Modeling a Conversational Agent using BDI Framework

Alexandre Yukio Ichida
Pontifical Catholic University of Rio Grande do Sul
alexandre.ichida@edu.pucrs.br

Felipe Meneguzzi
University of Aberdeen
Pontifical Catholic University of Rio Grande do Sul
felipe.meneguzzi@abdn.ac.uk

ABSTRACT

Building conversational agents to help humans in domain-specific tasks is challenging since the agent needs to understand the natural language and act over it while accessing domain expert knowledge. Modern natural language processing techniques led to an expansion of conversational agents, with recent pretrained language models achieving increasingly accurate language recognition results using ever-larger open datasets. However, the black-box nature of such pretrained language models obscures the agent's reasoning and its motivations when responding, leading to unexplained dialogues. We develop a belief-desire-intention (BDI) agent as a task-oriented dialogue system to introduce mental attitudes similar to humans describing their behavior during a dialogue. We compare the resulting model with a pipeline dialogue model by leveraging existing components from dialogue systems and developing the agent's intention selection as a dialogue policy. We show that combining traditional agent modelling approaches, such as BDI, with more recent learning techniques can result in efficient and scrutable dialogue systems.

CCS CONCEPTS

• **Computing methodologies** → **Reasoning about belief and knowledge**;

KEYWORDS

belief-desire-intention, task-oriented dialogue systems, autonomous agent, machine learning

ACM Reference Format:
Alexandre Yukio Ichida and Felipe Meneguzzi. 2023. Modeling a Conversational Agent using BDI Framework. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-March 31, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3555776.3577657

1 INTRODUCTION

Research and development of conversational agents has seen a renaissance in recent years as a result of advances on natural language processing [14]. Natural language is an essential interface between agents and humans in artificial intelligence applications related to human-computer interaction. Conversational assistants are autonomous agents that interact in natural language to assist in a wide range of tasks, such as domain-specific information retrieval. Task-oriented conversational agents focus on helping humans to achieve their goals in single or multi-domains, which differ from open-domain conversational agents regarding the domain covered in conversation with a human. Key to task-oriented agents is their goal-driven behavior[23], which leads the agent to make decisions by examining the dialogue state in a multi-turn conversation. Booking systems (i.e., restaurants, hotels) constitute a typical commercial application that involves a task-oriented agent, where the goal is to help a user book services given user-defined criteria.

Traditional approaches to artificial intelligence often model decision-making by borrowing terminology from folk psychology, which describes human mental attitudes to implement rational agents. The belief-desire-intention (BDI) model [1, 3] introduces a conceptual framework to implement autonomous agents composed of beliefs, desires, and intentions. The BDI architecture encodes the agent's behavior as plan rules to instruct it on achieving particular (implicit) goals given specific context conditions. [10]. Specifically, plans represent a sequence of actions the agent should perform given a set of conditions entailed by the agent's belief base, which are manually developed by humans in a structure named Plan Library. Such terminology is useful in developing and debugging autonomous agents in a variety of domains [13] since this architecture describes information about the agent beliefs.

In this paper we develop a BDI-style agent for task-oriented dialogues that seamlessly blends a symbolic practical reasoning mechanism with machine learning techniques for planning and natural language understanding. Our contributions are as follows. First, we develop a specialized BDI agent architecture for task-oriented dialogues that aims to help humans to achieve their goal in domain-specific tasks (Section 3). This allows us to represent knowledge in a logic-based format within the agent's belief base, and take advantage of the classical BDI reasoning cycle. This facilitates describing the agent reasoning during a dialogue. Second, we include a learnable dialogue policy in the agent architecture as an intention structure to reduce human effort in agent development. Third, we apply different learning paradigms and combine them to show how a BDI model can be optimized without predefined plan-rules. We show experimentally that the BDI dialogue agent can learn how to respond by using supervised learning with an annotated dataset and reinforcement learning by using a simulated environment.

2 BACKGROUND

Task-oriented dialogue systems are conversational agents that focus on achieving predefined user goals. Zhang et al. [23] divide existing approaches to task-oriented agent development into two types according to their architecture: pipeline methods and end-to-end

SAC '23, March 27-March 31, 2023, Tallinn, Estonia

ACM ISBN 978-1-4503-9517-5/23/03…$15.00
https://doi.org/10.1145/3555776.3577657

methods. End-to-end methods use a single model that receives the user's natural language utterance and directly outputs the agent response, often relying on novel neural network techniques by leveraging large annotated corpora. By contrast, a pipeline task-oriented dialogue system comprises different parts organized in the following components: natural language understanding (NLU) to identify the user intention; dialogue state tracking (DST) to maintain the state of conversations; dialogue policy to select a dialogue act to be performed; and natural language generation (NLG) to translate the action selected into an understandable answer.

The BDI model is a framework to develop autonomous rational agents bound with mental attitudes in its architecture [16]. Inspired by Bratman's philosophical work [1], the BDI framework introduces a practical reasoning approach that consists of three basic components: beliefs, desires, and intentions. Beliefs represent the information about the environment according to the agent's perceptions, which describe its current state. During the agent execution, the agent observes events from the environment and might include new beliefs or update the existing ones in a Belief Base data structure, which stores all agent beliefs. Desires represent states of affairs that the agent aims to achieve to satisfy its design goals. The intention component is a structure that consists of a set of instantiated plans adopted by the agent to achieve a subset of its desires.

