Learning Robot Speech Models to Predict Speech Acts in HRI

Ankuj Arora, Humbert Fiorino, Damien Pellier, Sylvie Pesty

To cite this version:

Ankuj Arora, Humbert Fiorino, Damien Pellier, Sylvie Pesty. Learning Robot Speech Models to Predict Speech Acts in HRI. Paladyn: Journal of Behavioral Robotics, 2018, 9 (1), pp.295 - 306. 10.1515/pjbr-2018-0015. hal-01910362

Research Article Open Access

Ankuj Arora*, Humbert Fiorino, Damien Pellier, and Sylvie Pesty

Learning robot speech models to predict speech acts in HRI

https://doi.org/10.1515/pjbr-2018-0015
Received November 29, 2017; accepted July 5, 2018

Abstract: In order to be acceptable and able to "camouflage" into their physio-social context in the long run, robots need to be not just functional, but autonomously psycho-affective as well. This motivates a long term necessity of introducing behavioral autonomy in robots, so they can autonomously communicate with humans without the need of "wizard" intervention. This paper proposes a technique to learn robot speech models from human-robot dialog exchanges. It views the entire exchange in the Automated Planning (AP) paradigm, representing the dialog sequences (speech acts) in the form of action sequences that modify the state of the world upon execution, gradually propelling the state to a desired goal. We then exploit intra-action and inter-action dependencies, encoding them in the form of constraints. We attempt to satisfy these constraints using a weighted maximum satisfiability model known as MAX-SAT, and convert the solution into a speech model. This model could have many uses, such as planning of fresh dialogs. In this study, the learnt model is used to predict speech acts in the dialog sequences using the sequence labeling (predicting future acts based on previously seen ones) capabilities of the LSTM (Long Short Term Memory) class of recurrent neural networks. Encouraging empirical results demonstrate the utility of this learnt model and its long term potential to facilitate autonomous behavioral planning of robots, an aspect to be explored in future works.

Keywords: human robot interaction, automated planning, SAT, LSTM

1 Introduction

For social robots, socio-communicative qualities are increasingly becoming indispensable as they strengthen the possibility of acceptance and emotional attachment to the robot [1, 2]. This acceptance is likely only if the robot fulfills a fundamental expectation that one being has of the other: not only to do the right thing, but also at the right time and in the right manner [1]. Classical Human Robot Interaction (HRI) approaches advocate that much like humans, robots can only learn with experience. Thus the most common strategy used to endow robots with social skills is to analyze, model and implement human behaviors either by observation, imitation or demonstration techniques [3]. The limitation with this approach is the misalignment of the degrees of freedom of the robot with those of the human instructor. The other approach is to "codify" speech into robots. While this ensures autonomy, encoding into a robot the subtleties of a social interaction is not trivial. A standard dialogue exchange integrates the widest possible panel of signs which intervene in the communication and are difficult to codify (the tone of the voice, emotion being conveyed etc.). Human emotions are masked in subtlety, making them complicated as it is to understand and respond to, rendering them even more difficult to program. Scripting these subtleties is time consuming, impractical and requires a lot of programming. In such a complex scenario, learning the underlying speech model of the robot from HRI dialog sequences is a promising alternative.

This learning can be done with the help of Artificial Intelligence (AI) techniques, by viewing the entire scenario in the Automated Planning (AP) paradigm. In this study, the body of HRI dialog exchanges is drawn from the Vernissage corpus [4] and consists of exchanges between a Nao robot (operated by a skilled operator in a Wizard-of-Oz setup) posing as a museum guide and two human visitors to the museum. An example of the dialog exchange can be seen in Listing 1.

Listing 1: Example of dialog exchange between NAO as a museum guide and museum visitors.

```
Nao : Hello , my name is Nao , what is
yours?
```

***Corresponding Author: Ankuj Arora:** Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France;
E-mail: ankuj.arora@univ-grenoble-alpes.fr
Humbert Fiorino: Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France;
E-mail: humbert.fiorino@univ-grenoble-alpes.fr
Damien Pellier: Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France;
E-mail: damien.pellier@univ-grenoble-alpes.fr
Sylvie Pesty: Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France; E-mail: sylvie.pesty@univ-grenoble-alpes.fr

```
Human : My name is Alan
Nao : It's a pleasure to meet you Alan,
can I tell you something about these
paintings?
Human : Yes, please
```

In the context of AP, each utterance is considered ideologically equivalent to an action (represented in the corpus in the form of an action name and signature). Thus, each dialog sequence is represented as a sequential and orchestrated action name-signature sequence which effectuates transitions in the world state, gradually propelling it towards a predetermined goal. By means of these utterances, the speaker *plans* to influence the listeners' beliefs, goals and emotional states [1, 2]; with the intent of fulfilling his own goals. These utterances, interleaved with head and body movements (i.e. multi modal acts) can then be modeled as sequences of actions in an AP system, and frameworks can be developed for providing their semantics [5, 6]. However, for the sake of simplicity, in this work we ignore the definite uncertainty in communication, as well as the head and body movements comprising the acts, and concentrate solely on utterances. The objective is then to leverage the advancements in Machine Learning (ML) to learn the underlying action model from the data i.e. learning the preconditions and effects of each of the constituent actions from their signatures. Learning the underlying model comprising action descriptions from action name-signature sequences could save the effort from having to code these action descriptions from scratch, thus promoting re usability. This model can further be fed to an automated planner to generate fresh dialogs, thus allowing the robot to communicate autonomously in future scenarios.

Our contribution in this paper is the following: given a HRI dialog corpus, our approach learns the robot's behavioral model comprising of the utterances encoded in the form of actions alongwith their signatures, preconditions and effects. The novelty of the approach lies in bridging of symbolic and connectionist machine learning methods to make progress on the problem of learning an interactive model of a human for use in human-robot interaction. The work can be considered a progressive step in the automatic generation of dialog sequences, an aspect which will become more and more relevant as social robotics makes inroads into the real world and manual interaction script generation becomes economically inviable. The approach is divided into three phases. In the first phase, HRI dialogues taken from the Vernissage corpus [4] are annotated with the following pre-engineered speech acts: *(sayHello, farewell, inform, say, ask, claim, deny, advise, thank, autoFeedback (feedback on last heard dialogue e.g. "oh", "great" etc.))*. These acts are drawn from the encoding of speech acts introduced in [7]. The reader is invited to read the aforementioned paper for more detail on the encoding scheme. The second phase consists of learning the speech model. The MAX-SAT framework is used for this purpose. The constraints used as the input for this framework are constituted by intra-action and inter-action specific constraints. These constraints are then fed to a weighted MAX-SAT solver, the solved constraints being used to reconstruct the underlying action model. Feeding this learnt model to a planner to generate fresh interactions and testing them in a real scenario is beyond the scope of this paper. We, however, demonstrate the utility of this learnt model by predicting speech acts likely to be executed by the robot for the same dialog corpus. For example, using the speech model, we predict whether in the scenario of the robot meeting a subject for the first time, the probability that the act of greeting the subject is followed by the act of introducing itself. This is performed with the aid of sequence labelling capabilities of recurrent neural networks. Deep learning, in general, offers multiple layers of representation wherein the features are not designed by human engineers but learn from data using a general-purpose learning procedure [8]. It has also been extensively used in the field of robotics [9]. In particular, the successes of LSTM [10], a deep learning technique being increasingly used in learning long range dependencies in the fields of speech and handwriting recognition [8], are exploited. A plan is an orchestrated execution of a series of actions which are by virtue interdependent in order to execute. There is thus a direct link between the principle of operation of the LSTM and plan execution, which this paper seeks to explore.

