BDI Agents in Natural Language Environments

Alexandre Yukio Ichida[1], Felipe Meneguzzi[1,2], and Rafael C. Cardoso[2]

[1] Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil
alexandre.yukio@edu.pucrs.br
[2] University of Aberdeen, Aberdeen, United Kingdom
felipe.meneguzzi@abdn.ac.uk, rafael.cardoso@abdn.ac.uk

Abstract. Developing autonomous agents to deal with real-world problems is challenging, especially when developers are not necessarily specialists in artificial intelligence. This poses two key challenges regarding the interface of the programming with the developer, and the efficiency of the resulting agents. In this paper we tackle both challenges in an efficient agent architecture that leverages recent developments in natural language processing, and the intuitive folk psychology abstraction of the beliefs, desires, intentions (BDI) architecture. The resulting architecture uses existing reinforcement learning techniques to bootstrap the agent's reasoning capabilities while allowing a developer to instruct the agent more directly using natural language as its programming interface. We empirically show the efficiency gains of natural language plans over a pure machine learning approach in the ScienceWorld environment.Please note that this paper has been accepted to AAMAS 2024 main track, and has only been submitted to EMAS 2024 for discussion at the workshop. We will not publish this paper in the post-proceedings of EMAS.

Keywords: BDI agents · natural language · LLMs · reinforcement learning

1 Introduction

The increasing adoption of Artificial Intelligence (AI) algorithms in human-facing applications creates two key problems in the development of the autonomous agents that interact with humans in such applications. First, agents must understand commands and respond to them in a human-understandable way. Second, agent developers must be able to ensure the responses generated by such agents are appropriate, regardless of the availability of training data.Specifically, modern AI applications communicate to a certain degree with humans in their language in order to support their daily tasks. Such agents process the natural language information provided by humans to make their own decisions, which can yield a natural language response or an execution of sequential steps (plans). To ensure safety and avoid unintended behaviour from such applications, a scrutable mental model is essential for an autonomous agent to provide the transparency of the agent's behaviour and explain its decisions [25].

Handling natural language is crucial for autonomous agents to communicate and cooperate with humans. An agent should reason over the natural language information to understand the circumstances of human problems and take an action that should be discernible to humans in order to support them. Processing human language is a complex task for computers since information encoded by natural language is unstructured and prone to ambiguity [15]. Recent deep learning approaches for natural language processing, such as Pre-Trained Language Models (PTLM), achieve very high accuracy in text classification and generation tasks [26]. However, approaches that rely exclusively on a single PTLM have limitations in reasoning tasks such as common-sense planning [33] and logical consistency [23] over natural language information. In fact, the PTLM reasoning process is opaque [29] since such approaches rely entirely on black-box models, and hence, understanding the agent's decisions remains a difficult task. An explicit mental model representation that describes the agent behaviour is essential to understanding such reasoning limitations and dealing with them.

Traditional approaches to AI 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 [7,31] introduces a conceptual framework to implement autonomous agents composed of beliefs, desires, and intentions. We argue that incorporating plans designed by humans is instrumental for the development of more controllable autonomous agents, instructing them to avoid unintended behaviour. Natural language plans allow humans to explicitly incorporate prior information which, in contrast to pure machine learning agent approaches, does not necessarily need to retrain the agent or adjust training data. Instructing the sequence of steps for an agent to perform a task does not require a training phase where the agent needs to explore or exploit states in the environment (e.g., ϵ-greedy reinforcement learning algorithms [5]). Instead, the agent can simply executes predefined plans supplied by a human.

This paper introduces NatBDI, a new class of agent architecture that uses the BDI reasoning cycle with components driven by natural language processing. We leverage the advantages of modern machine learning models in natural language (e.g., pretrained language models) by using them as black-box components within the BDI model. Our contributions are threefold. First, we describe the unstructured data from observations of human activities as agent's beliefs written in natural language and generate the natural language plan library of the agent. Second, we develop an agent interpreter following the BDI mental model that uses natural language information in its reasoning back-end. This back-end uses natural language inference to find the entailment relations between the belief base and the context of the plans in the plan library during plan selection. Finally, we rely on a fallback policy using black-box reinforcement learning architectures to help our agents to act autonomously in cases where the plan-selection mechanism finds no candidate (i.e., applicable) plans. We use simulated textual environments from ScienceWorld [34] to validate and evaluate the natural language understanding and decision-making capabilities of NatBDI. Our results indicate that even adding a small amount of natural language plans when combined with the modified BDI reasoning cycle for natural language environments can dramatically increase the reasoning performance of the agent over pure machine learning agents.

2 Background

This section briefly introduces the necessary background on the topics used in our approach. We start with an overview of how the BDI model works and define a simple BDI agent interpreter. Then, we discuss natural language inference and its relevance in defining entailment relations, a crucial feature used in our approach to allow BDI agents to operate in natural language environments. Finally, we describe the ScienceWorld benchmark environment, which we use as an example throughout the paper as well as in the experiments.

2.1 BDI Model

The BDI model is a framework to develop autonomous rational agents in terms of components that correspond to mental attitudes [31]. Inspired by Bratman's philosophical work [6], the BDI framework introduces a practical reasoning approach that consists of three basic mental attitudes: 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. Desires represent states of affairs that the agent aims to achieve to satisfy its design goals. Intentions represent the agent's commitment to achieve a specific subset of its desires, acting as a filter for the agent's reasoning to account for practical decision-making made by agents with physical limits to its computing power. As such, it serves not only as a concrete explanation for how real humans (with limited brain power) make decisions, but also as an overall blueprint to design computational agents running with the limitations of a physical computer. The intention component is a structure that often consists of a set of instantiated plans adopted by the agent in order to achieve a subset of its desires. Such plans are a sequence of steps that the agent executes to achieve a specific desire. Most practical agent architectures use a plan library that includes either full plans available to the agent, or planning rules that allow the agent to generate plans during runtime [28]. The BDI model is arguably the most widely used model to implement rule-based autonomous agents in the agent-oriented programming paradigm [8].