Since task-oriented agents interact with a user in a multi-turn conversation, it is necessary to track the dialogue state in order to answer consistently throughout the dialogue session. Pipeline task-oriented agents rely on the *dst* component to represent the state, while BDI agents use their belief base to store each agent's beliefs in an environment. Considering that the *dst* tracks the dialogue state by storing all observations about user utterances, we assume that there is a correspondence between the *dst* component and the belief base. A BDI agent interacts with the environment acting over by selecting an intention or plans to execute given a context entailed by its belief base. Similarly, a task-oriented agent relies on the actual dialogue state managed by the *dst* component to choose its next dialogue act. We can represent the dialogue acts as a plan since a task-oriented agent might execute multiple dialogues acts in a single answer analogous to a sequence of steps.

3 A BDI CONVERSATIONAL AGENT

In a BDI dialogue system, beliefs encode knowledge obtained during a conversation with a human, and the agent desire is to achieve the user goals by providing useful responses. Intentions represent which response the agent should commit to providing to a human given its current beliefs. This mental state representation allows us to inspect it during a conversation to explain why an agent has chosen a particular action, as recent research shows [4].

Traditional BDI agents often require a designer to build a plan library comprising either pre-determined plans or a set of complex planning rules. Such requirement creates two major problems to practical applications in the real world. First, they incur a substantial labor cost to agent development. Second, such agents have no mechanism to adapt its plan selection in cases where changes occur in the target environment. To minimize the human intervention and limitations of traditional BDI agents [10], we develop an intention selection as a dialogue policy that selects the agent response as a stochastic transition. In turn, our agent learns this policy via reinforcement learning, obviating the need for human fine-tuning of the plan selection criteria.

3.1 BDI Reasoning Cycle

The agent reasoning process starts with receiving the user utterance written in natural language. Similar to a pipeline task-oriented dialogue system, the agent extracts the user intention and parameters in the user utterance using an NLU component to formulate perceptions in a logical representation, which the agent can then include in its belief base. The agent then maps the formulated beliefs and executes an optimized dialogue policy to select which dialogue act should be triggered. Finally, the agent outputs an answer in natural language to the user expressing the dialogue acts predicted by the policy. We summarize the reasoning process in Figure 1.

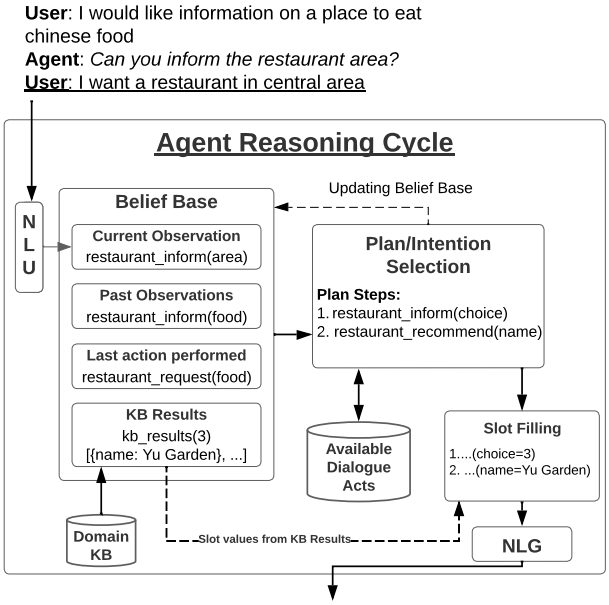

Figure 1: Diagram illustrating the agent architecture and its components given an example of the reasoning cycle in a dialogue session with a user.

In the agent reasoning process, both NLU and NLG are off-the-shelf components employed in existing dialogue systems with pipeline architecture. Notice that we do not assume any specific implementation of these components within the agent architecture, therefore, the agent is compatible with any existing NLU and NLG components. Indeed, Section 4 details the implementations we use in our experiments.

Within the reasoning cycle, the agent's belief base contains two key pieces of information during a dialogue, (translated) perceptions about user utterances and domain knowledge. The agent translates perceptions collected throughout a dialogue into beliefs that describe the user's intent and elements mentioned in an utterance.

The domain knowledge includes information and entities created by human experts (e.g., restaurants, hotels) that the agent should use to provide a response. The agent query the domain knowledge base (KB) about specific information from domain knowledge during the dialogue to satisfy the user goal given its perceptions and information stored in its Belief Base. The agent responds by taking into account the information retrieved from the knowledge base, which may lead to different directions in the dialogue flow. For example, when an entity required by the user is not in its domain knowledge, the agent should report that it could not follow the expected dialogue progression. On the other hand, if the entity exists, the agent can follow the direction to the original user goal.

Given the belief base built during dialogue, the agent adopts an intention to respond to the user through plan execution. Note that such plans consider a finite set of dialogue acts that follow the domain ontology, which keeps the agent response given the current domain under control. Consequently, the agent's internal state generates an event to update the belief base to include new beliefs or update existing ones considering the dialogue acts executed by the selected plan. In our agent instantiation, instead of manually creating plans for each available context, we apply machine learning methods to select single or multiple actions to learn the adopted plan. We detail the action selection process in Section 3.4.

