user input and consequent state recognition. However, this aspect will be investigated in future work.

Mixed-Initiative Dialogues: to keep the dialogue system flexible, our approach supports *mixed-initiative*, a research problem frequently discussed by the dialogue community [19]. The planned dialogue keeps predominantly a system-initiated flow, where the dialogue actions executed by the agent anticipate which information should be included in the intents that are expected in the next user input. However, the user may input some non-expected information at any moment. Addressing user-initiated actions adds extra complexity to the dialogue manager since it is difficult to predict what will be the user input. In fact, considering that the same sort of alternatives may appear in several places, it is not reasonable to anticipate all possible intents that may occur in every dialogue turn. Although this approach deals with non-deterministic actions, adopting such a strategy would lead to a scalability problem in the planning solution. This way, we have adopted replanning as the main strategy to handle mixed-initiative.

Replanning: replanning operations are necessary to adjust the policy to unexpected states. An example is when instead of replying to a previous system's question, the user mentions unrelated information. Before replanning, the *initial state* must be updated in the *planning problem* to match the current dialogue state. Only then, is the automated planner called to generate an updated policy. The executor then uses this policy to give continuity to the ongoing dialogue session. Depending on what triggered the replanning operation, corrective actions must be included in the updated plan. Corrective actions are additional actions that are included to fix what went wrong in the previous plan. Since predicting everything that can go wrong during the session might lead to scalability problems, these actions are an alternative that makes it possible to continue the dialogue flow upon errors. This work includes two corrective actions: (i) the *feedback_unexpected_intent* action acknowledges that the system has processed the unexpected information before continuing the information retrieval, and the (ii) *enable_fallback_question* action (Figure 3) enables the agent to provide a *Feedback* on what went wrong and invite the user to try to repeat the last information in a different way.

6. Evaluation

This section first reports on a synthetic evaluation of the approach's scalability, as well as its applicability in real-time scenarios, given replanning operations. These aspects were evaluated since real-time efficiency is crucial in a health-wise dialogue that needs to be responsive to address emergencies. Next, we briefly describe a user study involving health experts.

6.1. Synthetic Evaluation

Since the synthetic evaluation covers the dialogue manager only, integration with the natural language components was not required for this study. To conduct this evaluation, the ontology was first populated with data related to the original asthma scenario, which is based on a medical guideline. This scenario includes a set of 10 slots (symptoms) and 4 possible classification statuses. This scenario addresses realistic-sized health domains. However, to evaluate the scalability of the planning problem to different specification sizes, the scenario was expanded with synthetic data, implementing different numbers of slots. The number of classification alternatives, instead, was kept always the same. PRP planner was used to generating the policies. For the different

Table 1
Time required to generate policies according to the number of slots, specification and solution sizes

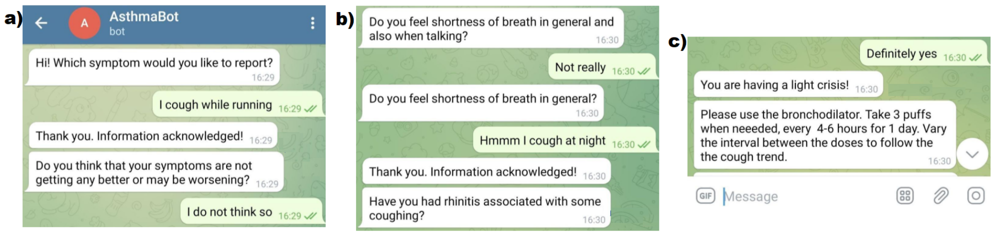
Number of slots	Specification size	Solution size	Time to generate (secs)
10	35	51214	0.38
12	39	238340	1.92
13	42	509695	4.26
14	-	-	-

problem sizes, we analyzed the time required to generate a new policy, which is a sum of the time taken to translate the ontology to PDDL and the search time reported by the planner. The scalability of the solution size (in terms of nodes and edges) with respect to the specification size (in terms of variables and actions) has also been analyzed. As can be seen in Table 1, the solution size grew exponentially in the specification size. This aspect is common when handling uncertainty within AI planning [16]. To our problem, this solution reinforces the contribution of using an automated technique to generate such complex policies, which are far beyond what could be manually done. Moreover, we can observe its benefit when it is necessary to update any domain variable since small changes affect large portions of the final solution.