Implementations of the BDI architecture often encode the agent's behaviour as plan-rules to instruct it on achieving particular (implicit) goals given specific context conditions [22]. In this context, plans represent a sequence of actions the agent should perform given a set of conditions (i.e., the context of the plan) entailed by the agent's belief base, which are usually developed manually by humans. Such terminology is useful in developing and debugging autonomous agents in a variety of domains [24] since this architecture describes information about the agent's beliefs. Algorithm 1 illustrates a simple reasoning cycle for a

Algorithm 1 A simple BDI Agent Interpreter.

```
1: procedure agentInterpreter($\mathcal{E}, \mathcal{B}, \mathcal{L}, \mathcal{I}$)
2:     while true do
3:         $\mathcal{E} \leftarrow$ updateEvents($\mathcal{E}$)
4:         $\mathcal{B} \leftarrow$ updateBeliefs($\mathcal{E}, \mathcal{B}$)
5:         $\mathcal{I} \leftarrow$ selectPlans($\mathcal{E}, \mathcal{B}, \mathcal{L}, \mathcal{I}$)
6:         $\mathcal{E} \leftarrow$ executeIntention($\mathcal{I}, \mathcal{E}$)
```

BDI agent interpreter [31] that includes four steps. Line 3 collects the events the agent observes from the environment. The agent uses these events to update its beliefs in Line 4. Given changes in its belief base, the agent proceeds to select plans in Line 5 from its plan library based on the current events, beliefs, and intentions. Plan selection also includes instantiation of intention structures that help the agent keep track of its progress. Finally, in Line 6 the agent executes intentions by selecting one of the instantiated intentions and executing the associated actions in the environment, which then leads to the generation of new events, closing the agent-environment loop.

2.2 Natural Language Inference

Automated reasoning and inference are essential topics in AI in general, and in autonomous agents in particular [11]. Natural Language Inference (NLI) is a widely-studied natural language processing task that is concerned with determining the inferential relation between a premise p and a hypothesis h [20]. In NLI, both p and h are sentences written in natural language. The challenge of this task differs from formal deduction in logic since natural language deals with informal reasoning [20]. The emphasis of NLI is on aspects of natural language such as lexical semantic knowledge and dealing with the variability of linguistic expression. Unlike the crisp relations between logical sentences in formal logic, NLI can define the logical relation in multiple ways: entailment, neutral and contradiction. Given a pair of premise-hypothesis p and h, the *entailment* relation occurs when h can be inferred from p [20]. When h infers the negation of p, the pair results in a *contradiction.* Otherwise, if none of these relations can be inferred, the relation of p and h is *neutral.* In this paper, NLI serves as the critical connection between the traditionally logic-based machinery of the BDI model and the unstructured and often ambiguous world of natural language data.

Given the premise sentence "Several airlines polled saw costs grow more than expected, even after adjusting for inflation" and the hypothesis "Some companies in the poll reported cost increases" In the NLI context, this example is a valid entailment inference because any person that interprets p would likely accept that h implies in the information of p. Even though it is a valid NLI classification, h is not a strict logical consequence of p due to the fact that p informs that airline companies **saw** the growth of the cost, not necessarily **reporting** the growth of the cost. This example reflects the informal reasoning of the task definition which relates to the ambiguity found in natural language [20].

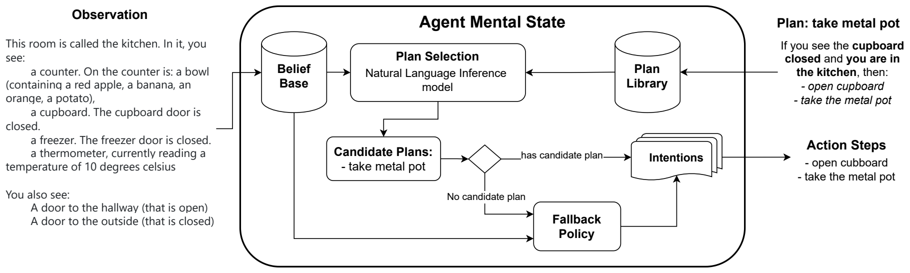
Observation
This room is called the kitchen. In it, you see:
a counter. On the counter is: a bowl (containing a red apple, a banana, an orange, a potato),
a cupboard. The cupboard door is closed.
a freezer. The freezer door is closed.
a thermometer, currently reading a temperature of 10 degrees celsius
You also see:
A door to the hallway (that is open)
A door to the outside (that is closed)
Agent Mental State
Belief Base
Plan Selection
Natural Language Inference model
Plan Library
Candidate Plans:
- take metal pot
has candidate plan
No candidate plan
Intentions
Fallback Policy
Plan: take metal pot
If you see the cupboard closed and you are in the kitchen, then:
- open cupboard
- take the metal pot
Action Steps
- open cubboard
- take the metal pot

Fig. 1. Diagram illustrating the NatBDI architecture to handle and actuate over natural language environments.