This paper is divided into the following sections: we firstly present some related work in Section 2, followed by the definition of our learning problem in Section 3. We then detail the functioning of our learning system in Section 4, following it up with empirical evaluations in Section 5. We conclude the paper with some perspectives and future work in Section 6.

2 Related work

We elaborate the state of the art of two different fields: machine learning (ML) in AP and ML in robotics. Learning action models in the field of AP have a considerable history. Some prominently used ML techniques to learn action models include: inductive techniques (absence of background knowledge, e.g. PELA [11]), transfer learning techniques (extraction of knowledge from one or more source tasks to apply to the target task when the latter has fewer high-quality training data [12] e.g. TRAMP [13]), reinforcement learning based approaches (e.g. [14]). More specific to our case, certain approaches have used the MAX-SAT framework to learn deterministic actions (e.g. ARMS [15]), macro-actions [16], models in Hierarchical Task Networks (HTNs) [17], and models in multi-agent setups ([18]). In particular, our approach is on the same lines of ARMS, which also generates intra-action and inter-action constraints (mined with the Apriori algorithm [19]). These approaches, however, use a partially ordered series of actions as input. Since in our problem multiple speech acts can co-occur at the same instant (e.g. multiple participants speaking at the same time), we are primarily interested in co-occurring actions. On the robotics side, there has been considerable use of ML for task-oriented robot behavior learning [20] and robot motion planning [21]. Deep learning has also been extensively used in the robotics community [9]. Our work, however, is concentrated on the learning of PDDL (Planning Domain Description Language) [22], a syntax for planning domains used for behavioral models for robots; a requirement not catered for in the aforementioned works. The literature also speaks of other approaches which have treated dialogues as planning operators. In [23] the authors introduce a new language called MAPL (Multi Agent Programming Language) which represents speech acts as operators with qualitative and quantitative temporal relations between them. Other works use AP to generate natural language sentences for communication [24] or treating utterances in the form of actions [25]. Contrary to the previous works, our problem is centered on the learning of the action model which serves as a prerequisite to be fed to a planner prior to it being utilized in the forms mentioned as in the aforedescribed works.

3 Definitions and problem formulation

We begin by laying out some important definitions of AP which symbolize the HRI interaction. In the field of AP, agents interact with the environment by executing actions which change the state of the environment, gradually propelling it from its initial state towards the agents' desired goal. In the classical representation, both the world state and actions are pre-engineered and constituted by properties called *predicates*. *States* are defined to be sets of ground (positive) predicates. Here, each action a∈A where A = $\{a_1, a_2, \dots a_n\}$, n being the maximum number of actions in the domain. We use actions and operators interchangeably in our context. These actions constitute a corpus which serves as a blueprint for these actions, called the *action model*. An *action model m* is the blueprint of all the domain-applicable actions belonging to the set A. Each action in the model is defined as an aggregation of : (i) the action name (with zero or more typed variables as parameters), and (ii) three lists, namely (pre, add and del). These are: the *pre* list (predicates whose satisfiability determines the applicability of the action), *add* list (predicates added to the current system state by the action execution) and the *del* list (predicates deleted from the current system state upon action execution), respectively. A *planning problem* is a triplet P = (s_0, g, m) composed of (i) the initial state of the world s_o, (ii) the desired goal g and (iii) the action model m. All the aforementioned elements in this section contribute to the formulation of a *plan*. A *plan*, given P, is a sequence of actions represented as: $\pi = [a_1, a_2, \dots, a_n]$ that drives the system from the initial state to the goal. Each action sequence, complete with initial state and goal information, constitutes a *trace*. Each dialogue sequence in HRI can be viewed as a trace in AP. An aggregation of these traces constitutes a trace set *T*, which in this case corresponds to our HRI dialog corpus. Having described these preliminaries, we can proceed to formulating our problem.

Given the aforementioned information, our problem can be formulated as follows: given (i) a set of HRI traces *T*, each trace consisting of a dialogue sequence encoded in the form of actions; our approach produces a complete domain model *m* encompassing all the domain-applicable operators which best explain the observed traces. This is done by encoding the inter-operator and intra-operator dependencies in the form of constraints and solving them as a satisfiability problem, then reconstructing the domain model from the satisfied constraints. We then proceed to use the learnt model to predict the label of the operators in the traces using LSTM techniques.

4 Approach

The approach can be divided into four phases, a snapshot of which is illustrated in Figure 1. The dialogs of the HRI traces are first annotated to symbolize a sequence of actions (each consisting of a name and a signature). In the first phase, we generalize the grounded actions in the trace set by replacing the variables by their pre-engineered types to obtain a trace set of operators. The second phase is dedicated to constraint generation; namely intra-operator

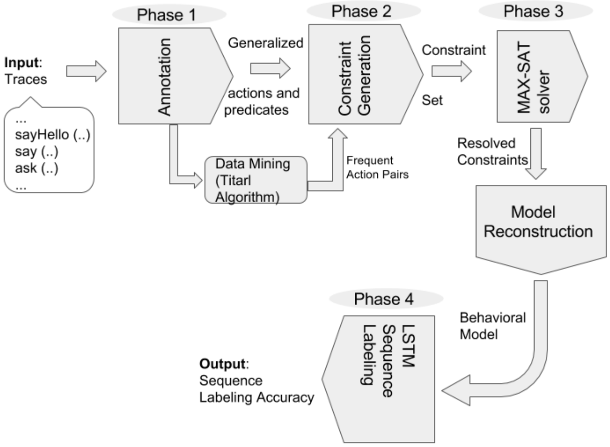


Figure 1: Approach phases.

and inter-operator ones. In the third phase, these constraints are supplied to a MAX-SAT solver and the satisfied ones are used to reconstruct the action model. This model consists of operator signatures, preconditions (alias for *pre* list), and effects (alias for *add* and *del* lists). It conforms to the semantics of STRIPS [26]. In the fourth phase, using the operators learnt in the previous phase as features, we predict the operators likely to be chosen by the robot based on previous ones during the dialog exchange highlighted in the corpus. This classification is done by encoding the operator sequences of the corpus in the form of input and output vectors to an LSTM. We elaborate these phases in the forthcoming subsections.