Our results show the feasibility of computing plans with specification sizes up to 39, which, in our example, corresponded to 12 slots and 4 classification groups that resulted in 23 actions. The next specification size, i.e. size 42 (13 slots and 4 classes), is already beyond the time range accepted by experts (i.e. 2 seconds). Meanwhile, the planner fails to compile problems including 14 slots or more. These results highlight the contribution of automated strategies to generate huge dialogue policies. Besides being costly, such a specification, if manually built, is prone to errors. However, we recognize that some efficiency is compromised with the aim of delivering a reusable approach. Although larger policies could be implemented through sub-dialogues, optimization is one of our interests for future work. An alternative to deal with the limitation on the specification size is through the implementation of sub-dialogues. That is, huge scenarios that present a huge number of slots, for example, can be divided into smaller problems addressed by different instances of an information-seeking *Subdialog* in the ontology. This way, the planner will address them separately, while covering the whole domain. Moreover, the specification of an open initial question (e.g. "*How are you feeling?*"), that can be represented as an *OpenQuestion* in the ontology, opens up the possibility that the end-user will provide some relevant information at this early step. This strategy enables the reduction of the search space for the replanning operation that follows since the information received provides some direction on the possible classification.

6.2. Expert's Evaluation

A dialogue manager covering the original asthma scenario was deployed into an agent that integrated the additional natural language components. The agent was capable of interacting with the end-user to retrieve asthma symptoms with the purpose of identifying what type of crisis the patient might be experiencing. A recommendation was given accordingly (Figure 4). For this agent, an early evaluation including four physicians was conducted. Three physicians


a)
AsthmaBot
bot
Hi! Which symptom would you like to report? 16:29
I cough while running 16:29
Thank you. Information acknowledged! 16:29
Do you think that your symptoms are not getting any better or may be worsening? 16:29
I do not think so 16:29

b)
Do you feel shortness of breath in general and also when talking? 16:30
Not really 16:30
Do you feel shortness of breath in general? 16:30
Hmmm I cough at night 16:30
Thank you. Information acknowledged! 16:30
Have you had rhinitis associated with some coughing? 16:30

c)
Definitely yes 16:30
You are having a light crisis! 16:30
Please use the bronchodilator. Take 3 puffs when neeeded, every 4-6 hours for 1 day. Vary the interval between the doses to follow the the cough trend. 16:30

Figure 4: Sample of a Human-Machine Conversation in the Asthma Domain.

had expertise in allergy and immunology, and one in pediatric pulmonology. The experts were invited to test the agent, playing the role of an end-user (i.e. asthma patient). After a few interactions, they were invited to fill out the System Usability Scale (SUS) [20] questionnaire. The SUS score for the agent was 68.75. A score of 68% has been used in the literature as the usability threshold for interactive systems. This way, the agent still can be considered above average, although there is lots of room for improvement. A more extensive user evaluation including the target users of the system is expected for future work.

7. Conclusion

This work presented our approach for the dynamic generation of dialogue managers for the health domain. We introduced the three units that compose this approach as well as the additional resources that support dialogue interactions, namely a conversational ontology and an agent-based framework for action selection within health scenarios. However, our focus was directed to the plan-based aspects of the approach; by modeling the dialogue as a FOND planning problem, our approach supports state tracking and delivers mixed-initiated and predictable policies. The feasibility of deploying a resulting dialogue manager into a real-world problem was tested for different problem sizes and an early user study showed experts' appreciation for a resulting agent. Our results show efficiency in automating the generation of policies for realistic-sized health dialogues, i.e. that address a medical guideline. Since in the dialogue community, there is no consensus on what comprises a good dialogue system, in future work we intend to complement the evaluation here described with extensive user studies. In these studies, we assess (i) the quality of the dialogue system and its acceptance by our target users as well as (ii) the reusability of the approach for different domains.

