Process-oriented Intents: A Cornerstone for Superimposition of Natural Language Conversations over Composite Services

Sara Bouguelia[1] 0000-0003-4538-0927, Auday Berro[1] 0000-0003-2411-5761, Boualem Benatallah[2] 0000-0002-8805-1130, Marcos Báez[3] 0000-0003-1666-2474, Hayet Brabra[4] 0000-0001-7484-2268, Shayan Zamanirad[5] 0000-0002-3371-9631, and Hamamache Kheddouci[1] 0000-0002-5561-6203

[1] LIRIS – University of Claude Bernard Lyon 1, Villeurbanne, France
{sara.bouguelia, auday.berro, hamamache.kheddouci}@univ-lyon1.fr
[2] Dublin City University, Ireland boualem.benatallah@dcu.ie
[3] Bielefeld University of Applied Sciences, Germany marcos.baez@fh-bielefeld.de
[4] SAMOVAR – Telecom SudParis, Institut Polytechnique de Paris, France
hayet.brabra@telecom-sudparis.eu
[5] University of New South Wales (UNSW), Sydney Australia
shayanz@cse.unsw.edu.au

Abstract. Task-oriented conversational assistants are in very high demand these days. They employ third-party APIs to serve end-users via natural language interactions and improve their productivity. Recently, the augmentation of process-enabled automation with conversational assistants emerged as a promising technology to make process automation closer to users. This paper focuses on the superimposition of task-oriented assistants over composite services. We propose a Human-bot-Process interaction acts that are relevant to represent natural language conversations between the user and multi-step processes. In doing so, we enable human users to perform tasks by naturally interacting with processes.

Keywords: Task-oriented Conversational Bots · REST APIs · Software-enabled Services · Composite Services · Process-oriented intents

1 Introduction

Task-oriented conversational services (or simply chatbots) emerged as engines for transforming online service-enabled digital assistance and powering natural interactions between humans, services, and things [15]. Recently, organizations leveraged chatbots in a variety of assistance tasks. For instance, the augmentation of process-enabled automation with task-oriented chatbots emerged as a promising technology to make process automation even closer to users [1,5]. This evolution promises to increase the benefits of automation by simplifying access and reuse of concomitant capabilities across potentially large number of evolving and heterogeneous data sources, applications and things [5,9]. While today's chatbots may automate some tasks, bot developers have recently started

investigating the incorporation of robotic process automation (RPA) to increase automation [14]. For instance, Devy chatbot was proposed to provide automated support in DevOps processes [6]. Authors in [13] developed a chatbot for agile software development teams which analyzes teams' project data to provide insights into their performance. In [12], the authors proposed an approach that automatically builds a chatbot from a process model to query process structure. Another work proposed a chatbot to query event data allowing users to get insights into specific process executions [11]. All these works are either about domain-specific chatbots or about querying the process execution or structure but do not focus on performing process tasks. Other works propose approaches to interact with business processes and perform process tasks through chatbots. For instance, Google proposes the use of a chatbot in so-called communication-enabled business process applications [8]. However, no specific details about the internals of the chatbot infrastructure are provided. The closest work to ours is [10], which proposed a methodology that takes a business process model as input and generates a chatbot to help the users interact with the process. However, the work does not focus on the recognition of process-oriented intents. In our previous work, we proposed various techniques for the superimposition of task-oriented chatbots on top of APIs [17,16,4,3].

In this paper, we focus on the superimposition of task-oriented chatbots over *composite services*. In doing so, we enable users to perform tasks by naturally interacting with service orchestrations involving multiple actions. Orchestrating human–machine conversations over composite services requires rich abstractions and knowledge to: (i) interact with a multi-step processes using natural language utterances, (ii) automatically recognise nuanced, context sensitive and possibly ambiguous process-aware user intents including starting a new task, inquiring about task progress, switching from one task to another and exceptional behavior such as canceling. Specifically, we identify fine-grained Human-bot-Process (HP) interaction acts that are relevant to represent natural language conversations between user and multi-step processes. In a nutshell, interaction acts are dialogue acts that characterise process-oriented intents in user utterances.

2 Preliminaries and Architecture

In this section, we first introduce some process-related concepts and assumptions. Second, we present a scenario illustrating interactions between a user and multi-step processes. Finally, we present the architecture.