2.3 ScienceWorld Text Environment

ScienceWorld is an interactive textual environment that simulates engines for thermodynamics, electrical circuits, matter and chemistry reactions, and biological processes at the level of a standard elementary school science curriculum [34]. Such challenge aims to evaluate the agent's reasoning about transitioning between locations and object interaction. It also aims to test the agent's understanding about combining distinct objects considering their state of matter. Current Large Language Models (LLM) can produce responses in question answering tasks as an information-retrieval system. However, Wang et al. [34] show that they still have limitations regarding reasoning about scientific knowledge to act or to plan to solve a task. The ScienceWorld environment evaluates the agent's capacity to use declarative scientific knowledge to act or plan in order to solve tasks that humans can perform with ease (e.g., melting ice). It includes 30 benchmarks that are split into 10 topics such as the change of state (boiling, melting, freezing), taking measurements (thermometer, boiling point), classification (find a non-living thing, find a plant), etc. For example, in changing matter state, one of the benchmarks is about boiling. In order to boil the water, the agent should walk around the map to find a metal pot, search for water, and use a stove. Here, the agent should use prior knowledge about an object's state to choose a metallic cup instead of a wooden one since it will contact fire. Furthermore, the agent should use the thermometer object to be aware of the boiling point of water since waiting too long can turn the water into vapour.

3 Agent Architecture Overview

This section describes the NatBDI architecture for natural language environments. We focus on the three key components of the BDI architecture and how we link them to a knowledge representation using pretrained language models. Section 3.1 describes a novel reasoning cycle to select and execute plans written in natural language using NLI to connect plan descriptions to the knowledge stored in the belief base. Section 3.2 describes belief formulation from natural language observations. Key to the operation of NatBDI is the way in which a designer writes plans in natural language for NatBDI's plan library, which we describe in Section 3.3. Finally, we describe a fallback mechanism in Section 3.5 that allows an agent to select plans when the plan library fails to provide applicable plans.

3.1 Reasoning Cycle

At each interaction with the environment, the agent perceives the state as a natural language description of the agent's current location. The agent stores such natural language descriptions of the environment in its belief base, which it then uses to make inferences over its course of action. Once the agent updates its belief base, it selects plans from its plan library, which can generate plans in one of two ways. Either the plan library contains human-designed plans to react to new perceptions, which we call *plan-rules*, or the agent resorts to *fallback plans* that the agent learns using reinforcement learning. Plan-rules react to user-defined context conditions that the agent checks using natural language inference. Figure 1 summarises the components of the agent reasoning cycle and their interaction.

We leverage the BDI event-driven approach to develop our agent reasoning cycle organised in the following steps. First, the agent receives the task description representing the main goal similar to the goal-addition event. Second, the agent then search in its plan library a plan that has the received goal as the triggering event and compares whether the current belief base entails the plan context. In this phase, the agent retrieves plan options from the plan library based on its current belief base. Given a plan selected by the agent, the interpreter analyses the steps contained in the plan body, which can be another goal-addition event or an action to be executed. In cases where no candidate plan is available, or the agent did not achieve the main goal, the reasoning cycle generates a failure in the decomposition process. Finally, at the end of Algorithm 1, if such execution returns a failure to the main reasoning cycle, the agent then starts to use fallback plans to attempt to deal with the failure.

3.2 Belief Base

Our approach to represent the natural language belief base follows the traditional BDI model, which organises the information perceived into literals in its mental model, except that our agent receives observations written in natural language instead of symbols. Since the belief base comprises simple sentences in natural language, it helps humans scrutinise the agent's mental state and understand the behaviour of the agent.

In our work, the agent maintains its belief base as a list of natural language sentences describing its observations about the current environment state. Observations consist of perceptions from the environment described in natural language. Specifically, our approach splits the full textual observation into distinct phrases to associate each sentence to a particular belief to be added in the belief base. Such sentences describe environment effects perceived from previous actions, objects seen by the agent in the current location, and items carried in the agent's inventory (these are all common observations in the ScienceWorld text environment). For instance, the textual observation described in Figure 1 is a paragraph that informs multiple facts of the current state of the environment. The agent represents each sentence as a single belief through splitting the text into a list of sentences. Given that, "You see a freezer" and "You see a thermomether" represent two distinct beliefs. In contrast with traditional BDI approaches where the agent checks for belief additions or deletions in the belief base, our agent simply rewrites the old belief base state with the new textual information perceived after performing an action. The belief base integrity relies on how the textual environment represents the effects of performed actions. Since ScienceWorld provides full state descriptions, our agent updates its belief base by overwriting old beliefs with ones perceived to avoid inconsistencies caused by action effects.

3.3 Natural Language Plan Library

Besides interacting with a world described in natural language, our agent architecture relies on a natural language interface for agent developers to encode plan-rules using natural language in order to facilitate the plan development for humans who are non-expert in an agent programming language (e.g., AgentSpeak [28]). We represent natural language plans in a controlled natural language [13] that contains clearly described conditions and beliefs. A controlled natural language is a subset of a natural language more amenable to automated processing, which we use to allow programmers to intersperse unrestricted natural language within the structure of a plan rule. In contrast with approaches that translate the controlled natural language plans into a symbolic representation (i.e., Prolog clauses) [13], our architecture reasons directly over natural language plan-rules during plan selection.

While NatBDI uses a controlled natural language in its definition, we interpret this language as following a similar structure to AgentSpeak plan-rules. Thus, each plan rule consists of a statement for the intended goal, a statement of the plan context, and a set of sentences defining the plan body. The goal is a sentence describing the task that the agent intends to perform. The plan context consists in a set of natural language sentences linked with the word "AND" to represent the logical conjunction between such statements. More formally, plans in our controlled natural language follow the template in Listing 1.1:

Listing 1.1. Template for a plan written in NatBDI controlled natural language.

```
IF <goal statement>
CONSIDERING <plan context statements>
THEN:
<plan body>
```

In order to accept hierarchical plans, the agent interpreter accepts sentences representing actions or subgoals in the plan body. Here subgoals work similarly to goal addition events in AgentSpeak. In the plan body section, we include the keyword "**PLAN TO**" to distinguish sentences that encode a goal addition, which is analogous to a goal addition event, with sentences describing actions. Much like in AgentSpeak, an agent keeps track of subgoals in a stack data structure. As the agent adopts new goals, they are stacked, and as it achieves them, they are unstacked. Such keyword helps the agent interpreter disambiguate between actions sent to the environment, and the internal reasoning for recursive subgoals. Actions consist of an imperative sentence that describes what the agent should perform in the environment.