4.1 Annotation and generalization

In this phase, dialogues in the dialogue corpus are first annotated to actions. These dialogues are taken from the Vernissage corpus which is a multimodal HRI dataset [4], and has a total of 10 conversation instances between Nao and human participants. In the scenario, the robot explains paintings in a room and then performs a quiz with the participants. We first annotate with predefined predicates the initial state of the world before the beginning of each interaction. We then annotate each robot dialogue within each trace as an action drawn from the following set of actions: *(sayHello, farewell, inform, say, ask, claim, deny, advise, thank, autoFeedback (feedback on last heard dialogue e.g. "oh", "great" etc.))*. The participant responses are encoded in the form of *(speech, silence, laughter)*. These actions and predicates are derived from the encoding of speech acts in [7]. This encoding is chosen as it depicts speech acts in logical form, rendering it more conducive to translate into PDDL operators. These actions are represented in the form of constituents of a PDDL model [22] whose snippet is represented in Listing 3. This model is referred to as the ground truth model, or the correct baseline model that the empirically generated model will compare to in order to gauge the accuracy of our proposed learning approach. This ground truth model consists of the names of the 11 actions described above, alongwith each of their preconditions and effects in the form of predicates. The interpretation of these predicates is detailed in Table 2. An example of the annotation process can be seen in Table 1.

The next step is generalization, which is done by scanning each action in each of the traces and substituting its instantiated variables with their corresponding types. This produces a trace set of operator sequences, with the generalized actions constituting an operator schema O_S. We then create a dictionary of all possible relevant predicates to each operator, the keys of the dictionary identified by the operator names. Each operator in the operator schema is associated with its relevant predicates, where a predicate p is said to be relevant to an operator $o \in O_S$ if they share the same variable types. We denote the relevant predicate dictionary as $relPre$, with the set of relevant predicates to an operator o_i (represented as key) can be denoted as $relPre_{o_i}$ (represented as value of key o_i). The generalization procedure is represented in Figure 2. The complexity of this phase comes out to be lmn, where l is the number of actions per trace, m is the number of traces, and n is the number of unique predicates in the dictionary.

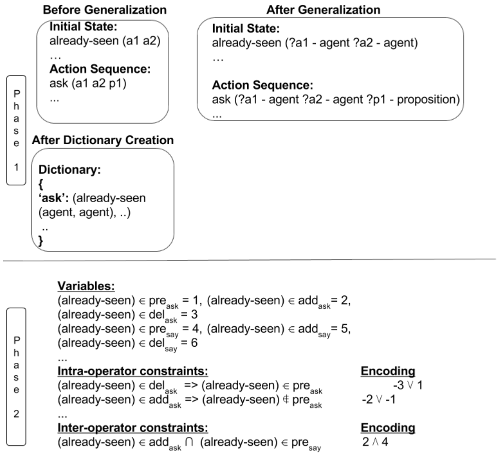
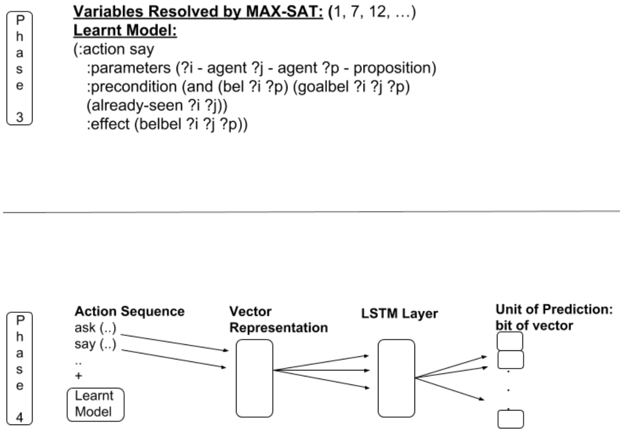

Before Generalization
Initial State:
already-seen (a1 a2)
...
Action Sequence:
ask (a1 a2 p1)
...
After Generalization
Initial State:
already-seen (?a1 - agent ?a2 - agent)
...
Action Sequence:
ask (?a1 - agent ?a2 - agent ?p1 - proposition)
...
Phase 1
After Dictionary Creation
Dictionary:
{
'ask': (already-seen (agent, agent), ..)
..
}
Phase 2
Variables:
(already-seen) $\in pre_{ask}$ = 1, (already-seen) $\in add_{ask}$ = 2,
(already-seen) $\in del_{ask}$ = 3
(already-seen) $\in pre_{say}$ = 4, (already-seen) $\in add_{say}$ = 5,
(already-seen) $\in del_{say}$ = 6
...
Intra-operator constraints:
(already-seen) $\in del_{ask}$ => (already-seen) $\in pre_{ask}$
(already-seen) $\in add_{ask}$ => (already-seen) $\notin pre_{ask}$
...
Encoding
$-3 \vee 1$
$-2 \vee -1$
Inter-operator constraints:
(already-seen) $\in add_{ask} \cap$ (already-seen) $\in pre_{say}$
Encoding
$2 \wedge 4$

Figure 2: Illustration of Phases 1 and 2 of our approach. In Phase 1, the dialog corpus is annotated with pre-engineered speech acts to produce a representation closer to AP. The new representation is then used to formulate intra and inter-operator constraints in the second phase.



```
(:action say
   :parameters (?i - agent ?j - agent ?p - proposition)
   :precondition (and (bel ?i ?p) (goalbel ?i ?j ?p)
   (already-seen ?i ?j))
   :effect (belbel ?i ?j ?p))
```

Figure 3: Illustration of Phases 3 and 4 of our approach. Phase 3 involves solving of the formulated constraints as a SAT problem to reconstruct the underlying model. Phase 4 involves using the previously learnt model to perform sequence labelling on the speech acts comprising the training data.

Listing 2: Predicates and their implications in speech acts.

```
bel (?i ?p) - agent ?i believes
in the proposition ?p

goal (?i ?p) - proposition ?p is the
goal of agent ?i

ideal (?i ?p) - proposition ?p should
ideally be true for agent ?i

belbelapproval (?i ?j ?p) - agent ?i
believes that agent ?j believe
that ?i gives its approval for
proposition ?p

belideal (?i ?j ?p) - agent ?i
believes that proposition ?p
should be the ideal for agent ?j

goal (?i ?p) - proposition ?p is not
the goal of agent ?i

notbel (?i ?p) - agent ?i does not
believe in the proposition ?p

belbel (?i ?j ?p) - agent ?i believes
that agent ?j believes in the
proposition ?p

belgoal (?i ?j ?p) - agent ?i believes that
the goal of agent ?j is proposition ?p

notbelbel (?i ?j ?p) - agent ?i believes that
agent ?j does not believe in the proposition ?p

goalbel (?i ?j ?p) - the goal of agent ?i
is that agent ?j believes in proposition ?p

goalresp (?i ?j ?p) - the goal of agent ?i
is that agent ?j assumes responsibility
of proposition ?p

notgoalresp (?i ?j ?p) - the goal
of agent ?i is not that the agent ?j
assumes responsibility of proposition
?p

belbelgoalresp (?i ?j ?p) - agent
?i believes that agent ?j believes
that the goal of ?i is ?j assumes
the responsibility of proposition
?p

belbelnotgoalresp (?i ?j ?p) - agent
?i believes that agent ?j believes
that the goal of ?i is not that ?j
assumes the responsibility of
proposition ?p

goalbelgoalrespWeak (?i ?j ?p) - goal
of agent ?i is that agent ?j believes
that the goal of ?i is that ?j assumes
the responsibility of proposition ?p
to a lesser magnitude

goalbelstr (?i ?j ?p) - goal of agent
?i is that agent ?j believes strongly
in proposition ?p

belbelgratitude (?i ?j ?p) - goal of
agent ?i is to express to agent ?j
its gratitude for proposition ?p

belresp (?i ?j ?p) - goal of agent ?i
is that agent ?j assumes
responsibility for proposition ?p

belbelmoralsatisfaction (?i ?j ?p) -
agent ?i believes that agent ?j
believes that agent ?i assumes
moral satisfaction with
proposition ?p

look (?i ?j) - agent ?i looks
at agent ?j

at (?i ?r) - agent ?i is at room ?r

already-seen (?i ?j) - agent ?i has
already seen agent ?j

never-seen (?i ?j) - agent ?i has
never seen agent ?j
```