Figure 1 illustrates the reasoning cycle of the agent in a dialogue related to a restaurant booking domain, given the underlined user utterance. First, the *nlu* component recognizes the intention *inform* and slot *place* from the domain *restaurant* to formulate the agent perceived belief. Second, the agent queries the domain knowledge base to represent the knowledge base results following its beliefs. Third, given an optimal dialogue policy, the agent selects its recommended action and informs the number of choices available. Finally, the *nlg* component translates the symbolic representation of each dialogue act in the selected plan into a natural language utterance.

3.2 Belief Base Representation

We include an NLU component in our BDI agent to formulate its beliefs from user natural language utterances given the utterance perceived during a dialogue. Such information describes the user's natural language utterance as a structured semantic representation, which consists of a set of finite symbols. Similar to traditional BDI agents, working with a structured semantic representation allows representing the agent's observation with a logical language.

The belief base stores all information perceived and collected from the domain knowledge base to describe the dialogue state, which has a similar role to the DST component in the task-oriented pipeline agent architecture. In our agent implementation, the belief base comprises the following dialogue elements: the observation perceived in the current turn, past observations, the actions triggered from the plan executed in the previous turn, and knowledge base results. The agent stores all observations perceived from past dialogue turns and unifies them with the current one in its belief base to track the dialogue state. While the agent executes the selected plan, the beliefs are updated to prepare the belief base for the next dialogue turn, which results in an update of the past observations to consider the current one and overwriting the last dialogue acts performed with the plan selected.

As illustrated in Figure 1, the agent queries the knowledge base by using as criteria the current and past observations perceived throughout the dialogue, which results in a new belief to represent how many results the agent retrieves. The agent uses the number of results to discern whether an entity stored in the domain knowledge base corresponds to the information mentioned by the user. We did not apply any approaches that prioritize or explore in-depth entity details retrieved from the knowledge base. Hence, for simplicity, the agent selects a random element in cases where the knowledge base query results in more than a single entity. In this paper, we describe the beliefs using a first-order logic language encoding the symbolic representation above. We encode the user intent as a predicate since it describes the user dialogue act and its argument encodes the slot information. In order to distinguish the context of user intent, we include the utterance domain in the predicate term. Given that, we formalize the representation of agent beliefs as follows:

Definition 3.1 (Belief Representation). Given a set of symbols D representing available conversation domains, I user intents, and S for slots defined by an ontology, the belief representation is a unary predicate symbol $d\_i(s)$ where $d \in D$, $i \in I$, $s \in S$.

The symbolic representation of beliefs uses the delexicalized form of each slot retrieved by the NLU component. The delexicalized form does not include the slot value retrieved and uses solely the slot type to reduce symbol diversity and, consequently, reduce the belief vocabulary size. However, in order to fulfil the agent response with slot values retrieved from the knowledge base in the slot filling step, we store all slot values retrieved in the agent's internal state. For example, if the user informs its intention to book a *restaurant* in a central area, we use the intention *inform* and slot *area* to describe the belief as $restaurant\_inform(area)$. The symbolic representation of dialogue acts performed by the agent in selected plans shares the same symbol schema of perceived beliefs. Given the same symbol represented as a dialogue act $restaurant\_inform(area)$, this example can represent the act that results in a message informing the restaurant area for the user.

3.3 Dialogue Policy as Intention Selection

The role of the dialogue policy in pipeline agent architectures is to specify the agent's response to a human throughout the dialogue with a human. The dialogue policy describes how the agent should respond to a human with a probability distribution to represent the likelihood to select a specific action. By contrast, traditional BDI agents use plans and actions that are explicitly programmed by humans. The agent then selects these plans to be adopted as intentions through an intention selection process. Pipeline agents learn dialogue policies through supervision or experience acquired in a simulated environment. In our agent, we develop an intention selection mechanism similar to a dialogue policy. We formalize the dialogue policy component using the following definition:

Definition 3.2 (Intention Selection). The Intention Selection policy π is a neural network function $f : \mathbb{R}^{|Bel|} \rightarrow \mathbb{R}^{a}$ that maps the belief base and results into a plan of actions to be performed by the agent.

Specifically, the agent uses a simple multi-layer perceptron as a function f that receives the dialogue state in a vector representation and generates another vector indicating the actions selected by the

policy. Note that using a symbolic representation of user actions grounded by an ontology allows formulating a fixed-size vocabulary with all unique beliefs. Given a vocabulary v that contains all beliefs supported by the agent, the policy input representation is a binary vector with size $|v|$ mapping each position to a single belief mapped to 1 if a specific belief is present in the belief base and 0 otherwise.

The output of f is a numerical vector representing the agent's probability of adopting each action, given the belief base on the current dialogue turn. The probability distribution provided by the dialogue policy consists of a multi-discrete action space where it represents a single discrete space for each available dialogue act. In this paper, we use the symbolic dialogue act representation similar to the symbolic belief representation, in which we generate a symbol for all available actions handled by the agent. Multi-discrete action space is amenable to represent plans since it allows the agent to execute multiple dialogue acts in the same turn (e.g., inform the name and the area of a specific restaurant as a sequence of steps). Regarding the execution step order of selected plans, we use the sequence index of the probability distribution vector generated by the dialogue policy. Figure 2 illustrates the inputs and outputs of the dialogue policy neural network.