References

[1] M. Santos Teixeira, M. Dragoni, A review of plan-based approaches for dialogue management, Cognitive Computation (2022) 1–20.
[2] W. N. Price, Big data and black-box medical algorithms, Science translational medicine 10 (2018) eaao5333.
[3] L. Laranjo, A. G. Dunn, H. L. Tong, A. B. Kocaballi, J. Chen, R. Bashir, D. Surian, B. Gallego, F. Magrabi, A. Y. Lau, et al., Conversational agents in healthcare: a systematic review, Journal of the American Medical Informatics Association 25 (2018) 1248–1258.

[4] D. Walton, E. C. Krabbe, Commitment in dialogue: Basic concepts of interpersonal reasoning, SUNY press, 1995.
[5] H. Weld, X. Huang, S. Long, J. Poon, S. C. Han, A survey of joint intent detection and slot-filling models in natural language understanding, CoRR abs/2101.08091 (2021).
[6] A. Gatt, E. Krahmer, Survey of the state of the art in natural language generation: Core tasks, applications and evaluation, J. Artif. Intell. Res. 61 (2018) 65–170. URL: https://doi.org/10.1613/jair.5477. doi:10.1613/jair.5477.
[7] M. S. Teixeira, C. da Costa Pereira, M. Dragoni, Information usefulness as a strategy for action selection in health dialogues, in: 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE, 2020.
[8] C. Muise, T. Chakraborti, S. Agarwal, O. Bajgar, A. Chaudhary, L. A. Lastras-Montano, J. Ondrej, M. Vodolan, C. Wiecha, Planning for goal-oriented dialogue systems, arXiv preprint arXiv:1910.08137 (2019).
[9] S. Sreedharan, T. Chakraborti, C. Muise, Y. Khazaeni, S. Kambhampati, −d3wa+−a case study of xaip in a model acquisition task for dialogue planning, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 30, 2020.
[10] J. Baskar, H. Lindgren, Cognitive architecture of an agent for human-agent dialogues, in: International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer, 2014, pp. 89–100.
[11] K. Lee, J. K. Kim, M. W. Park, L. Kim, Situation based dynamic planning for dialog support with hierarchical knowledge, in: 2016 International Conference on Platform Technology and Service (PlatCon), IEEE, 2016, pp. 1–4.
[12] G. Behnke, D. Ponomaryov, M. Schiller, P. Bercher, F. Nothdurft, B. Glimm, S. Biundo, Coherence across components in cognitive systems—one ontology to rule them all, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[13] M. Schiller, G. Behnke, M. Schmautz, P. Bercher, M. Kraus, M. Dorna, W. Minker, B. Glimm, S. Biundo, A paradigm for coupling procedural and conceptual knowledge in companion systems, in: 2017 International Conference on Companion Technology (ICCT), IEEE, 2017.
[14] T. W. Bickmore, D. Schulman, C. L. Sidner, A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology, Journal of biomedical informatics 44 (2011) 183–197.
[15] M. S. Teixeira, V. Maran, M. Dragoni, The interplay of a conversational ontology and ai planning for health dialogue management, in: The 36th ACM/SIGAPP Symposium On Applied Computing, 2021.
[16] M. Ghallab, D. Nau, P. Traverso, Automated Planning: theory and practice, Elsevier, 2004.
[17] A. Botea, C. Muise, S. Agarwal, O. Alkan, O. Bajgar, E. Daly, A. Kishimoto, L. Lastras, R. Marinescu, J. Ondrej, et al., Generating dialogue agents via automated planning, arXiv preprint arXiv:1902.00771 (2019).
[18] C. Muise, S. A. McIlraith, V. Belle, Non-deterministic planning with conditional effects, in: Twenty-Fourth International Conference on Automated Planning and Scheduling, 2014.
[19] M. Walker, S. Whittaker, Mixed initiative in dialogue: An investigation into discourse segmentation, arXiv preprint cmp-lg/9504007 (1995).
[20] J. Brooke, et al., Sus-a quick and dirty usability scale, Usability evaluation in industry (1996).