# Using a Recursive Neural Network to Learn an Agent's Decision Model for Plan Recognition

**Francis Bisson** and **Hugo Larochelle** and **Froduald Kabanza**
Département d'informatique
Université de Sherbrooke
Sherbrooke (QC) J1K 2R1, Canada
{*firstname.lastname*}*@usherbrooke.ca*

## Abstract

Plan recognition, the problem of inferring the goals or plans of an observed agent, is a key element of situation awareness in human-machine and machine-machine interactions for many applications. Some plan recognition algorithms require knowledge about the potential behaviours of the observed agent in the form of a plan library, together with a decision model about how the observed agent uses the plan library to make decisions. It is however difficult to elicit and specify the decision model *a priori*. In this paper, we present a recursive neural network model that learns such a decision model automatically. We discuss promising experimental results of the approach with comparisons to selected state-of-the-art plan recognition algorithms on three benchmark domains.

## 1 Introduction

Understanding complex events that unfold in a given situation is an essential aspect of situation awareness in human cognition and intelligent decision-making. For many situations, this requires the ability to infer the intentions, goals and plans of others from the observation of their actions, that is, *plan recognition* [Schmidt, 1976; Baker *et al.*, 2009].

In this paper we consider the case of a plan recognition problem where one observer is trying to infer the goal or plan of one observee, assuming that the observer does not execute actions that influence the decisions of the observee, that is, the *keyhole* plan recognition problem. One of the main frameworks used to solve this problem is to explicitly explore a space of goal or plan hypotheses, generated using a plan library specified *a priori* and consistent with the sequence of observations, in order to select the hypothesis that best explains the observations [Geib and Goldman, 2009; Geib *et al.*, 2008; Kabanza *et al.*, 2013; Sukthankar and Sycara, 2005; Avrahami-Zilberbrand and Kaminka, 2005].

Such plan-library-based approaches assume that the observee indeed behaves by making decisions within the action space circumscribed by the plan library. However, it is difficult in practice to specify *a priori* a library of plans that cover all the potential behaviours, let alone specifying *a priori* how the observee makes decisions using the plan library. The gap between the space of behaviours conveyed by the plan library and the actual decision-making space of the observee characterizes the plan recognizer's accuracy error margin. To reduce this error, one strategy is to learn the plan library and to learn the observee's decision-making model which indicates how they decide to act with the plan library.

In this paper, we describe a solution to the second aspect of the problem. Assuming a given plan library, we cast the problem of learning the probabilistic decision-making model for an agent behaving based upon the library as *multinomial logistic regression*. We train a recursive neural network to learn the vector representation of plan hypotheses, and compute a score for each one of them. We then use a softmax classifier to train the model to yield high scores for correct hypotheses and a low score for incorrect ones to allow ranking the hypotheses by score.

We have compared the approach against a plan-library-based [Geib and Goldman, 2009] and an inverse-planning-based [Ramírez and Geffner, 2010] plan recognition algorithm on three benchmark domains. We obtained better recognition accuracy on all domains. The idea is general, but to be specific we grounded it in a framework that generates plan hypotheses by parsing a sequence of observations using a plan library of hierarchical task networks (HTNs). This framework underlies, for example, the PHATT [Geib and Goldman, 2009], Yappr [Geib *et al.*, 2008] and DOPLAR [Kabanza *et al.*, 2013] algorithms. To simplify the exposure we assume fully observable agents and briefly discuss in the conclusion issues related to relaxing this restriction.

In the next section of this paper, we give a brief state of the art along with some background on the plan recognition algorithms we later compare to. Then, we give a formal description of HTN plan libraries and explain how they are used to generate and explore a space of plan hypotheses that explain a sequence of observations. This sets the stage for a detailed description of the recursive network model that learns the observee's HTN decision-making model. Finally we present and discuss experiments before concluding with some hints on the way forward with our future investigations.

## 2 Related Work

Plan recognition problems in general may involve different multiagent configurations: a single or a team of observer(s), and a single or a team of observee(s). The observer(s) may

have partial or full observational capabilities. The observer(s) and the observee(s) may have competing or shared goals, leading to situations of cooperative or adversarial plan recognition [Kott and McEneaney, 2006]. The observer(s) and the observee(s) may have interactions that affect each other's decisions, leading to situations of game-theoretic plan recognition [Lisý *et al.*, 2012].