For example, the natural language plans described in Listing 1.2 shows how a human can explicitly instruct the agent to achieve particular tasks. In the first plan, a human knows that the metal cup is in the cupboard which is initially closed and instructs the agent on how to obtain the metal cup using a natural language plan. The triggering event is the goal "get the metal pot" and it should be considered as plan candidate if the context of the plan is true, that is, if the agent is in the kitchen and it is seeing a closed cupboard. The plan body consists of two actions written in natural language instructions that will be interpreted by the textual environment. First the agent should open the cupboard and then it should take the metal pot. The second plan triggers when the goal is to melt water. This illustrates part of a plan that requires a subgoal (get the metal pot).

Listing 1.2. Plans in natural language to pick the metal pot and melt water in ScienceWorld.

```
IF your task is to get the metal pot
CONSIDERING you are in the kitchen
        AND you see the cupboard closed
THEN:
open the cupboard,
take the metal pot

IF your task is to melt water
THEN:
PLAN TO get the metal pot
pick up thermometer
...
```

3.4 Entailment with Natural Language Beliefs

Most implementations of the BDI architecture select plans based on a context condition that, when entailed by the agent's belief base, trigger plan adoption. This creates a filter for options to be executed by the agent afterwards [22]. Concretely, if the belief base logically entails the conditions described in a plan, then the agent commits to executing its steps within an intention structure. Since the agent deals with natural language information, computing entailment within natural language becomes a key challenge. We leverage an NLI model grounded by machine learning to emulate the entailment operation over natural language information.

In the BDI model, the plan-selection mechanism assumes that the entailment inference consists of a Boolean value indicating whether a plan is candidate/applicable or not. Recent approaches develop natural language inference models as a three-way classification method generating the following three logical relations: entailment, contradiction, and neutral [36]. Since our main objective is to infer whether the beliefs entail a specific plan context, we unify the contradiction and neutral classes as a non-entailment relation. Thus, we use the natural language inference model as a binary classifier (i.e., returning a Boolean signal) under a closed-world assumption.

In practice, to infer whether the belief base entails a plan context, we need to compare if each sentence in the plan context has at least one belief that entails it. Specifically, the agent needs to compare every belief in belief base with all plan contexts to decide if such plan should be selected. Since beliefs and the plan context consist of multiple sentences, NatBDI processes the entailment inference between these two structures by creating the Cartesian product between both sentence sets. Formally, given two set of sentences representing the belief base $\mathcal{B}$ and $\mathcal{C}$ respectively, we create a matrix represented as $\mathcal{C} \times \mathcal{B}$ in Equation 1. For each belief $b \in \mathcal{B}$ and plan context $c \in \mathcal{C}$ pair contained in the $\mathcal{C} \times \mathcal{B}$, we employ the natural language inference represented as nli function in Equation 2 resulting in a Boolean matrix $E_{i,j}$. Matrix $E_{i,j}$ contains all inference results for each i-th plan context and j-th belief.

$$M_{i,j} = \mathcal{C} \times \mathcal{B} = \{(c_i, b_j) \mid c \in \mathcal{C} \land b \in \mathcal{B}\} \tag{1}$$

$$E_{i,j} = \{nli(c_i, b_j) \mid (c_i, b_j) \in M_{i,j}\} \tag{2}$$

To check whether a plan is a candidate to be selected by the agent, we apply the disjunction given the Boolean values in $E_{i,j}$ for each i-th context with all j-th beliefs. Given each Boolean generated by the disjunction, we apply the conjunction for each c resulting in a single Boolean value. Thus, we define entailment of a plan context from the belief base in NatBDI as in Equation 3.

$$\mathcal{B} \models \mathcal{C} \doteq \bigwedge_{c_i \in \mathcal{C}} c_i \bigvee_{b_j \in \mathcal{B}} b_j \tag{3}$$

For example, consider a state with a belief base that contains the following sentences: "this room is called the kitchen" and "you see a cupboard, the cupboard door is closed". An inference between such belief base and the plan with context composed of the sentences "you are in the kitchen" and "you see a closed cupboard" works as follows. First, we compute the values of matrix $E_{i,j}$ with i-th row representing context sentences and j-th column representing belief sentences as follows: $\left[\begin{smallmatrix} T & F \\ F & T \end{smallmatrix}\right]$. Belief "this room is called the kitchen" entails context "you are in the kitchen" while "you see a cupboard, the cupboard door is closed" entails "you see a closed cupboard". The first disjunction operation between all beliefs

Algorithm 2 Fallback policy plan formulation.

```
1: procedure fallbackPolicy($\mathcal{E}, \mathcal{B}, \mathcal{L}_{FB}, \mathcal{I}$)
2:     $bd \leftarrow \mathcal{L}_{FB}(\mathcal{E}, \mathcal{B})$
3:     $\mathcal{I} \leftarrow \mathcal{I} \cup \langle \mathcal{E}_0, bd \rangle$
4:     return $\mathcal{I}$
```

with each context results in the following matrix: $\begin{bmatrix} T \\ T \end{bmatrix}$ and, consequently, the conjunction operation results in the scalar Boolean $\begin{bmatrix} T \end{bmatrix}$, which is the final result of the entailment inference, and therefore we can conclude that the information in the belief base entails the plan context. Consequently, this is a candidate plan.