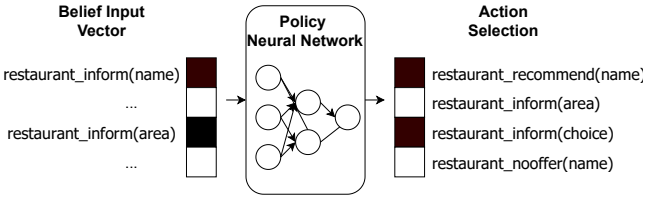

Figure 2: Process of action selection illustrated in Figure 1 by using the policy neural network.

3.4 Dialogue Policy Training

We combine two learning paradigms to enhance the agent plan selection given its beliefs. Similar to the work from Goecks et al. [6], we develop an actor-critic [8] approach to optimize the dialogue policy using a supervised learning method and a reinforcement learning method. Specifically, we develop the dialogue policy optimization in the following steps: First, we train the dialogue policy using an annotated dataset example of conversations between a user and a simulated human. Second, we optimize the dialogue policy using a reinforcement learning algorithm given a simulated environment where the agent interacts and receives rewards for performed actions.

3.4.1 Supervised Training. The supervised training of dialogue policies uses a dialogue dataset that contains the respective annotated dialogue acts. The dialogue policy must predict multiple actions and, hence, we apply supervised learning similar to a classification task considering each dialogue act as a mutually non-exclusive class. Inspired by the Behavioral Cloning approach [11], we use each dialogue in the dataset as trajectories, iterating over turns and collecting each annotated dialogue state-action (s, a). In our architecture, the agent belief base represents the state s, whereas a is the expected action to be performed given its current beliefs. Thus, the supervised training optimizes the dialogue policy π_θ^{bc} by minimizing a loss function $L_{bc}(a, \pi_\theta^{bc}(s))$. Since each action space is an independent distribution and each output is a value between 0 and 1, in this step, we use a binary cross-entropy loss function:

$$L_{bc} = a \cdot \log \sigma(\pi_\theta^{bc}(s)) + (1 - a) \cdot \log(1 - \sigma(\pi_\theta^{bc}(s))) \quad (1)$$

3.4.2 Simulated Environment Training. In the second step of the policy optimization, we reuse the policy network parameters learned from the previous step as a starting point of the simulated environment training. This simulated environment is amenable to applying reinforcement learning approaches. Specifically, given the policy π_θ^{bc} with parameters θ pretrained with the supervised training method, we then apply Proximal Policy Optimization (PPO) [19] as a second optimization step to generate a policy π_θ^{bc+ppo}. Schulman *et al.* [19] argue that traditional Policy Gradient methods can generate large policy updates that degrade the policy performance. In our case, such large updates can lead to information loss of policy π_θ^{bc+ppo} by erasing the learning parameters from π_θ^{bc}.

In our work, we apply a PPO algorithm based on a clipped surrogate objective by using a generalized advantage estimation. Specifically, given a dialogue turn t, we compute the probability ratio $r_t(\theta)$ between the new and the old policies considering the update step and optimize the surrogate objective L^{CLIP} following the equations:

$$r_t(\theta) = \frac{\pi_{\theta_{old}}(a_t|s_t)}{\pi_{\theta_{new}}(a_t|s_t)} \quad (2)$$

$$L^{CLIP}(\theta) = \mathbb{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \quad (3)$$

where ϵ is a hyperparameter that defines the bounds of the clipping operation over the ratio $r_t(\theta)$ [19]. Clipping the policy update restricts the probability ratio between $\pi_{\theta_{old}}$ and $\pi_{\theta_{new}}$ into the interval $[1-\epsilon$ and $1+\epsilon]$, which minimizes the problem of traditional Policy Gradient stated before.

$\hat{A}$ represents the result of Generalized Advantage Estimation (GAE) [18], in which we use a Value Function $V(s)$ as follows:

$$\hat{A} = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (4)$$

$$\text{where } \delta_t = R_t + \gamma V(s_{t+1}) - V(s_t) \quad (5)$$

where γ is a hyperparameter that represents a discount factor and R_t is the immediate reward. We represent the value function $V_\mathbf{w}$ as a neural network with learnable parameters $\mathbf{w}$ with an architecture similar to the policy network, except that $V_\mathbf{w}$ predicts a single value instead of action probabilities. Although both models do not share the same parameters, we optimize the value network and policy network in the same step. We minimize the loss function L by computing the simple subtraction between v_{target} and v_t as stated in line 9 of Algorithm 1. We summarize our implementation of PPO update step considering a complete dialog in Algorithm 1.