Here we focus on one-on-one plan recognition problems, assuming the observer is passive and does not perform actions that might interfere with the observee's decision-making process. A typical plan recognition algorithm in this context requires as input a model conveying background knowledge on the decision-making and acting capabilities of the observee, as well as a sequence of observations. The so-called inverse-planning approaches only require a model of the observee's primitive actions [Ramírez and Geffner, 2010; 2011; Baker *et al.*, 2009]. The so-called plan-library-based approaches in contrast require in addition a plan library, i.e., the recipes the observee may follow to make decisions. Such knowledge can be provided for example as an HTN or a partially-ordered multiset context-free grammar (pomset CFG) [Geib and Goldman, 2009; Kabanza *et al.*, 2013], or as Markov logic formulae [Song *et al.*, 2013].

The idea of recognizing plans by exploring a hypothesis space conveyed by a plan library goes back to the seminal work of [Kautz and Allen, 1986]. Given a plan library in a hierarchy of first-order logic formulae, plan recognition was reduced to finding the minimal cover set of top-level goals in the hierarchy sufficient to explain the observations. Another early seminal work, by [Charniak and Goldman, 1993], also relies on a plan library, but supported probabilistic reasoning.

The HTN-based parsing approach which underlies the PHATT [Geib and Goldman, 2009], Yappr [Geib *et al.*, 2008] and DOPLAR [Kabanza *et al.*, 2013] algorithms have pursued this plan-library-based line of inquiry, by incorporating an explicit probabilistic model of decision-making and execution of the observee, that is, a model about how the observee decides and executes actions with the plan library. This model is key to these approaches being able to chose a plan hypothesis that best explains a sequence of observations.

Unlike plan-library-based approaches, inverse-planning-based approaches [Ramírez and Geffner, 2010; 2011; Baker *et al.*, 2009] don't require any strategic knowledge on how the observee makes decisions. All they need is a set of primitive actions executable by the observee. The intuition behind these approaches is that, given a sequence of observations and a set of potential goals, the goal that is most likely pursued by an agent is the one for which the optimal plan is most consistent with the observations so far. The probability of a goal is thus proportional to the difference between the cost of an optimal plan for that goal consistent with the observations and the cost of an optimal plan for that same goal, but inconsistent with the observations. The computation of this probability requires invoking a planning algorithm twice for each goal and for each observation update. While this might be a source of significant overhead, these approaches have the advantage of accounting implicitly for the entire space of primitive behaviours of the observee rather than the space constrained by a plan library.

[Sohrabi *et al.*, 2011] introduced an approach that is somewhat in-between plan-library-based and inverse-planning-based. Like inverse-planning-based approaches, plan hypotheses for a sequence of observations are generated using an automated planner. The best explanations are found using preferences specified in Past Linear Temporal Logic. Similar to plan libraries, the temporal logic formulae are domain-dependent knowledge that must be specified *a priori*.

The approach of [Wiseman and Shieber, 2014] also supposes the use of traditional planners to generate plan hypotheses, but it uses a machine learning technique to select the best hypotheses. More specifically, a heuristic beam search through the space of valid abductive proofs is used to generate a set of plan hypotheses. Hypotheses are then mapped into feature space based on a list of structural features of abductive proofs, and the best hypothesis is selected by a trained reranker. [Min *et al.*, 2014] recently proposed a goal recognition algorithm that uses a neural network architecture. A fundamental limitation of both approaches is that they require the discriminative features to be specified *a priori* by an expert. Moreover, the features in the work of Wiseman and Shieber are based on the structure of abductive proofs, which the expert must master, not on the domain or the observations.

Our approach in this paper is inspired to some extent by this latter work, but avoids its shortcomings. By using a deep learning technique with a recursive neural network (RNN), our approach is able to learn automatically the meaningful features that enable discrimination between correct and incorrect plan hypotheses while at the same time learning to select the best plan hypothesis. Instead of formulating plan hypotheses as abductive proofs, in this paper we represent them as trees for a sequence of observations using an HTN plan library (that is, a pomset CFG) within the HTN-based parsing framework of PHATT [Geib and Goldman, 2009], Yappr [Geib *et al.*, 2008] and DOPLAR [Kabanza *et al.*, 2013]. In this context, what the RNN learns is indeed the observee's decision-making model, assuming he is behaving by executing an HTN plan library, i.e., the model required by these approaches to select the best plan hypotheses.