3.5 Fallback Policy

Previous research on dealing with BDI plan selection as a learning problem includes learning context conditions [32], as well as modelling a BDI agent as one interacting in a Partially Observable Markov Decision Process (POMDP) [30]. A POMDP is a model of stochastic environment in which the agent perceives states indirectly via observations. Both models work with belief concepts to represent the agent state since POMDP agents store the states in a set of belief states while BDI agents use their belief base component for the same role. Such work shows that the BDI procedure of selecting and executing a plan can be implemented as a POMDP state estimator. Given such correspondence, we integrate a mechanism for plan selection trained by reinforcement learning to generate a plan when there is no candidate plan in plan library (i.e., the natural language inference fails to produce candidates).

To produce plans consistent with the *plan-rules* format we use for the plan library, we train a fallback policy that can generate, for each possible context, a plan body consisting of a single action. Thus, at each turn the agent either runs steps from a human-defined plan-rule, or an action from the fallback policy. Since the fallback policy might contain ineffective plans due to training limitations, we keep track of the number of times the agent responds with the same plan for the same event, which we control with an l parameter. Given a predefined step limit l, the agent uses the policy $\mathcal{L}_{FB}$ with the current events $\mathcal{E}$ and belief base $\mathcal{B}$ to predict which action should be executed, and create a single-action plan body resulting in a new intention. Algorithm 2 details how we use the fallback policy to generate new intentions.

4 Evaluation

In this section, we describe our experiments to evaluate NatBDI in reasoning tasks within a textual environment. We conduct the experiments using a Python implementation of the architecture[3] on the ScienceWorld environment.

[3] Available at https://github.com/yukioichida/nat-bdi

First, we detail the implementation settings to execute our experiments. Second, we compare the effects of using plan-rules to help agents in reasoning tasks considering prior knowledge introduced by humans through plans written in natural language. Finally, we show how different NLI models affect the inference performance in sentence pairs.

4.1 Experiment Setup

Our experiments use two different types of tasks based on the performance of current approaches in the ScienceWorld environment (Section 2.3) to evaluate NatBDI. These tasks consist of the *melt* task, which requires the agent to melt an element, and the *find-non-living-thing* task, which requires the agent to take a non-living object and put it into a container detailed in the task description. All current approaches [27,38,1,9,14,34] perform poorly on the "melt" task since it requires more sophisticated reasoning than "find-non-living-thing". The first usually requires the agent to find and take the target element, put it into the appropriate container (e.g., metal pot) and find a device to melt the element (e.g., stove or blast furnace). By contrast, the second task only requires the agent to take a non-living thing and put it into a container, requiring simpler reasoning skills. We choose such tasks to evaluate the agent's performance in a simple task, which even using a reinforcement learning agent can result in a reasonable score, and a hard task to show the gains of including natural language plans to deal with the limitation of current machine learning techniques. Each task variation contains different environment settings, which varies from having objects placed in different locations to having different objects in task description (i.e., "melt water", "melt mercury", etc).

For each experiment, we initialise the plan library by analysing annotated sequence of steps provided by the ScienceWorld authors in both tasks. Since the agent can begin a task in different locations, we generate navigation plan-rules automatically by running multiple navigation tasks where we use heuristic search to find nearly optimal trajectories, and collecting all trajectories found into plan-rules. In these navigation plan-rules, we use the location at turn $t-1$ as plan context and include in the plan body a move action to the next location at turn t. We use the task description provided by the environment as the main goal, which is analogous to the initial goal-addition event in AgentSpeak.

In order to evaluate our approach, we integrate into NatBDI existing LLMs pretrained with NLI datasets. For the experiments described in Section 4.2 we use the *roberta-large* language model [18] trained using the MultiNLI [36] dataset provided by the HuggingFace repository [37]. We explore the use of other LLMs such as Bert [12] and MiniLM [35] to evaluate the effects of using smaller language models in Section 4.3. We execute all the inference steps described in Section 3.4 in a batch approach to leverage the computational resources using a single NVIDIA RTX 3060 GPU.

Our fallback policy uses Deep Reinforcement Relevance Network (DRRN) [14], the current state-of-the-art for ScienceWorld. We trained a DRRN policy for each task following the Wang et al. method [34] to use it as a fallback policy when

Table 1. Comparison of our natural language BDI agent in two tasks in the ScienceWorld environment with different plan library sizes. We show the average scores obtained and the average number of actions performed in each phase out of all task variations (Var). The task "find" represents the "find-non-living-thing" task The bold font identifies which approach (BDI or DRRN) contributed more to the total score.

Task	**Var**	**Episodes**	**Number plan-rules**	**Score (Total)**	**Score (BDI)**	**Score (DRRN)**	**Number BDI actions**	**Number RL actions**
find	75	242	0	0.66	0.00	**0.66**	0.00	50.00
			8	0.75	0.30	**0.45**	3.33	38.00
			15	0.84	**0.58**	0.26	6.25	24.00
			23	0.91	**0.79**	0.12	7.64	13.33
			30	0.98	**0.98**	0.00	9.19	4.00
melt	9	457	0	0.03	0.00	**0.03**	0.00	50.00
			4	0.14	**0.11**	0.03	5.11	44.44
			7	0.36	**0.34**	0.02	10.89	33.33
			10	0.57	**0.56**	0.01	17.11	22.22
			13	0.67	**0.67**	0.00	20.89	16.67

no candidate plan-rule can be found. To avoid infinite execution of ineffective plans from a poorly learned policy, whenever we select a fallback plan, we keep track of the number of times the agent repeatedly recurs to the fallback policy for the same event. If the agent keeps recurring to the fallback policy over a fixed number of times, we deem the intention to have failed. In our experiments we define a 50 turn limit for the fallback policy phase.