Preliminaries. A *business process* is a collection of coordinated tasks to achieve a concrete goal [7]. The *schema* of a process can be represented in a variety of forms, such as Petri nets and Event-Driven Process Chains [7]. For simplicity, we represent the process schema as a directed acyclic graph. Figure 1 shows an example of a Travel Booking Process graph. The process graph nodes represent *activities* and *decision points*. A process is associated with a set of *exception handling policies*. Exception handling policies are directives that model exceptional situations together with a set of actions that are used to handle exceptions (e.g.,

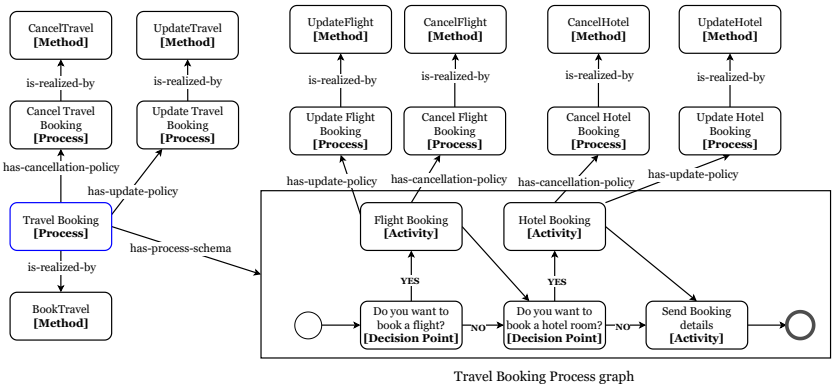

Fig. 1. Example of a Travel Booking Process Model

cancel a travel booking) [2]. In this paper, we consider that a process is realized by a composite service. Furthermore, a composite service is accessible through an API that includes: a main method to invoke the normal process behavior (e.g., *BookTravel* method), and exception handling methods to handle exceptional behaviors of a process (e.g., *CancelTravel* method) or an activity (e.g., *CancelHotel* method). We also consider that a process has a set of *correlation attributes* that uniquely identify an *instance* [7].

Scenario. Figure 2 shows an example of a user-chatbot conversation in which the user is interacting with the Travel Booking process illustrated in Figure 1. There are interactions that are triggered by the user and others by the chatbot. During normal process execution (e.g., booking travel): (i) the chatbot can *ask for more information to fulfill a task* (message 10) or *provide information* about a performed task (message 6), (ii) the user can *provide information* (message 11) or *inquire about task progress* (message 23). As mentioned before, a process is associated with possible run-time exceptions. In message 27, the user wants to change the check-out date. This interaction (changing date) is triggered by the user and involves the update of a previously performed task. However, exception interactions can also be triggered by the chatbot. For example, assume that the airline company canceled the user's flight. The chatbot can trigger an interaction that involves notifying the user about the cancellation and proposing alternatives such as changing the travel date. In this paper, we focus on interactions from the user side; chatbot-initiated interactions are outside the paper scope.

Architecture. To support natural language conversations with processes, the chatbot needs a set of services to initiate, monitor, and control task-related conversations. The *Natural Language Understanding (NLU)* service aims to extract HP interaction acts and slot-value pairs from the utterance. The *Process Embedding Service (PES)* aims to identify the *process* that corresponds to the utterance. The *Dialogue Manager (DM)* service aims to infer the dialogue state in terms of user intent and its slot-value pairs. This DM relies on the *Context Knowledge Service (CKS)* to recognize user intent and infer missing information. Once the DM recognizes the intent and collects all required information, it per-

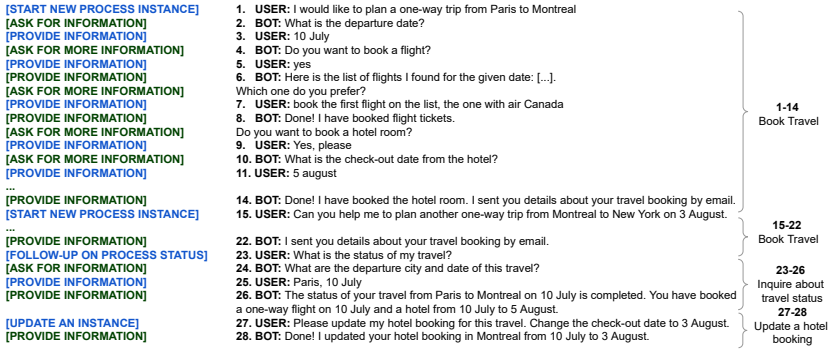

Fig. 2. Example of a user-chatbot conversation. Interaction acts in blue are triggered by the user and those in green are triggered by the chatbot.

forms the corresponding action and sends the results to the *Natural Language Generator (NLG)*. The NLG generates then human-like responses to the user. In what follows, we describe PES and CKS.