There is a sharp contrast between our approach and that of [Min *et al.*, 2014] which also uses a neural network to infer goals from observations. Their approach still requires the features (conveyed by the representation of actions) to be known *a priori* and they use a *feed-foward* neural network. We expect the use of a *recursive* neural network in our approach to enable the learning of more complex behavioural features.

## 3 Goal and Plan Hypotheses Generation through HTN Parsing

Here we summarize the gist of HTN parsing in PHATT [Geib and Goldman, 2009], Yappr [Geib *et al.*, 2008] and DOPLAR [Kabanza *et al.*, 2013]. An HTN plan library is fundamentally a partially-ordered multiset context-free grammar [Kabanza *et al.*, 2013]. Plans generated from an HTN plan library can be understood as partially-ordered strings of actions. Goals are conveyed by root (top-level) tasks. Thus, a HTN describes how to achieve a goal (task) by recursively decomposing it into subgoals (subtasks) down into executable

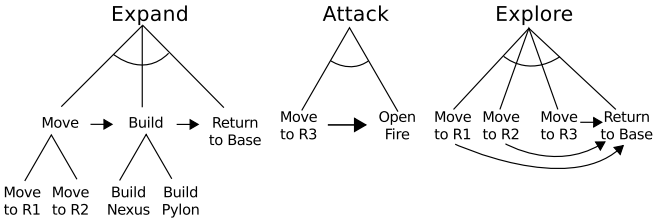



Figure 1: Simple plan library for the StarCraft RTS game [Kabanza *et al.*, 2013]

actions (primitive tasks). Plan hypotheses that explain a sequence of observations are conveyed by derivation trees of the observations.

**Definition 1.** *More formally, a* plan library *is a tuple* $L = (A, G, I, R)$*, where* $A$ *is a finite set of action symbols (terminals),* $G$ *is a finite set of goal symbols (nonterminals),* $I \subseteq G$ *is a set of root goal symbols, and* $R$ *is a set of partially-ordered production rules. Each rule takes the form* $g \rightarrow [\beta; C]$*, where* $g \in G$*,* $\beta \in (A \cup G)^*$ *is a string of terminal and nonterminal symbols, and* $C$ *is a set of ordering constraints of the form* $(i, j)$*, meaning that the* $i^{th}$ *symbol of* $\beta$ *must precede the* $j^{th}$ *symbol of* $\beta$*.*

A rule $g \rightarrow [\beta_0, \ldots, \beta_n; C]$ means that goal $g$ can be accomplished by achieving or executing each of the $\beta_i$ (where $0 \leq i \leq n$) in any order consistent with constraints $C$. There can be many rules with a common left-hand side $g$, representing alternative choices for accomplishing goal $g$. A simple plan library for the StarCraft RTS game is illustrated in Figure 1 [Kabanza *et al.*, 2013].

The hypothesis generation algorithm is an incremental parser over a plan library and a sequence of observation consisting of terminal symbols that produces all hypotheses consistent with the plan library and the observations.

**Definition 2.** *A* hypothesis *explaining a sequence of observations* $\sigma \in A^*$ *for a plan library* $L = (A, G, I, R)$ *is a forest of partial parse trees augmented with ordering constraints. The root symbol of each parse tree must be in the set* $I \subseteq G$*.*

A hypothesis may have several root symbols (parse trees) to account for an observee interleaving the execution of several plans concurrently. A hypothesis accounts both for the *goal* the observee is pursuing and the *plan* (i.e., the interleaving of actions). Each symbol in a hypothesis may also be *observed* or not.

**Definition 3.** *A terminal symbol is* observed *in a parse tree iff it matches an observation from the sequence of observations, otherwise it is* unobserved*. A nonterminal symbol is* observed *iff all of its children in the derivation are also observed; a nonterminal symbol is* unobserved *if it is not yet derived.*

The probability of a generated hypothesis is computed using the observee's *HTN decision model*, which is in fact a probability distribution over all decision choices conveyed by the HTN plan library: a distribution about how the agent chooses or commit to goals (*a priori* probabilities for root goals), a distribution about how the agent chooses among alternatives in the HTN hierarchy, and a distribution about how the agent interleaves independent or concurrent actions (referred to as choice in the pending set in Yappr and PHATT).