4.2 Constraint generation

In this phase, we detail the intra-action constraints for individual operators and inter-operator temporal constraints among operators.

4.2.1 Intra-operator constraints

In order to satisfy the semantics of STRIPS [26], each operator in O_s must satisfy certain *intra-operator constraints*. Thus, for each operator $o_i \in O_s$ and relevant predicate $p \in relPre_{o_i}$: (i) p cannot be in the *add* list and the *del* list at the same time, and (ii) p cannot be in the *add* list and the *pre* list at the same time. The relevant predicates of each operator are encoded to generate variables. Each association of a relevant predicate with one of a *(pre, add, del)* list of an operator can be encoded as a variable. These variables and constraints are illustrated in Figure 2.

4.2.2 Inter-operator constraints

Any operator sequence exhibits inter-operator dependencies, which can be uncovered by means of data mining techniques to facilitate the process of learning. In the case of an HRI scenario, operators may be temporally concurrent: for example, two agents may be speaking at the same time. These temporal dependencies can be explored with the help of pattern mining techniques. The aforementioned trace set T can thus be treated as a set of transactions S in the pattern mining domain. S can be written as $S = [s_1, s_2, \dots, s_n]$, where each transaction s_n represents a trace. A transaction is a set of symbols, in our case representing operators. A *time series* is a set of unique time points. A *symbolic time series* is generally used to represent series of ordered (but non time-sampled) events (e.g. $< a, b, d, a, c >$). A *symbolic time sequence* is a multi-set of time points. Contrary to time series, *symbolic time sequences* can deal with several events defined at the same temporal location. This difference is also illustrated in the Figure 7. We symbolize the concurrent utterances in the Vernissage corpus in the form of operators chaining together in a symbolic time sequence [29].

Listing 3: Ground truth speech model called "hri" to symbolize dialog exchange in Vernissage corpus. The pre-engineered types are *(agent, proposition, room)* are defined adjacent to the *:types* keyword. The predicates are defined adjacent to the *:predicates* keyword. This is followed by the definition and description of the constituent actions, (their preconditions and effects defined adjacent to the *:precondition* and *:effect* keywords respectively.

```
(define (domain hri)
(:requirements :strips :typing)
(:types
   agent - object
   robot human - agent
   proposition - object
   room - object
)
(:predicates
   (bel ?i - agent ?p - proposition)
   (goal ?i - agent ?p - proposition)
   (ideal ?i - agent ?p - proposition)
   (approval ?i - agent
   ?p - proposition)
   (belbelapproval ?i - agent
   ?j - agent ?p - proposition)
   (belideal ?i - agent ?j - agent
   ?p - proposition)
   (notgoal ?i - agent
   ?p - proposition)
   (notbel ?i - agent
   ?p - proposition)
   (belbel ?i - agent ?j - agent
   ?p - proposition)
   (belgoal ?i - agent ?j - agent
   ?p - proposition)
   (notbelbel ?i - agent ?j - agent
   ?p - proposition)
   (goalbel ?i - agent ?j - agent
   ?p - proposition)
   (goalresp ?i - agent ?j - agent
   ?p - proposition)
   (notgoalresp ?i - agent ?j - agent
   ?p - proposition)
   (belbelgoalresp ?i - agent
   ?j - agent ?p - proposition)
   (belbelnotgoalresp ?i - agent
   ?j - agent ?p - proposition)
   (goalbelgoalresp ?i - agent
   ?j - agent ?p - proposition)
   (goalbelstr ?i - agent ?j - agent
   ?p - proposition)
   (goalbelgoalrespWeak ?i - agent
   ?j - agent ?p - proposition)
   (belbelgratitude ?i - agent
   ?j - agent ?p - proposition)
   (belresp ?i - agent ?j - agent
   ?p - proposition)
   (belbelmoralsatisfaction ?i - agent
   ?j - agent ?p - proposition)
   (look ?a - agent ?b - agent)
   (at ?a - agent ?l - room)
   (already-seen ?i - agent ?j - agent)
   (never-seen ?i - agent ?j - agent)
)


;; One agent greets another agent
(:action sayHello
  :parameters (?i - agent ?j - agent
  ?r - room)
  :precondition (and (at ?i ?r) (at ?j ?r)
  (look ?i ?j) (never-seen ?i ?j))
  :effect (and (already-seen ?i ?j)
  (not (never-seen ?i ?j)))
)

;; One agent bids farewell to another agent
(:action farewell
  :parameters (?i - agent ?j - agent ?r - room)
  :precondition (and (at ?i ?r) (at ?j ?r)
  (look ?i ?j) (already-seen ?i ?j))
  :effect ((not(look ?i ?j))
)

;;One agent informs another agent
(:action inform
         :parameters (?i - agent ?j - agent
    ?p - proposition)
         :precondition (and (bel ?i ?p)
    (goalbel ?i ?j ?p) (notbelbel ?i ?j ?p)
    (already-seen ?i ?j))
        :effect (belbel ?i ?j ?p)
)

;;One agent says something to another agent
(:action say
         :parameters (?i - agent ?j - agent
?p - proposition)
         :precondition (and (bel ?i ?p)
    (goalbel ?i ?j ?p) (already-seen ?i ?j))
        :effect (belbel ?i ?j ?p)
)

;;One agent asks/tells/suggests to another agent
(:action ask
        :parameters (?i - agent
    ?j - agent ?p - proposition)
         :precondition (and
    (goalresp ?i ?j ?p)
         (goalbelgoalresp ?i ?j ?p)
    (already-seen ?i ?j))
        :effect (and
    (belbelgoalresp ?i ?j ?p))
)

;;One agent claims something to
;;another agent
(:action claim
         :parameters (?i - agent
    ?j - agent  ?p - proposition)
        :precondition (and (bel ?i
    ?p)
    (goalbelstr ?i ?j ?p)
        (already-seen ?i ?j))
        :effect (belbel ?i ?j ?p)
)

;;One agent denies something to
;;another agent
(:action deny
        :parameters (?i - agent
    ?j - agent
    ?p - proposition)
        :precondition (and (bel ?i
    ?p)
    (goalbel ?i ?j ?p)
        (already-seen ?i ?j))
        :effect (belbel ?i ?j ?p)
)

;;One agent advises something
;;to another agent
(:action advise
        :parameters (?i - agent
    ?j - agent  ?p - proposition)
        :precondition (and
    (goalresp ?i ?j ?p)
        (goalbelgoalrespWeak ?i ?j
    ?p) (already-seen ?i ?j))
        :effect (belbelgoalresp ?i
    ?j ?p)
)

;;One agent thanks another agent
(:action thank
        :parameters (?i - agent
    ?j - agent ?p - proposition)
        :precondition (and (goal
    ?i ?p)
    (belresp ?i ?j ?p)
        (already-seen ?i ?j))
        :effect (belbelgratitude
    ?i ?j ?p)
)

;;One agent thanks another agent
(:action autoFeedback
        :parameters (?i - agent ?j -
    agent
    ?p - proposition)
        :precondition (and (goal ?i ?p)
    (belresp ?i ?j ?p)
        (already-seen ?i ?j))
        :effect (belbelmoralsatisfaction
    ?i ?j ?p)
)
)
```