The PES aims to identify processes from natural language utterances. It has (i) a process knowledge model and (ii) a set of services to leverage this knowledge. The process knowledge model is denoted as a process knowledge graph (P-KG) with specific types of nodes and relationships. In particular, nodes can describe *Processes*, *Paths*, and *Activities*. Part of the information in the P-KG is the graph representation of what we find in the process model definition. Such information typically includes the process *name*, the process *description*, activity *name*, and activity *description*. Furthermore, a *Process* node has relationships such as *is-realized-by* that denotes that a process is performed by an API method (refer to Figure 1). The PES features the following three services:

- *Vector generation Service* is used to construct vector embeddings for process elements. It takes as input a process element `e` and generates its vector embedding. It generates: (i) an *activity vector* by aggregating the information from activity *name* and its *description*; (ii) a *path vector* by aggregating vectors of activities in this path; and (iii) a *process vector* by averaging the information from process *name*, process *description* and all path vectors.
- *Process Identification Service* aims to identify the corresponding process of a given utterance `u`. First, it generates the embedding vector of the utterance. Second, it calculates the cosine similarity between this utterance vector and the vector of each process in the P-KG. Then, according to a predefined threshold, processes with similarities greater than this threshold are kept and ordered. Finally, the service returns the process scoring the highest similarity.
- *Method Identification Service* aims to identify a process API method that corresponds to a given process intent. It takes as input a process `p` (e.g., *Travel Booking*), a process intent `i` (e.g., *canceling task*) and an activity `a` (e.g., *Hotel Booking*) and returns the corresponding API method.

In [3], we proposed the CKS that enables to capture contextual knowledge from different sources. We extend the CKS with two additional services:

– *Process Instance Identification Service* returns the list of instances for a given process. It takes as input a process `p` and returns the list of instances of `p`.
– *Correlation attribute Value Retrieval Service* provides values of correlation attributes for a given process instance id. It takes as input an instance id and returns an array of attribute-value pairs.

3 Process-aware User Intents

Conversations regarding a given process-aware task may involve several turns (e.g., starting a travel booking, later inquiring about booking status, modifying travel dates, or canceling the booking). We propose HP interaction acts to characterize a set of elementary user intents in conversations between users and multi-step processes. Specifically, we derive four types of process-oriented intents: *start new process instance* intent, *follow-up on process status* intent, *canceling task* intent, and *task update* intent.

We propose five general steps to recognize these process-oriented intents from an utterance `u`: (i) detect the *HP interaction act class* expressed in `u`; (ii) invoke the *Process Identification PES* service to get the corresponding process `p`; (iii) invoke the *Process Instance Identification CKS* service to retrieve the set of instances `set_i` of `p`; (iv) extract the values of correlation attributes of the instance that the user is referring to; (v) compare the extracted values of correlation attributes with those of `set_i` to check if the identified instance already exists or not. In what follows, we describe the process-oriented intents and define rules that combine detection of HP interaction acts with additional context and process knowledge to recognize and realize each of these intents.

Start New Process Instance. This intent allows to identify whether the user utterance expresses a task that requires the creation of a new process instance. In general, when users ask for a new task, they provide general information describing this task, and sometimes they provide more detailed information about this task. The chatbot needs this information to identify the process and to check if the utterance concerns the creation of a new process instance. Figure 3 shows the specification of the rule related to *start-new-process-instance* intent. This rule consists of *trigger* and *action* clauses. The trigger clause defines three boolean conditions. (1) The condition `IS_START()` checks if the utterance `u` expresses a *start new instance* HP interaction act. (2) The condition `IS_SIM()` checks if the process `p` corresponds to the utterance `u`. (3) The last condition `EXIST_INSTANCE()` compares values of the correlation attributes with those of the existing instances to check if the identified instance does not exist. If the conditions are satisfied, the chatbot (1) invokes the *Method Identification PES* service to get the method `m` and (2) triggers `m` to start the process execution.