4.2 Experiment Results

In this section, we describe the scores that our natural language BDI agent obtains. First, we measure the BDI agent performance when varying the number of plan-rules in the agent's plan library for each of the two tasks we selected from the ScienceWorld environment. We show the effects in the overall score and its progress by adding natural language plans designed by humans. Second, we evaluate the trade-off between plan-rules and the DRRN fallback policy, and the impact that the fallback policy can have depending on the size and quality of the plan-rules library. Finally, we evaluate using a DRRN by itself, that is, an agent with no plan-rules that relies only on the fallback policy.

Table 1 shows the metrics collected when executing the agent in multiple variations for each task, 75 variations for the first task and 9 for the second. The number of episodes seen in DRRN training for "find-non-living-thing" and "melt" are 242 and 457 respectively. The total score represents the accumulated reward received by the agent averaged over a number of variations of the task in the same environment, which we then break down the score obtained from using plan-rules and from using fallback policy. The agent achieves a large improvement in score by adding natural language plan-rules when compared to the score obtained by using only the DRRN policy (i.e., setup with 0 plan-rules). Even a small number of plan-rules leads the BDI agent to outperform the DRRN alone by a large margin in the hardest task (melt). Introducing natural language plan-rules also leads to fewer actions to achieve a goal as a result of our natural language plan-rules encoding nearly optimal trajectories in their plan body to solve the task.

Our experiment shows that DRRN policy by itself could not deal with important aspects of the environment or reason over the task in both tasks, which corroborates the results presented in [34]. Particularly, in the "melt" task, the DRRN-only agent receives a score greater than zero due to random move actions that send the agent to the exact location of the target element. Although the trained policy could achieve better scores in the "find-non-living-thing", its performance is due to the environment which rewards the agent with a 0.5 score by solely taking a non-living-object, which is more predominantly featured in the environment than living objects.

The BDI agent achieves good scores relying on manually designed natural language plan-rules, but it does not obtain a perfect score since we use a limited number of plan-rules, which do not cover all possible variations of the tasks. This is a more realistic setting, as we would not expect the developer to be able to create all possible plan-rules. For example, in the task "melt", there are variations in which some objects are not working (i.e., broken stove), which requires the agent to plan more dynamically to find an alternative plan at runtime. Our expectation here was that the fallback policy could be useful, however, due to the low performance of DRRN in the "melt" task this was not the case. If future reinforcement learning algorithms improve the performance over the state-of-the-art, then they can be used as a fallback policy in NatBDI to achieve even better performance.

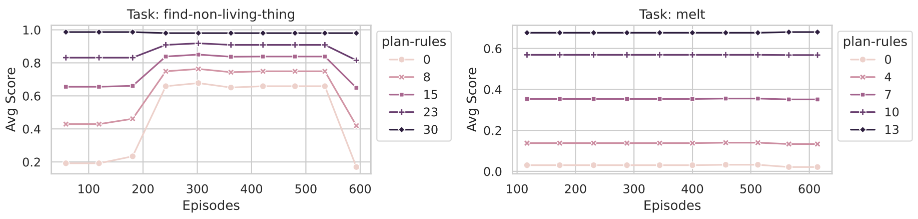

Fig. 2. Scores per episodes when scaling the number of plan-rules.

Figure 2 illustrates the score evolution throughout training episodes, given a specific number of natural language plan-rules. In this experiment, we generate multiple DRRN-trained policies, distinguishing them by the number of trained episodes to evaluate the training efficiency and compare it within NatBDI as a fallback policy component. Regarding the "find-non-living-thing" task, the policy efficiency improves throughout the training phase, but it is prone to overfitting. This is apparent in the dip in performance shown in the results with more training episodes. In the "melt" task, the number of training episodes do not seem

Table 2. Results of using different LLMs for NLI. The following columns describe them: model size (*Params*); accuracy on MultiNLI matched test set (*MNLI*-m); score obtained using NatBDI; average number of *actions* performed, *errors* raised and plan-rules (*Plans*) used; lexical overlap computed on entailment pairs (*LO(E)*); average word number in belief (*|B|*) and context (*|C|*) sentences; and total sentence pairs processed. The task "find" represents the "find-non-living-thing" task. We highlight the best scores in bold font.

Model	Params	MNLI-m	Task	Score	Actions	Errors	Plans	LO(E)	|B|	|C|	Pairs
MiniLM (L6)	22M	82.2	find	0.69	7.65	0.37	2.57	0.64	7.84	4.20	1076
			melt	0.23	8.78	1.00	3.59	1.34	10.96	4.83	691
Bert (base)	110M	84.6	find	0.84	9.61	0.20	2.72	0.60	8.40	4.16	1075
			melt	0.33	11.44	0.67	3.56	1.16	11.12	4.80	690
Roberta (large)	355M	90.8	find	**0.98**	9.19	0.08	2.84	0.40	7.30	4.19	1076
			melt	**0.67**	20.89	0.33	5.67	1.21	11.06	5.25	790

to affect the performance of DRRN. This makes sense since we have seen in previous results that DRRN performs very poorly in this task, and therefore increasing the number of training episodes here has no discernible effect.