Various approaches to mine temporal time sequences are presented in the literature. Winepi [27] is a well known algorithm which learns episodes and association rules based on the episodes. The *face* algorithm allows for mining of chronicles from symbolic time sequences [28]. These approaches are, however, not equipped to deal with temporal inaccuracies in the temporal events. We choose the Temporal Interval Tree Association Rule Learning (Titarl) algorithm [29] as it allows the representation of imprecise (non-deterministic) and inaccurate temporal information between speech acts considered as symbolic events. Following is an example of a rule mined with Titarl: “If there is an event A at time t, then an event of type B will occur between times t+5 and t+10 with a 95% chance". The temporal relationships between operators can be uncovered by means of *association rule learning*, which is the search for association rules. An association rule is a $conditions \rightarrow implications$ pattern.

We hypothesize that if an association rule frequently correlates two operators, there must be a reason for their frequent co-existence. The input traces are parsed and fed to the Titarl algorithm, which produces temporal association rules [29]. The operators featuring in the frequent temporal rules are suspected to share a “semantic” relationship among themselves, which can be represented in the form of *inter-operator constraints*. These constraints have been proposed by the ARMS [15] system, and may serve as heuristics to explain the frequent co-existence of these operators. More precisely, if there is an operator pair (o_i, o_j), $0 \leq i < j \leq (n-1)$ where n is the total number of operators in the plan; and pre_i, add_i and del_i represent o_i’s *pre*, *add* and *del* list, respectively:

- A predicate *p* which serves as a precondition in the *pre* lists of both o_i and o_j cannot be deleted by the first operator.
- A predicate p added by the first operator o_i ($p \in add_{o_i}$) which serves as a prerequisite for the second operator o_j ($p \in pre_{o_j}$), cannot not be deleted by the first operator o_i.
- A predicate p that is deleted by the first operator o_i is added by o_j. In other words, an operator re-establishes a predicate that is deleted by a previous operator.
- The above plan constraints can be combined into one constraint and restated as: $\exists p((p \in (pre_i \cap pre_j) \wedge p \notin (del_i)) \vee (p \in (add_i \cap pre_j)) \vee (p \in (del_i \cap add_j))$

A snippet of the aforementioned constraints is illustrated for the action pair *(ask, say)* and relevant predicate *already-seen* in the Figure 2.

The details pertaining to the resolution of the intra and inter-operator constraints with the help of a SAT solver to reconstruct the underlying action model is detailed in the Figure 3.

4.3 LSTM based operator classification

As mentioned before in section 3, a plan is a chained series of interdependent operators directed towards the accomplishment of a goal. Thus, extracting patterns from sequences of previously executed operators is likely to provide strong evidence to predict the label of the next operator in the chain; inspiring our investigation of long short-term memory networks (LSTMs) [10] for action sequence labelling. In the following subsections we present our data encoding method for the input and output vector of the LSTM.

Table 1: Sample annotation of traces. The “Utterance” in the second column at the timestamp (in seconds) from the beginning of the interaction mentioned in the column “Timestamp” is annotated with the speech act in the column “Annotation”. The (..) represents the operator parameters.

Timestamp	Utterance	Annotation
35.803	Hello	sayHello (..)
35.803	Should I tell you something about these paintings?	ask (..)
395.619	So, I am starting the quiz!	claim (..)
556.885	Great, your answer is perfectly right.	auto Feedback (..)
629.532	So. It’s the end of this quiz.	say (..)

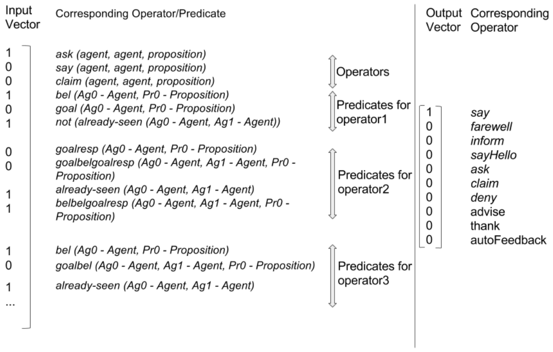

Input Vector
Corresponding Operator/Predicate
1
0
0
1
0
1
0
0
1
1
1
0
1
...
ask (agent, agent, proposition)
say (agent, agent, proposition)
claim (agent, agent, proposition)
bel (Ag0 - Agent, Pr0 - Proposition)
goal (Ag0 - Agent, Pr0 - Proposition)
not (already-seen (Ag0 - Agent, Ag1 - Agent))
goalresp (Ag0 - Agent, Pr0 - Proposition)
goalbelgoalresp (Ag0 - Agent, Ag1 - Agent, Pr0 - Proposition)
already-seen (Ag0 - Agent, Ag1 - Agent)
belbelgoalresp (Ag0 - Agent, Ag1 - Agent, Pr0 - Proposition)
bel (Ag0 - Agent, Pr0 - Proposition)
goalbel (Ag0 - Agent, Ag1 - Agent, Pr0 - Proposition)
already-seen (Ag0 - Agent, Ag1 - Agent)
Operators
Predicates for operator1
Predicates for operator2
Predicates for operator3
Output Vector
Corresponding Operator
1
0
0
0
0
0
0
0
0
0
say
farewell
inform
sayHello
ask
claim
deny
advise
thank
autoFeedback

Figure 4: Vector representations for the operator “ask” and successive operator “say” for the learnt behavioral model.