4 EXPERIMENTS AND RESULTS

4.1 Implementation

We conduct our experiments using the Convlab components and evaluation methods [24]. Convlab is a standardized task-dialogue system platform that provides a wide variety of configurations and

Algorithm 1 PPO update step of Dialog Policy

Require: dialog states s, dialog actions a, immediate rewards R, number of dialogue turns T, pretrained dialogue policy network π_θ^{bc}, value function $V_\mathbf{w}$, discount factor γ

$\pi_\theta \leftarrow \pi_\theta^{bc}$
$\pi_{\theta_{old}} \leftarrow \pi_\theta$
for $t \leftarrow (T-1), ..., 0$ **do**
    $v_{target} \leftarrow R_t + \gamma * V_\mathbf{w}(s_{t+1})$
    $\hat{A}_t \leftarrow$ Compute GAE (Equation 4-5)
end for
for $t \leftarrow 0, ..., T$ **do**
    $v_t \leftarrow V_\mathbf{w}(s_t)$
    Minimize $L(v_{target}, v_t)$ to optimize $V_\mathbf{w}$
    Compute ratio between $\pi_{\theta_{old}}$ and π_θ (Eq. 2) selecting the action a_t
    Optimizes π_θ using $L^{CLIP}(\theta)$ using $\hat{A}_t$ (Eq. 3)
end for

settings to develop pipeline agents. We integrate off-the-shelf implementations and pretrained models for NLU and NLG components provided by the Convlab platform.

We use the MultiWOZ task-oriented dialogue corpus [21], which contains ten thousand annotated dialogues involving multiple domains. The dialogues cover the following domains: restaurant, hotel, attractions, taxi, train, policy, and hospital. The corpus also includes a predefined ontology with intent and slot taxonomies and a knowledge base with examples for each domain. For the belief vocabulary, we use a single vocabulary to represent all domains contained in MultiWOZ domain ontologies for input beliefs and available actions.

The policy neural network is a two-layer 100-dimensional network, which we optimize in both training steps using the RMSprop optimizer. In the PPO step, the value model uses the same neural network specification of the policy except for its output layer, which predicts a single value to estimate the state value. Since hyperparameter tuning is out of the scope of this work, we use the same hyperparameters used in Schulman et al. work [19]. We use *1e-3* as supervised learning rate and a smaller learning rate (*1e-4*) in PPO in order to preserve the knowledge transferred between steps.

In cases where the agent should execute the same dialogue acts multiple times in the same dialogue turn (e.g. the same dialogue act and slot for the same domain), we include the number of times that the agent executes the action as a suffix in the symbolic representation. For instance, the dialogue act `restaurant_inform(name)-3` should be triggered when the agent recommends three different restaurant names in the same response. This representation approach is the same used by the other Convlab provided models, and hence, we use it to make a fair comparison in our experiments. In order to deal with a tractable number of available combinations of domain/dialogue act/slots/occurrences, we use a limited subset of dialogue acts by selecting the 276 most frequent actions in annotated dialogues, which is the same number used by other Convlab provided models.

In order to carry out experiments in a dialogue environment to train the PPO algorithm, we use the user simulator provided by Convlab that simulates multi-turn dialogues using MultiWOZ settings. The simulator is an agenda-based user simulator [17] that uses a stack-like structure with complex predefined dialogue heuristics to mimic the user behavior. The user's stack contains randomly initialized user goals, which contain all slots required to conduct its actions through heuristics during an interaction with an agent. We limit dialogues to 40 utterances/turns to avoid endless dialogues in which the agent fails to provide useful answers. Since most interactions in the ConvLab simulator end within 10 turns, we capped the dialogs to 4 times this number of turns.

Table 1: Performance on MultiWOZ domain separated into three groups: an agent with explicitly programmed rules, our approach, and an end-to-end model.

NLU	DST	Policy	Comp rate	Succ rate	Inform P/R/F1	Turn (s/all)
BERT	RuleDST	RulePolicy	90.5	81.3	79.7/92.6/83.5	11.6/12.3
MILU	RuleDST	RulePolicy	93.3	81.8	80.4/94.7/84.8	11.3/12.1
BERT	BeliefBase	BeliefPPO	77.5	72.6	70.0/86.4/74.4	13.7/17.3
MILU	BeliefBase	BeliefPPO	72.4	65.2	66.5/81.7/70.5	13.5/18.1
	DAMD		39.5	34.3	60.4/59.8/56.3	15.8/29.8

4.2 Metrics

In this section, we detail all ConvLab metrics used in our experiments in order to measure the agent results in a simulated environment. [24]. We use dialogue metrics that track single turns to measure how well the agent could respond to the user utterance(*inform precision/recall/F1*). We also include metrics that track the entire dialogue since errors in a specific turn can be propagated to the following turns, which affects the overall results (*complete/-success rate*).

The *inform precision/recall/F1* measures the number of requests in which the agent replies as expected in a single turn. The *complete rate* measures whether the agent achieves all simulator goals at the end of the dialogue. The *success rate* measures whether the agent fulfills all user requests with the correct expected information, and the agent performs the booking operation expected by the simulator. Finally, we include the average number of turns of the entire simulated dialogues (both in successful dialogues, and over all dialogues) to measure the agent time efficiency. Note that complete rate and success rate may lead to different values since the simulated user may not necessarily have a booking operation as a goal.

4.3 MultiWOZ Task Completion Results

In this experiment, we compare our approach with other pipeline configurations to evaluate the effectiveness of our model to solve the user task. We also include end-to-end methods to compare our BDI agent with a black-box model. We conduct our experiments using metrics similar to Takanobu *et al* [20], which compares different task-oriented dialogue with mixed architecture types.