4.3 Natural Language Inference Model Analysis

As a final set of experiments, we measure the impact of using different LLM implementations for NLI in NatBDI. We evaluate three LLMs with distinct number of parameters fine-tuned with the MultiNLI dataset to analyse the effects of the model size in ScienceWorld tasks. We organise the NLI model experiment in the following points: First, we provide details about each LLM by describing their sizes and performance on NLI dataset used in its pretraining process. Second, we detail the scores obtained in each ScienceWorld task for each LLM. Finally, we discuss the use of sentences processed using lexical overlap to measure the contrast between beliefs and plan context sentences. Table 2 shows the performance obtained by NatBDI using all plan-rules for each task, that is, with the configuration described in the last row of each task from Table 1.

NatBDI executes plans hierarchically, following the traditional reasoning cycle of BDI agents. Therefore, errors encountered during plan decomposition will abort the plan-selection process, which turns later subgoals unreachable. Incorrect inferences between beliefs and plan contexts results in incorrect plan selection, which leads the agent to select irrelevant plans or to an early stop due to not finding any candidate plan. In fact, the *roberta-large* results score higher than smaller language models such as *bert-base* and *MiniLM*, in both the MultiNLI matched test set and in the two tasks from the ScienceWorld text environment. This indicates that larger models can provide better NLI, and consequently, lead to more successful plan selection and execution when used for NLI in NatBDI. In contrast, the use of *bert-base* and *MiniLM* models result in fewer inferences since such models could not progress as well as *roberta-large* in the plan decomposition, which translates to fewer plans executed in general.

To measure the difficulty in inferring entailment between the belief base and the plan context, we compute the lexical overlap between sentences used as premise and hypothesis to count the number of identical words between the pair. Given a sentence pair consisting of a belief and a context, we compute the number of words contained in beliefs that are absent in the plan context. In cases where lexical overlap is high between the premise and the hypothesis, the inference tends to easily infer entailment relation since both sentences are similar and may express the same idea. For example, it is trivial for an NLI model to infer entailment between the belief-context pair "you see a pot" and "you see a container" due to the large lexical overlap (i.e., 3 words). Hence, in such cases, sophisticated language models exploit shallow syntactic heuristics to infer logical entailment between sentences [21]. Our results show that the amount of lexical overlap is low when comparing to the average word number in both sentences considering our manually designed plan-rules. The average lexical overlap in entailment sentence pairs is lower than the average number of plan context words since most beliefs contain more words.

5 Related Work

This section covers related work on NLI, the use of natural language in BDI agents, and natural language representation of beliefs.

There is a wide-range of pretrained NLI models openly available that are trained on well-known datasets such as SNLI [4] and MNLI [36]. Both datasets consist of premises and hypotheses represented exclusively at the sentence level. Regardless of the assumptions of these common datasets, state-of-the-art approaches to NLI fine tune pretrained language models that process a diverse range of text lengths in the pretraining task. Clark et al. [10] introduce a model that leverages pretrained language models to make inferences over facts consisting of multiple sentences. In this case, the model informs whether a statement is true given a set of facts and rules described in natural language. While traditional NLI models classifies pairs of sentences into either having entailment/-contradiction or an undetermined relation (neutral), we enforce the closed-world assumption. Thus, whenever we fail to find entailment, we assume the context query is false.

Few approaches combine BDI agents and natural language, such as in [17], and more recently in [19] and in [16]. In [17], the authors propose using *sEnglish* (system-English, a controlled natural language) to provide a natural language environment for programming BDI agents to be deployed in robotic applications. Their approach relies on ontologies and translations from *sEnglish* to agent programs in the Jason [3] implementation of AgentSpeak. In [19], they combine natural language processing for translating natural language sentences into a logical form, first-order logic as a cognitive reasoner, and BDI agent as a reactive reasoner in a cognitive chatbot framework called AD-Caspar. Neither [17] nor [19] use LLMs or exploit natural language inference for plan selection as we do in NatBDI.

In [16], they leverage component correspondences between task-oriented dialogue systems and BDI architecture to develop a BDI conversational agent. In contrast with our approach, which infers logical entailment directly over natural language information through LLMs, they use a function to translate utterances into symbolic beliefs.

As an agent interacts with the environment over time, its belief base can grow arbitrarily large with newly perceived information. However, since the observations are natural language information, it is difficult to detect whether a belief overwrites information about previous states. Atzeni's work [2] applies case-based reasoning in textual environments by memorising past problems and their solutions to solve new problems. Similar to this approach, in future work we envision a memory component within the belief base to store temporal information perceived by the agent. This component must deal with the scalability issue, since the belief base can grow, at best, linearly throughout time. Since our agent stores vectorised information of natural language beliefs, summarising text by pruning irrelevant past information can alleviate problems in the expansion of the belief base.

6 Conclusion

In this paper, we develop the seminal approach for an entire class of BDI-based agent architectures that use machine learning components to deal with natural language information. Combining a natural language interface and reasoning capabilities with the folk psychology abstraction of mental states in the BDI model provides the dual benefit of improving human understanding of the underlying machine learning models and the agent's handling of noisy information. We leverage the BDI model, a well-known approach to agent-oriented programming, to develop agents with mental states amenable to being scrutinised. The natural language plan library allows humans to create plans to customise the agent's behaviour and helps avoid unintended conduct. Unlike modern PTLM approaches that solely use a black-box model approach to reason over natural language, we explored the BDI architecture to uncover the agent's mental state to understand its behaviours. Our empirical results show that even a few manually designed plan-rules in natural language can substantially improve performance of agents working in textual environments. This is especially true for tasks that require longer horizon reasoning or complex causality.

Future work includes a number of extensions. First, our experiments currently comprise a subset of the ScienceWorld benchmarks, given the need to develop plans for each task. Thus, we will expand our experimentation to the entire ScienceWorld suite, as well as to other textual environments such as Jericho [14]. Second, given the fast pace of development in reinforcement learning, we aim to improve fallback policies. Finally, while fallback policies help mitigate the need to develop a plan library for every single situation faced by the agent, our key direction for future work lies on learning plan-rules from data. This should allow human developers to co-design an agent's plan library in an efficient and transparent way.