4.3.1 Data encoding for labelling of operator sequences

We use the sequence labelling capabilities of LSTM to identify the most likely operator that succeeds a given one. The input to our LSTM system is a large corpus comprising vector representations of each operator of each trace. Each trace is taken one by one, and the comprising operators are sequentially encoded into input and output vectors; thus producing a large corpus of vectors. Each operator of each trace is represented by two distinct vectors: an input vector which encodes the operator, and an output vector which classifies the successive operator. These vec-

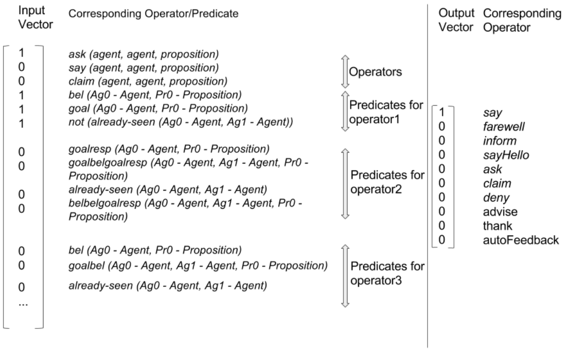

Input Vector
Corresponding Operator/Predicate
1 ask (agent, agent, proposition)
0 say (agent, agent, proposition)
0 claim (agent, agent, proposition)
1 bel (Ag0 - Agent, Pr0 - Proposition)
1 goal (Ag0 - Agent, Pr0 - Proposition)
1 not (already-seen (Ag0 - Agent, Ag1 - Agent))
0 goalresp (Ag0 - Agent, Pr0 - Proposition)
0 goalbelgoalresp (Ag0 - Agent, Ag1 - Agent, Pr0 - Proposition)
0 already-seen (Ag0 - Agent, Ag1 - Agent)
0 belbelgoalresp (Ag0 - Agent, Ag1 - Agent, Pr0 - Proposition)
0 bel (Ag0 - Agent, Pr0 - Proposition)
0 goalbel (Ag0 - Agent, Ag1 - Agent, Pr0 - Proposition)
0 already-seen (Ag0 - Agent, Ag1 - Agent)
...
Operators
Predicates for operator1
Predicates for operator2
Predicates for operator3
Output Vector
Corresponding Operator
1 say
0 farewell
0 inform
0 sayHello
0 ask
0 claim
0 deny
0 advise
0 thank
0 autoFeedback

Figure 5: Vector representations for the operator "ask" and successive operator "say" for the features respective to the relevant predicates in the encoding scheme sans speech model.

tors serve as the input and output respectively to the LSTM cells, the encoding of which represents the core of this section. This corpus of vector representations is divided into a training and validation set to eventually train the LSTM on the training set and gauge its performance on the validation set. At the output of this learning system, we obtain an accuracy of prediction on the folds of validation data. The encoding of the input and output vectors is presented in the following paragraphs.

The input vector representing an operator in a trace is encoded in the following fashion. It is divided into two sections: one section which labels the entire set of operators in the domain, and the other which labels the relevant predicates for the operators in the domain. In the first block, there is a slot for each operator in the domain. The slot for the operator currently being encoded is labeled as 1, and the slots for the remaining operators in the domain are labeled as 0. Thus if $(o_1, o_2, \dots, o_n) \in O_s$ is the set of domain-applicable operators, the first n elements of the vector will be representing this first block, with the entry for the operator currently being encoded being switched to 1, the other $(n-1)$ slots for the remaining $(n-1)$ operators being kept at 0. Once this first block has been assigned, we dedicate blocks of elements in the vector specific to each operator in the domain. Thus for n operators in the domain, there are n different blocks (plus the one block for all the domain applicable operators as explained above). Each operator-specific block contains one entry for each predicate relevant to that particular operator. For example, if *[goalresp (Ag0 - Agent, Pr0 - Proposition), already-seen (Ag0 - Agent, Ag1 - Agent)]* are two predicates relevant to the operator *ask*, they will constitute two entries in the *ask* operator block. We thus create operator-specific blocks and for each operator, the number of blocks for the input vector standing at $(n+1)$. The dimension d of this input vector is directly proportional to the number of operators in a domain, as well as the number of predicates relevant to each operator. The dimension d of a vector for a specific domain will always remain the same, with the switching of a slot from 0 to 1 in the vector signalling the execution of a particular operator. If $(o_1, o_2, \dots, o_n) \epsilon O_s$ represents the operator schema, the dimension of the input vector is given as:

$$d = n + \sum_{i=1}^{n} relPre_{o_i} \tag{1}$$

Here $relPre_{o_i}$ are the number of relevant predicates for the operator o_i. The output vector predicts the label of the operator that follows the operator currently being encoded in the trace. Very much like the input, the output is encoded as a binary vector. It consists of a single block which has as many slots as the number of operators in the domain, one for each operator. The slot representing the succeeding operator to the operator being currently encoded is set to 1, the others being set to 0. For example, let us assume that the operator currently being encoded is *ask* and the next operator in the trace is *say*. The input and output vectors for the *ask* operator are represented in Figure 4. While the number of operators in a trace thus the number of vectors representing all the operators in a trace may vary, the LSTM requires a fixed sized input. This is ensured by calculating the maximum trace length *batchLen* (maximum number of operators per trace) for all the traces, and padding the shorter lists with d-dimensional vectors filled with zeros. This padding is done for all the traces till all the traces have the same *batchLen* number of operators. The same padding procedure is adopted for the output vectors.

The input vectors are identical in the way they are labelled for the training and validation set. The first section is represented in the same way, with the label of the currently encoded operator set to 1. In this case, the slots in the vector which correspond to the relevant predicates in each operator of the empirical model are set to 1. For example, as illustrated in the Figure 4, if the empirical model is represented by $(operator_1$: $bel(Ag0 - Agent, Pr0 - Proposition), not(already - seen(Ag0 - Agent, Ag1 - Agent)), operator_2$: $(already - seen(Ag0 - Agent, Ag1 - Agent), belbelgoalresp(Ag0 - Agent, Ag1 - Agent, Pr0 - Proposition)), operator_3$: $(bel(Ag0 - Agent, Pr0 - Proposition), already - seen(Ag0 - Agent, Ag1 - Agent))$, then the slots in the vector for the $operator_1$ action which represent the predicates $(bel, \neg(already - seen))$ are switched to 1, the rest of the predicates being kept at 0. This scheme is replicated for the other two operators as well.

In the evaluation phase, the aforementioned scheme is compared with an encoding scheme sans the presence of the speech model. In this alternative scheme, the first section is represented in the same way, with the label of the currently encoded operator set to 1. For the second section, the slots in the vector which correspond to the relevant predicates in each operator are set to 1. For example, as illustrated in the Figure 5, if the action currently being encoded is *ask*, the predicates relevant to the action *ask* i.e. the first block of the second section corresponding to the first operator are labelled to 1, the rest of the blocks of this section being kept at zero.

```
 (:action  say-hello
:parameters( ?Ag0 - Agent ?Ag1 - Agent
?Roo0 - Room)
:precondition (and  (already-seen  ?Ag0 ?Ag1)
(at  ?Ag0 ?Ro0) (at  ?Ag1 ?Ro0))
:effect (and  (not (already-seen ?Ag0 ?Ag1))
(not (at ?Ag0 ?Ro0))))
```

Figure 6: Snapshot of PDDL representation of the learnt action *say-Hello*.