As the NLU role, we use a pretrained BERT-based [5] and the MILU (Multi-Intent Language Understanding) recurrent neural network [7], with both implementations provided by Convlab and optimized using the MultiWOZ corpus. The RuleDST is the default DST component for all Convlab implementations that includes MultiWOZ specific details in the dialogue state (i.e. informable slots, booking slots). Similar to the RuleDST, we develop a BeliefBase dialogue state tracker that represents all agent observations into

a logical representation. For all agent settings, we use the TemplateNLG to generate a natural language response, which relies on fixed phrases templates and fill values contained in the dialogue state on it. To compare with pipeline agents, we include the end-to-end Domain Aware Multi-Decoder (DAMD) neural network [22], which relies on an encoder network that receives the dialogue states to decode into a natural language response directly. DAMD uses a supervised learning method, jointly with a data augmentation technique applied in the MultiWOZ corpus. Table 1 shows metrics collected from 1000 simulated dialogues between the agent and a simulated user for different pipeline configurations.[1]

In this comparison, we also include a dialogue policy with explicitly programmed rules (RulePolicy), which outperforms machine learning techniques in all metrics including end-to-end approaches. Such results highlight the benefit of using explicitly programmed rules, even if it requires extensive human effort to anticipate each task-oriented dialogue situation. We show the difference of using a more suitable NLU component by using two different implementations. While our approach results are better using the BERT NLU component, the RulePolicy could achieve better results using the MILU component. Such differences in results convey the importance of using the best possible NLU component implementation since errors in understanding propagate to the subsequent components. The results of all pipeline architecture-based agents outperform the end-to-end approach in turn metrics (inform p/r/f1), and by a large margin in multi-turn metrics (complete/success rate).

4.4 MultiWOZ Domain Results

In this section, we detail the experiments stratified by each MultiWOZ domain to evaluate the agent performance given a specific context. This experiment uses the dialogue policy isolated from NLU components to avoid invalid beliefs. Similar to the previous Section, we simulated the agent through 1000 simulated dialogue sessions, with results shown in Table 2.

Although we achieve perfect metrics in the hospital and police domains, the number of dialogues in such domains is small compared to others, so these results might not necessarily be representative of performance in such domains. Since our agent achieves perfect metrics for the taxi (176 dialogues), hospital (48 dialogues), and police domains (32 dialogues), we did not include their results in Table 2. We observe that the conversations related to the hotel domain contain loops in 18% of simulated dialogues. The dialogue loop occurs when the simulated user keeps repeating the same specific request until it reaches the turn limit, which means the agent could not deal with such user dialogue actions related to the hotel domain. Zhu et al [24] argue that the hotel domain is more prone to dialogue loops using the Convlab simulated environment.

4.5 Dialogue Policy Training

We evaluate the improvement of transfer learning instead of using a single machine learning technique by using a simulated user to collect the effectiveness and rewards obtained of each policy. In this experiment, we compare only the policy isolated from the NLU component since it might generate errors in intent slot recognitions as stated before. The simulator sends a set of intent and slots and the policy should answer correctly during the dialogue in order to achieve the user's goal. Besides the metrics mentioned before, we measure the average reward gained by the agent by following a specific dialogue policy instance.

[1]Results extracted in September 14, 2022 from https://github.com/thu-coai/ConvLab-2

Table 2: Dialogue policy results stratified per MultiWOZ domain with the respective number of domain dialogues over 1000 simulated dialogues.

Domain	Dialogues	Succ. Rate	Inform P/R/F1	Turn (succ/all)
attraction	303	99.7	70.5/97.6/79.6	4.2/5.7
restaurant	463	98.9	63.4/82.6/69.1	9.2/9.2
hotel	394	75.1	38.1/74.3/47.6	9.6/10.6
train	348	98.2	90.7/90.7/90.7	6.2/6.2

Table 3: Results obtained from dialogue policies optimized from different methods.

Policy	Compl. Rate	Succ. Rate	Inform P/R/F1	R (avg)
π_θ^{bc+ppo}	96.4	95.2	71.1/97.5/79.6	21.9
π_θ^{bc}	95.3	85.3	69.7/96.0/78.2	20.4
π_θ^{ppo}	12.8	7.2	26.7/82.1/36.8	-31.1

Initializing the weights of π_θ by using supervising learning before applying the PPO algorithms outperforms by a large difference when compared with PPO trained from scratch and slightly outperforms the Behavioral cloning approach. The supervised learning step optimizes the dialogue by predicting the action based on a state of a separated annotated trajectory (i.e. MultiWOZ annotated dataset), which does not consider multi-turn dynamics. The success rate difference between π_θ^{bc+ppo} and π_θ^{bc} highlights the difference of both learning approaches since the second learning step rewards the agent only in episodes in which the agent achieves all goals. By contrast, the reinforcement learning step optimizes by using a reward function that gives a positive reward only whether the agent could achieve all simulated user objectives throughout the dialogue, as described in Section 4.1.

To enable a fair comparison, we train the π_θ^{ppo} policy using a higher learning rate (the same used in π_θ^{bc} policy training) than the policy π_θ^{bc+ppo}, to compensate for the pre-training included in the supervised learning phase of π_θ^{bc+ppo}. Since policy π_θ^{bc} outperforms policy π_θ^{ppo} by a substantial margin, we do not execute PPO before supervised learning. In this experiment, the results of π_θ^{bc+ppo} are better than the results in Table 1 due to the absence of the NLU component, of which outputs are error-prone.