Follow-up on Process Status. This intent allows inquiring about process instance status (e.g., pending, in-progress or completed). For example, in Figure 2 utterance 23, the user is inquiring about travel status. Figure 3 shows the specification of the rule related to this intent. The conditions are the same as those defined in the previous rule, except that this rule needs to detect a *follow-up* HP interaction act and the process instance should exists. If all conditions

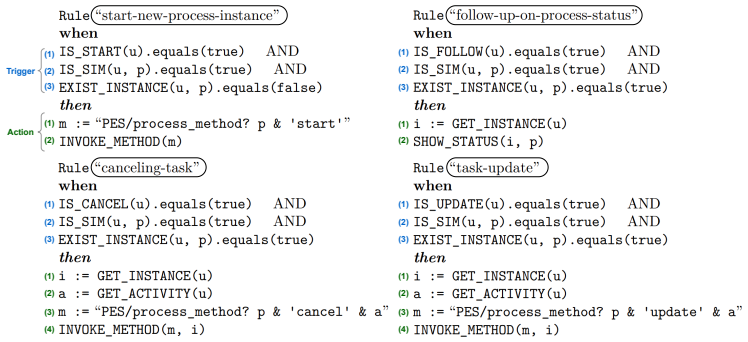

Fig. 3. Rules to recognize and realize the identified process-oriented intents.

are satisfied, the chatbot (1) retrieves the corresponding instance and (2) lists the status of this instance.

Task Update. This intent allows to identify whether the user wants to update an existing process instance (e.g., update the hotel check-out date). A user can request to update an information in the whole process, or a specific activity in the process. We model an activity as an input parameter of the intent *task update*, thus the chatbot can extract from the user utterance the activity that the user wants to update. The conditions are the same as those defined in the first rule, except that this rule needs to detect a *task update* HP interaction act and the process instance should exists. If all conditions are satisfied, the chatbot (1) retrieves the corresponding instance, (2) extract the value of the *activity* parameter, (3) invokes the *Method Identification PES* service to get the the corresponding method `m` and (4) triggers `m` to update the corresponding task.

Canceling a Task. This intent allows to identify whether the user utterance expresses a task cancellation of an existing process instance. The user can request to cancel the whole process (e.g., canceling travel bookings), or a specific task in the process (e.g., canceling hotel booking). The steps to recognize and realize *canceling task* intent are the same as those in the *task update* intent (Figure 3).

4 Implementation and Experiments

The first objective of the study was to explore the *effectiveness* of the proposed approach, i.e., its capability of recognizing correctly the process-oriented intents presented in Section 3 and reducing unnecessary interactions. The second objective was to assess the impact of enabling interaction with a process as opposed to leaving users to orchestrate services themselves to fulfill their goals.

Experimental design.[6] Participants were recruited via email from the extended network of contacts of the authors. The call for volunteers resulted in a total of 17 participants. The evaluation scenario required participants to perform tasks associated with three underlying processes (Travel Booking, Shopping and

[6] Experimental materials: `https://tinyurl.com/ICSOC22StudyMaterials`

Table 1. Performance of experimental conditions for each task according to the relevant metrics. Values in bold denote best performance.

Task	Baseline-bot			Process-bot			
	M1 TURNS	M2 PROMPTS	M4 INTENT	M1 TURNS	M2 PROMPTS	M3 PROCESS	M4 INTENT
T1 (new)	54,14%	45,59%	51,75%	**95,73%**	**91,67%**	**92,86%**	**88,68%**
T2 (update)	31,58%	26,67%	25,00%	**85,71%**	**80,00%**	**91,67%**	**83,33%**
T3 (follow up)	47,15%	51,28%	27,86%	**82,61%**	**88,89%**	**85,71%**	**85,71%**
T4 (cancel)	27,27%	18,75%	33,33%	**80,00%**	**75,00%**	**91,67%**	**83,33%**
Mean	40,04%	35,57%	34,49%	**86,01%**	**83,89%**	**90,48%**	**85,27%**

Scheduling an appointment). Participants were asked to complete 4 tasks in this scenario (T1: starting new process instances, T2: updating information of process instances, T3: following up on process statuses, T4: canceling process instances). We followed a within-subjects design tasking participants to complete the above tasks by interacting with two chatbots representing the following conditions: (i) *Baseline-bot* that implements a standard conversational management; (ii) *Process-bot* that support PES and CKS services as well as the defined rules.

Procedure. The study was conducted online. After reading the informed consent and agreeing to participate, participants were introduced to the evaluation scenario and tasks (T1-T4). They were asked to perform those tasks with the two chatbots, in a counter-balanced design. After interacting with each chatbot, participants were asked: to select their preferred chatbot, to specify why, and provide quantitative feedback on their experience along three dimensions: *naturalness* (ability to fulfill user tasks in natural language), *repetitiveness* (ability to avoid redundant questions) and *understanding* (ability to interpret requests).