5 Evaluation

The objective of our evaluation is to obtain the best possible accuracy of: (i) the learnt model vis-a-vis the ground truth model and (ii) sequence labeling obtained with the learnt speech model. Concerning inter-operator constraint representation, two measures are defined for association rules of the form $a \Rightarrow b$, b being the head and a being the body of the rule. The *confidence* of a rule is the percentage of occurrences of the rule's body which also matches the body and the head of the rule i.e. $support(body)/support(head + body)$. The *support* of a rule is the percentage of occurrences of the rule's head which also matches the body and the head of the rule i.e. $support(head)/support(head + body)$ [29]. The Titarl-mined association rules along with their confidence and support are recorded. The rules with a highest value of confidence and support (empirically determined) are retained for inter-operator constraint generation (see Table 2).

Finally we encode all the intra and inter-operator constraints in the form of a weighted MAX-SAT problem. The weights of the CNF clauses representing the constraints are determined differently for the inter and intra-operator cases. While the weights of the intra-operator clauses are empirically determined, the weights of the inter-operator clauses are equal to the support of the association rules. This problem can be stated as: Given a collection C of m clauses, $(C_1, \dots, C_m)$ involving n logical variables with clause weights w_i, find a truth assignment that maximizes the total weight of the satisfied clauses in C [15]. We use 2 SAT solvers: the MaxSatSolver [30] and the MaxWalkSat [31]. The solution produced by either solver contains all the variables which evaluate to true, which are then used to reconstruct the empirical model.

The difference between the ground truth model and the empirically determined model is represented in the form of a *reconstruction error*. This error is based on the similarity between the relevant predicates and the empirically determined predicates per operator. Let $diffPre_{o_i}$ represent the difference in *pre* lists of operator o_i in the ground truth model and the empirical model. Each time the *pre* list of the ground truth model presents a predicate which is not in the *pre* list of the empirical model, the count $diffPre_{o_i}$ is incremented by one. Similarly, each time the *pre* list of the empirical model presents a predicate which is not in the *pre* list of the ground truth model, the count $diffPre_{o_i}$ is incremented by one. Similar counts are computed for the *add* and *del* lists as $diffAdd_{o_i}$ and $diffDel_{o_i}$ respectively. This total count is then divided by the number of relevant constraints for that particular operator $relCons_{o_i}$ to obtain the cumulative error per operator. This error is summed up for every operator and averaged to obtain an average error E for the entire model. This cumulative error for the model is represented by:

$$E = \frac{1}{n} \sum_{i=1}^{n} \frac{diffPre_{o_i} + diffAdd_{o_i} + diffDel_{o_i}}{relCons_{o_i}} \qquad (2)$$

The obtained cumulative error is mentioned in the third column of Table 3. The relatively high error rates can be attributed to the fact that owing to the linear structure of the relevant predicate dictionary, there is little variation among the constraints produced specific to each operator, as well as their weights. A great deal of constraints are thus solved, thus introducing a great deal of noise in the reconstructed model. This noise can be illustrated with a snippet of the learnt model in the Listing 6. We conclude from these results that the ground truth model needs to be fine tuned and reworked upon to ensure that the operators being learnt are not semantically as close to ensure a better learning rate.

5.1 LSTM based sequence labeling

In this work, we explore two hyperparameters: the number of hidden units (set between (100, 200)), and the dropout

Table 2: Representation of the temporal rules mined with Titarl algorithm.

Rule	Confidence (%)	Support (%)
ask → say	73	96
inform → deny	69	100
autoFeedback → ask	100	60

Table 3: Cumulative error in model learning (* = with MaxWalkSat, ** = with MaxSatSolver).

Variables	Clauses	Model Error (E)	Execution Time (s)
759	3606	41.78*, 39.78**	36.44*, 36.25**

rate [8] (set between (0.5, 0.75)); both of which have significant potential to influence the predictions. We use a softmax layer for classifying given operator sequences. The batch size is set to $batchLen$. We also use categorical cross entropy for the loss function and an adam optimizer (gradient descent optimizer). Each of the 10 sequences consists of about of 400 dialogues each, which is divided using five-fold cross validation. Every training example is presented to the network 10 times i.e. the network is trained for 10 *epochs*. The results are summarized in the Figure 8 and are the obtained with the encoded behavioral model. The accuracy represented in the figure is the *validation accuracy* (accuracy measured on the validation set), which is the proportion of examples for which the model produces the correct output. It is represented as the fraction of examples classified correctly. The accuracy is recorded for 1, 2, 5 and 10 traces. As can be seen, the accuracy improves with the number of traces. If we consider the learning performance with the encoding scheme sans model in Figure 9, this prediction accuracy is not so impressive. The highest accuracy obtained is in the range of the upper forties. This demonstrates the fact that the presence of the behavioral model in the feature vector boosts the sequence labeling capacity of the learning system.

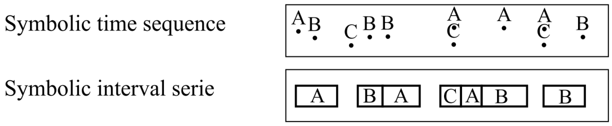


Figure 7: Difference between symbolic time series and symbolic time sequences. While items in a symbolic time series are partially ordered and do not repeat, items in a symbolic time sequence may be co-occurring and repeat as well [29].

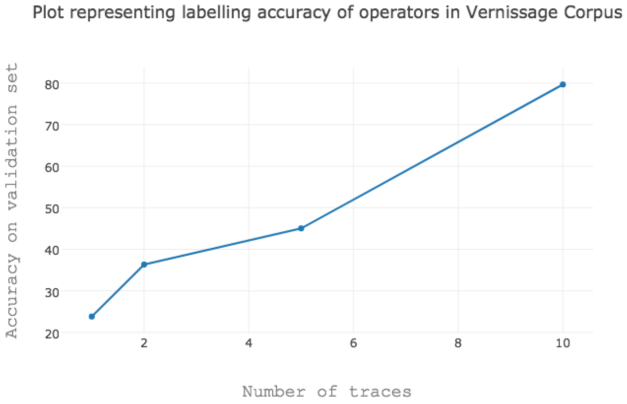


Figure 8: LSTM operator labelling accuracy (128 hidden units, 0.8 dropout rate).