The results of PPO trained from scratch (π_θ^{ppo}) show that our reward function might not be suitable since it yields spare rewards, making exploration challenging. Such reward sparsity [9] leads to the agent sometimes iterating over 40 dialogue turns and not receiving any positive reward in a simulated conversation. The reward sparsity problem worsens not only by the increased dialogue horizon but also by the high action dimensionality generated by

the agent dialogue policy. Similar to recent works [6, 12], we note that including annotated trajectories helps deal with this exploration problem by using a supervised step as the initial step. Results from Table 3 show that using the MultiWOZ annotated datasets to shape our policy substantially improves performance compared to training the policy from scratch.

4.6 Scrutinising The Task-Oriented BDI Agent

In this experiment, we examine the agent's mental state during failed dialogues to understand its beliefs during plan selection, and describe how a human can integrate programmed plans to assist the agent. Since learning methods can induce blind spots in our agent that might be simple for humans to deal with, we show that our agent is amenable to being scrutable and being partially improved by a human expert. Specifically, given the plan selected by the agent, we scrutinise the agent's beliefs in a specific dialogue turn describing the resulting response. In this experiment, we include a natural language justification of each answer based on its belief at the dialogue turn using a controllable sentence with each symbol filled.

Table 4 shows an example of a dialogue with a correct answer and then with an incorrect answer. In this dialogue, the agent informs the number of hotels retrieved in the knowledge base and correctly informs the slot requested by the user. However, the agent could not inform the requested information properly due to an incorrect dialogue act predicted by the dialogue policy in the third turn. The justification of its beliefs for the wrong answer sounds unreasonable since the dialogue policy maps the belief base state with an incoherent intention (i.e., did not inform the hotel name as expected by the user).

To deal with such problems during the evaluation of the agent's mental state, we introduce a manually engineered plan to illustrate a study case where humans can explicitly improve the learned dialogue policy in this experiment. Since the rule-based policy outperforms all learnable dialogue policies in Table 1, these results show that the agent's dialogue policy is amenable to having blind spots when trained with the ConvLab simulated user. In Listing 1, we illustrate a simple plan written in the AgentSpeak language to support the agent in dealing with this limitation reproduced from the learned dialogue policy. Specifically, we use a interpreter [2] of the AgentSpeak language in this experiment due to its wide use as agent-oriented programming language used to develop BDI agents [3]. In this case, we define the AgentSpeak plan[15] with the belief inform addition as the triggering event and the expected dialogue act (i.e., inform the requested slot) as the body when existing at least a single hotel in the knowledge base to avoid informing an inexistent name. We describe the alternative third turn response in Table 4 using a manually developed AgentSpeak plan comparing with the response generated by the learned policy.

Listing 1: AgentSpeak plan included in our agent to deal with the third turn in the example described in Table 4

```
+Hotel_Request(Name): Hotel_Results(N) & N>0
                      <- Hotel_Inform(Name)
```

[2]https://github.com/niklasf/python-agentspeak

Table 4: Dialogue with the agent's belief base and a justification written in natural language regarding its response. The last turn contains the learned policy response (LP) and the AgentSpeak plan response (A).

User:	**I want a hotel in east area for 4 people**
Beliefs:	**Current beliefs:** Hotel_Inform(Area), Hotel_Inform(People) **KB beliefs:** Hotel_results(4)
Response:	**There are 4 of those. One is a hotel. Others are guesthouses. What price range are you looking for?**
Justification:	Because the user the area and people, the domain knowledge base has 4 Hotels that satisfies the user preferences.
User:	**What is the phone of this hotel?**
Beliefs:	**Current beliefs:** Hotel_Request(Phone) **KB belief:** Hotel_results(4) **Past beliefs:** Hotel_Inform(Area), Hotel_Inform(People)
Response:	**The hotel phone number is 01223315702.**
Justification:	Because the user request the phone, and the user has already informed the area and people, the database has 4 hotels that satisfies the user preferences.
User:	**What is the name of this hotel?**
Beliefs:	**Current beliefs:** Hotel_Request(Name) **KB belief:** Hotel_results(4) **Past beliefs:** Hotel_Inform(Area), Hotel_Inform(People), Hotel_Request(Phone)
Response: (LP)	**What price range are you looking for?** Hotel_Request(Price)
Response: (A)	***The name is Express by holiday inn cambridge.*** *Hotel_Inform(Name)*
Justification:	Because the user request the hotel name, and the user has already informed the area and people, and the user has already requested the name the domain knowledge base has 4 Hotels that satisfies the user preferences.

4.7 Discussion

Recent end-to-end models achieve state-of-the-art results in response generation by using large annotated datasets and leveraging novel neural networks techniques. However, the use of a single black-box model obscures the agent's reasoning and its decision-making process is not scrutable. Our results show that a modular task-oriented agent brings more visibility to its decision-making process, and achieves better results in simulated multi-turn dialogues.

Our experiment shows how a human can scrutinize the agent's behavior through its explicit representation of beliefs. We formulate the justification in a natural language sentence by exploring its beliefs to help humans understand the agent's behavior during a dialogue. Furthermore, given an unexpected response and scrutinized beliefs, we show how humans can correct the agent by introducing plans in our experiments. Our experiments corroborate that explicitly programmed plans can handle specific instances of error not covered in an annotated dataset or an episode in a simulated environment. Although we introduce manually developed plans in our agent, we need to integrate programmed and learned plans to select one of them dynamically in future work.