Data analysis. We performed an analysis of conversation logs so as to assess the effectiveness of our approach in recognizing the process-oriented intents. These are computed in relation to optimal conversation scenarios (i.e., scenarios assuming ideal accuracy of process-oriented intent recognition) that we designed based on participants conversations. The effectiveness is calculated by considering the following metrics: number of (M1) conversation turns, (M2) prompts asking for missing information, (M3) process correctly identified and (M4) process-oriented intents correctly recognized.

Results. Table 1 shows the relative performance by task of both baseline-bot and process-bot in relation to the optimal reference scenario. For the four tasks, we can see that process-bot experienced a boost in performance M1 and M2, approaching the efficiency of the reference scenario in terms of number of turns (M1) and prompts (M2). This level of performance is possible thanks to the PES and CKS services and the defined rules that allow to perform a mean relative performance across tasks for process identification (M3) and intent recognition (M4) of 90,48% and 85,27% respectively. In contrast, not supporting these rules leads the baseline-bot to perform poorly in comparison, with the best performance being at around 36,70% for the considered metrics. Regarding the user experience, all but two participants (15/17 participants) expressed a preference towards the process-bot as opposed to the baseline-bot. The feedback to the specific user experience questions, highlighted the reasons behind the preference. The majority of participants reported that process-bot interactions described *natu-*

ralness (11/17), less *repetitiveness* (11/17) and *understanding* (12/17), whereas the baseline-bot was poorly rated on these fronts (2/17).

5 Conclusions and Future Work

We proposed process-oriented intents that are relevant to represent natural language conversations between the user and multi-step processes. We devised an approach that combines recognition of these intents from user utterances with additional context and process knowledge to enable users to perform tasks by naturally interacting with service orchestrations. Future work includes identifying a new pattern that allows selecting a service based on subjective attributes.

References

1. Barukh, M.C., et al.: Cognitive augmentation in processes. In: Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future, pp. 123–137. Springer (2021)
2. Benatallah, B., et al.: The self-serv environment for web services composition. IEEE internet computing **7**(1), 40–48 (2003)
3. Bouguelia, S., et al.: Context knowledge-aware recognition of composite intents in task-oriented human-bot conversations. Proc. CAiSE 2022
4. Bouguelia, S., et al.: Reusable abstractions and patterns for recognising compositional conversational flows. Proc. CAiSE 2021
5. Brabra, H., et al.: Dialogue management in conversational systems: a review of approaches, challenges, and opportunities. IEEE Transactions on Cognitive and Developmental Systems (2021)
6. Bradley, N., et al.: Context-aware conversational developer assistants. In: 2018 IEEE/ACM 40th ICSE. pp. 993–1003. IEEE (2018)
7. Dumas, M., et al.: Fundamentals of business process management, vol. 1. Springer (2013)
8. Gaulke, D., et al.: Interactive user interface to communication-enabled business process platforms method and apparatus (May 26 2015), uS Patent 9,043,407
9. Hofmann, P., et al.: Robotic process automation **30**(1), 99–106 (2020)
10. Kalia, A.K., et al.: Quark: a methodology to transform people-driven processes to chatbot services. In: Proc. ICSOC. pp. 53–61. Springer (2017)
11. Kobeissi, M., et al.: An intent-based natural language interface for querying process execution data. In: Proc. ICPM. pp. 152–159. IEEE (2021)
12. López, A., et al.: From process models to chatbots. In: International Conference on Advanced Information Systems Engineering. pp. 383–398. Springer (2019)
13. Matthies, C., et al.: An additional set of (automated) eyes: chatbots for agile retrospectives. In: Proc. BotSE. pp. 34–37. IEEE (2019)
14. Rizk, Y., et al.: A unified conversational assistant framework for business process automation. arXiv preprint arXiv:2001.03543 (2020)
15. Sheth, A., et al.: Cognitive services and intelligent chatbots: current perspectives and special issue introduction. IEEE internet computing **23**(2), 6–12 (2019)
16. Zamanirad, S., et al.: Hierarchical state machine based conversation model and services. Proc. CAiSE 2020
17. Zamanirad, S., et al.: Programming bots by synthesizing natural language expressions into api invocations. Proc. ASE 2017