6 Conclusion

In order to be acceptable and be able to "camouflage" into their physio-social context in the long run, robots need to be not just functional, but autonomously psycho-affective as well. This motivates a long term necessity of introducing behavioral autonomy in robots, such that they can autonomously communicate with humans without the need of "wizard" intervention. This paper proposes a technique to learn robot speech models from human-robot dialog exchanges. It views the entire exchange in the Automated Planning (AP) paradigm, representing the dialog sequences (speech acts) in the form of action sequences that modify the state of the world upon execution, gradually propelling the state to a desired goal. We then exploit intra-action and inter-action dependencies, encoding them in the form of constraints. We attempt to satisfy these constraints using a weighted maximum satisfiability model known as MAX-SAT, and convert the solution into a speech model. This model can be put to many uses, such as planning of fresh dialogs, automatic generation of dialogs etc. In this study, the learnt model is used to predict speech acts in the dialog sequences using the sequence labeling (predicting future acts based on previously seen ones) capabilities of the LSTM (Long Short Term Memory) class of recurrent neural networks. While this work represents a first step in this direction of behavioral autonomy via Automated Planning, it leaves a major scope of improvement as well. While this study has focused on learning speech models, true behavioral autonomy can only be achieved using behavioral models (models which consist of speech acts interleaved with co-verbal gestures). This speech model thus needs to be enriched with gestural information in order to render it complete and realistic. The planning for fresh dialog sequences can only be obtained with a behavioral model under the hood. Devising techniques for further reduction of the learning error between

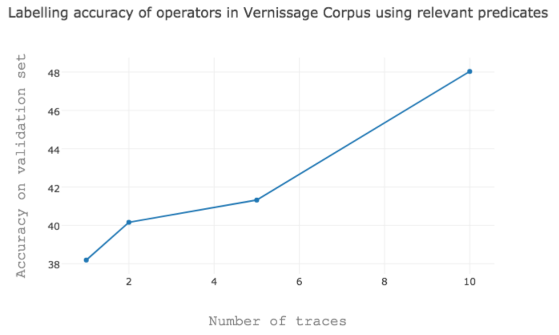


Figure 9: LSTM operator labeling accuracy with relevant predicates as constituents of the feature vector (128 hidden units, 0.8 dropout rate).

the learnt model and ground truth model, and exploring the interleaving between physical and speech acts is in the scope of future work.

References

[1] C. Breazeal, Emotion and sociable humanoid robots, International Journal of Human-Computer Studies, 2003, 59(1), 119-155, https://doi.org/10.1016/S1071-5819(03)00018-1

[2] C. Breazeal, Social interactions in HRI: the robot view, IEEE Transactions on Systems, Man, and Cybernetics, 2004, 34(2), 181-186, https://doi.org/10.1109/TSMCC.2004.826268

[3] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and autonomous systems, 2003, 57(5), 469-483, https://doi.org/10.1016/j.robot.2008.10.024

[4] D. B. Jayagopi, S. Sheikhi, D. Klotz, J. Wienke, J. M. Odobez, S. Wrede el al., The vernissage corpus: A multimodal human-robot-interaction dataset (No. EPFL-REPORT-182715), 2012

[5] C. R. Perrault, An application of default logic to speech act theory, Intentions in Communication, 1990, 161-186

[6] M. D. Sadek, Dialogue acts are rational plans, The Structure of Multimodal Dialogue (Second VENACO Workshop), 1991

[7] J. Riviere, C. Adam, S. Pesty, C. Pelachaud, N. Guiraud, D. Longin et al., Expressive multimodal conversational acts for SAIBA agents, International Workshop on Intelligent Virtual Agents, 2011, 316-323

[8] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature, 2015, 521(7553), 436-444

[9] O. Sigaud, A. Droniou, Towards deep developmental learning, IEEE Transactions on Cognitive and Developmental Systems, 2016, 8(2), 99-114

[10] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation, 1997, 9(8), 1735-1780

[11] S. Jiménez, F. Fernández, D. Borrajo, The PELA architecture: integrating planning and learning to improve execution, Association for the Advancement of Artificial Intelligence, 2008

[12] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data engineering, 2010, 22(10), 1345-1359

[13] H. H. Zhuo, Q. Yang, Action-model acquisition for planning via transfer learning, Artificial Intelligence, 2014, 212, 80-103

[14] R. García-Martínez, D. Borrajo, An integrated approach of learning, planning, and execution, Journal of Intelligent and Robotic Systems, 2000, 29(1), 47-78

[15] Q. Yang, K. Wu, Y. Jiang, Learning action models from plan examples using weighted MAX-SAT, Artificial Intelligence, 2007, 171(2-3), 107-143

[16] H. H. Zhuo, T. A. Nguyen, S. Kambhampati, Refining incomplete planning domain models through plan traces, International Joint Conference on Artificial Intelligence, 2013, 2451-2458

[17] S. Yoon, S. Kambhampati, Towards model-lite planning: A proposal for learning & planning with incomplete domain models, ICAPS 2007 Workshop on Artificial Intelligence Planning and Learning, 2007

[18] H. H. Zhuo, H. Muñoz-Avila, Q. Yang, Learning action models for multi-agent planning, The 10th International Conference on Autonomous Agents and Multiagent Systems, 2011, 1, 217-224

[19] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, International Conference on Very Large Data Bases, 1994, 1215, 487-499

[20] W. K. Fung, Y. H. Liu, Adaptive categorization of ART networks in robot behavior learning using game-theoretic formulation, Neural Networks, 2003, 16(10), 1403-1420

[21] F. Stulp, A. Fedrizzi, L. Mösenlechner, M. Beetz, Learning and reasoning with action-related places for robust mobile manipulation, Journal of Artifial Intelligence, 2012, 43, 1-42

[22] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso et al., PDDL-the planning domain definition language, 1998

[23] M. Brenner, Multiagent planning with partially ordered temporal plans, International Joint Conference on Artificial Intelligence, 2003, 3, 1513-1514

[24] K. Garoufi, A. Koller, Automated planning for situated natural language generation, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010, 1573-1582

[25] T. Marques, M. Rovatsos, Classical planning with communicative actions, European Conference on Artificial Intelligence, 2016, 1744-1745

[26] R. E. Fikes, N. J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence, 1971, 2(3-4), 189-208

[27] H. Mannila, H. Toivonen, A. I. Verkamo, Discovery of frequent episodes in event sequences, Data Mining and Knowledge Discovery, 1997, 1(3), 259-289

[28] C. Dousson, T. V. Duong, Discovering Chronicles with Numerical Time Constraints from Alarm Logs for Monitoring Dynamic Systems, International Joint Conference on Artificial Intelligence, 1999, 99, 620-626

[29] M. Guillame-Bert, J. L. Crowley, Learning temporal association rules on symbolic time sequences, In: Proceedings of the Asian Conference on Machine Learning, PMLR, 2012, 25, 159-174

[30] B. Borchers, J. Furman, A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems, Journal of Combinatorial Optimization, 1998, 2(4), 299-306

[31] H. A. Kautz, B. Selman, Y. Jiang, A general stochastic approach to solving problems with hard and soft constraints, Satisfiability Problem: Theory and Applications, 1996, 35, 573-586