5 RELATED WORK

Recent work on autonomous agents explore reasoning techniques to improve the communication between agents (humans or systems) regarding the explanation[4] and scrutiny[2]. Dennis and Oren [4] introduce an approach to explaining the behavior of a BDI agent using a formal dialogue with a simple BDI language named SimpleBDI. Although this approach does not handle dialogues in natural language, it deals with the formal dialogue between two participants with BDI mental attitudes and represents it via traces, which reveal each agent's motivation. In our work, we deal with natural language dialogues between a user and an agent, in which only the latter participant has its mental state explicitly represented. Caminada et al.[2] introduce an approach to demonstrate a justification to a human for actions executed by an autonomous agent by leveraging automated planning techniques. Such argument is relevant nowadays since recent works on task-oriented dialogue systems [22] often rely on pre-trained language models since they have significant performance on unseen data. This work formulates the justification by analyzing the facts contained in a knowledge base that supports each precondition of executed action in a recursive way. To evaluate the scrutiny of our agent, we apply the same idea to understand the agent's motivations to perform a specific action by exploring its beliefs perceived in dialogue with a human.

6 CONCLUSION AND FUTURE WORK

This paper develops a specialisation of the BDI architecture for task-oriented dialogue systems. The resulting architecture incorporates an agent's mental state, facilitating scrutability and the introduction of situation-specific responses. We describe the similarities between the pipeline tasks-oriented dialogue systems and the BDI mental attitudes and how we can build modular agents. Instead of relying only on programmed rules explicitly like traditional BDI agents, we explore different learning techniques and combine them to show their effectiveness in a BDI agent.

Our work opens many possible avenues for further research on BDI agents for human interaction applications. As future work, we plan to integrate explicitly programmed rules to complement the dialogue policy in order to reduce the potential blind spots in a dialogue. We need to improve the plan selection to alternate between learned and programmed plans in a more sophisticated way to cover all AgentSpeak language details. Considering that our agent is compatible with beliefs represented with logical expressions, as an important future work, we want to explore complex existing model checking approaches to improve the interpretability to our agent. Finally, we plan to integrate our agent with more formal approaches to explain the BDI agent's behavior in a dialogue context [4].

REFERENCES

[1] Michael E. Bratman. 1987. *Intention, Plans and Practical Reason.* Harvard University Press.

[2] Martin W. Caminada, Roman Kutlak, Nir Oren, and Wamberto Weber Vasconcelos. 2014. Scrutable Plan Enactment via Argumentation and Natural Language Generation. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '14)*. Richland, SC, 1625–1626.

[3] Lavindra de Silva, Felipe Meneguzzi, and Brian Logan. 2020. BDI Agent Architectures: A Survey. In *Proceedings of the Twenty-Ninth IJCAI*. 4914–4921.

[4] Louise A. Dennis and Nir Oren. 2021. Explaining BDI Agent Behaviour through Dialogue. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '21)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 429–437.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *n Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[6] Vinicius G. Goecks, Gregory M. Gremillion, Vernon J. Lawhern, John Valasek, and Nicholas R. Waytowich. 2020. Integrating Behavior Cloning and Reinforcement Learning for Improved Performance in Dense and Sparse Reward Environments. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20)*. Richland, SC, 465–473.

[7] Dilek Hakkani-Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm.. In *Proceedings of the 17th Annual Conference of the International Speech Communication Association*. 715–719.

[8] Vijay Konda and John Tsitsiklis. 1999. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller (Eds.), Vol. 12. MIT Press.

[9] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. 2022. Exploration in deep reinforcement learning: A survey. *Information Fusion* (2022).

[10] Felipe Meneguzzi and Lavindra de Silva. 2015. Planning in BDI agents: a survey of the integration of planning algorithms and agent reasoning. *The Knowledge Engineering Review* 30, 1 (2015), 1–44.

[11] D. Michie. 1990. Cognitive models from subcognitive skills. , 71-99 pages.

[12] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 6292–6299.

[13] Lin Padgham and Michael Winikoff. 2005. *Developing intelligent agent systems: A practical guide*. Vol. 13. John Wiley & Sons, Inc.

[14] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational AI: The science behind the Alexa prize. *arXiv preprint arXiv:1801.03604* (2018).

[15] Anand S Rao. 1996. AgentSpeak(L): BDI agents speak out in a logical computable language. In *European workshop on modelling autonomous agents in a multi-agent world*. Springer, 42–55.

[16] Anand S Rao and Michael P Georgeff. 1995. BDI agents: from theory to practice.. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, Vol. 95. 312–319.

[17] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Association for Computational Linguistics, Rochester, New York, 149–152.

[18] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[20] Ryuichi Takanobu, Qi Zhu, Jinchao Li, Baolin Peng, Jianfeng Gao, and Minlie Huang. 2020. Is Your Goal-Oriented Dialog Model Performing Really Well? Empirical Analysis of System-wise Evaluation. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 297–310.

[21] Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. MultiWOZ 2.2: A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020*. 109–117.

[22] Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-Oriented Dialog Systems that Consider Multiple Appropriate Responses under the Same Context. In *34th AAAI Conference on Artificial Intelligence*.

[23] Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences* 63, 10 (2020), 2011–2027.

[24] Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.