References

1. Ammanabrolu, P., Hausknecht, M.: Graph constrained reinforcement learning for natural language action spaces. In: International Conference on Learning Representations (2020)
2. Atzeni, M., Dhuliawala, S.Z., Murugesan, K., Sachan, M.: Case-based reasoning for better generalization in textual reinforcement learning. In: International Conference on Learning Representations (2021)
3. Bordini, R.H., Wooldridge, M., Hübner, J.F.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley & Sons, Chichester, UK (2007)
4. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 632–642. Association for Computational Linguistics (Sep 2015)
5. Brafman, R.I., Tennenholtz, M.: R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research **3**, 213–231 (2002)
6. Bratman, M.E.: Two faces of intention. Philosophical Review **93**, 375–405 (1984)
7. Bratman, M.E., Israel, D.J., Pollack, M.E.: Plans and resource-bounded practical reasoning. Computational Intelligence **4**(4), 349–355 (1988)
8. Cardoso, R.C., Ferrando, A.: A review of agent-based programming for multi-agent systems. Computers **10**(2), 16 (2021)
9. Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. In: Advances in Neural Information Processing Systems. vol. 34, pp. 15084–15097 (2021)
10. Clark, P., Tafjord, O., Richardson, K.: Transformers as soft reasoners over language. In: Proceedings of the Twenty-Ninth International Joint Conferences on Artificial Intelligence. pp. 3882–3890 (2020)
11. Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence **42**(2-3), 213–261 (1990)
12. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: 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). pp. 4171–4186. Association for Computational Linguistics (Jun 2019)
13. Fuchs, N.E., Schwitter, R.: Specifying logic programs in controlled natural language. arXiv (arXiv:cmp-lg/9507009) (1995)
14. Hausknecht, M., Ammanabrolu, P., Côté, M.A., Yuan, X.: Interactive fiction games: A colossal adventure. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 7903–7910 (2020)
15. Hirschberg, J., Manning, C.D.: Advances in natural language processing. Science **349**(6245), 261–266 (2015)
16. Ichida, A.Y., Meneguzzi, F.: Modeling a conversational agent using bdi framework. In: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing. p. 856–863 (2023)
17. Lincoln, N., Veres, S.M.: Natural language programming of complex robotic BDI agents. Journal of Intelligent & Robotic Systems **71**(2), 211–230 (2013)
18. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv (arXiv:1907.11692 [cs.CL]) (2019)

19. Longo, C.F., Riela, P.M., Santamaria, D.F., Santoro, C., Lieto, A.: A framework for cognitive chatbots based on abductive–deductive inference. Cognitive Systems Research **81**, 64–79 (2023)
20. Maccartney, B.: Natural Language Inference. Ph.D. thesis, Stanford University, Stanford, CA, USA (2009)
21. McCoy, T., Pavlick, E., Linzen, T.: Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3428–3448. Association for Computational Linguistics (Jul 2019)
22. Meneguzzi, F., de Silva, L.: Planning in bdi agents: a survey of the integration of planning algorithms and agent reasoning. Knowledge Engineering Review **30**(1), 1–44 (2015)
23. Nye, M., Tessler, M., Tenenbaum, J., Lake, B.M.: Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. In: Advances in Neural Information Processing Systems. vol. 34, pp. 25192–25204 (2021)
24. Padgham, L., Winikoff, M.: Developing intelligent agent systems: A practical guide, vol. 13. John Wiley & Sons, Inc. (2005)
25. Qian, P., Unhelkar, V.: Evaluating the role of interactivity on improving transparency in autonomous agents. In: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems. p. 1083–1091 (2022)
26. Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X.: Pre-trained models for natural language processing: A survey. Science China Technological Sciences **63**(10), 1872–1897 (2020)
27. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research **21**(1), 5485–5551 (2020)
28. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-agent World. vol. 1038, pp. 42–55 (1996)
29. Rudin, C., Radin, J.: Why are we using black box models in ai when we don't need to? a lesson from an explainable ai competition. Harvard Data Science Review **1**(2) (2019)
30. Schut, M., Wooldridge, M., Parsons, S.: On partially observable mdps and bdi models. In: Foundations and Applications of Multi-Agent Systems, pp. 243–259. Springer (2002)
31. de Silva, L., Meneguzzi, F., Logan, B.: Bdi agent architectures: A survey. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. pp. 4914–4921. International Joint Conferences on Artificial Intelligence Organization (7 2020)
32. Singh, D., Sardina, S., Padgham, L., Airiau, S.e.p.: Learning context conditions for bdi plan selection. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems. pp. 325–332 (2010)
33. Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., Kambhampati, S.: Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. arXiv (arXiv:2206.10498 [cs.CL]) (2023)
34. Wang, R., Jansen, P., Côté, M.A., Ammanabrolu, P.: Scienceworld: Is your agent smarter than a 5th grader? arXiv (arXiv:2203.07540 [cs.CL]) (2022)
35. Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., Zhou, M.: Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: Advances in Neural Information Processing Systems. vol. 33, pp. 5776–5788 (2020)

36. Williams, A., Nangia, N., Bowman, S.: A broad-coverage challenge corpus for sentence understanding through inference. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 1112–1122. Association for Computational Linguistics (Jun 2018)
37. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45 (Oct 2020)
38. Yao, S., Rao, R., Hausknecht, M., Narasimhan, K.: Keep CALM and explore: Language models for action generation in text-based games. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. pp. 8736–8754 (2020)