Defining the HTN decision model is one of the key limitations of PHATT, Yappr and DOPLAR. By default, these algorithms use a uniform probability distribution for the different choices. [Geib and Goldman, 2009] have explained the rationale for a uniform distribution by default, while also recognizing the importance of developing methods that can learn a more accurate HTN decision model.

## 4 Recursive Neural Network Approach

Given the rich structure of the HTN decision model (probability choices over a recursive HTN), we set out to investigate the use of a recursive neural network (RNN) to learn it automatically. It has two main components: the RNN architecture, which yields a learned feature representation of hypotheses, and the logistic regression scorer, which uses the RNN representation in order to assign probabilities to hypotheses.

### 4.1 RNN Feature Representation of Hypotheses

An RNN in general is a type of multilayer neural network that learns the representation of recursive, hierarchical structures, such as the hypotheses we consider in this paper. RNNs can map such structures into a vectorial feature space by recursively merging the learned feature representation of the components of the structure, from its leaves towards its root. Although there exist various metrics for comparing trees that could potentially be used for ranking explanations, our approach goes beyond simple ranking and learns directly the observee's HTN decision model, which accurately predicts how the agent chooses goals and actions.

RNNs have been used before in a number of applications that involve complex syntactic structures, such as natural language sentences and scene image parsing [Socher *et al.*, 2011] and describing images with sentences [Socher *et al.*, 2014]. However, the models used to date could only predict the binary tree hierarchical structure of natural language sentences and scene images [Socher *et al.*, 2011], or only consider the relative position of nodes in the tree [Socher *et al.*, 2014]. As such, they are not sufficient for learning an HTN decision model.

The RNN architecture that we describe in this paper captures the structural and semantic characteristics of an HTN decision model, which also underlie plan hypotheses, in particular the ordering constraints as well as the distinction between observed and unobserved symbols. The intuition is that these properties are necessary to discriminate between correct and incorrect hypotheses.

In a nutshell, given one partial parse tree in a hypothesis, the RNN first represents its leaves using feature vectors that are learned for each of the leaf symbols. Then, it computes the feature representation of each component higher up in the parse tree, as a nonlinear (*merging*) transformation of each component's children, much like the hidden layer of a feed-forward neural network. Finally, a representation for the whole hypothesis is obtained by merging the representation of the roots of all parse trees in the hypothesis.

More specifically, let vector $\mathbf{h}_\theta(x)$ be the feature vector representation of symbol $x$ in a parse tree, where $\theta =$

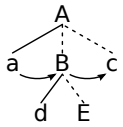



Figure 2: Simple hypothesis with 1 root symbol. Lowercase symbols are terminals, and uppercase symbols are nonterminals. A symbol with a dashed line above it is unobserved.

$(\mathbf{R}, \mathbf{W}, \mathbf{U}, \mathbf{V}, \mathbf{Q}, \mathbf{b}, \mathbf{d})$ are the RNN's parameters. Let $n$ also be the size of these vectors, i.e., the number of features. For $x$ corresponding to a leaf symbol (either a terminal or a nonterminal), $\mathbf{h}_\theta(x)$ is simply the corresponding column $\mathbf{R}_x$ in a learned representation matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, where $m$ is the total number of terminal and nonterminal symbols in the plan library. Then, each non-leaf representation $\mathbf{h}_\theta(x)$ is computed from the children of $x$ as follows:

$$\mathbf{h}_\theta(x) = g\Bigg(\mathbf{b} + \frac{1}{w}\sum_{i=1}^{w} \mathbf{W}\,\mathbf{h}_\theta(x_i) + \frac{1}{u}\sum_{k=1}^{u} \mathbf{U}\,\mathbf{h}_\theta(x_k) + \frac{1}{v}\sum_{j=1}^{v} \mathbf{V}\,\mathbf{c}_{j,\theta}(x)\Bigg), \tag{1}$$

where:

- $\mathbf{W} \in \mathbb{R}^{n \times n}$ and $\mathbf{U} \in \mathbb{R}^{n \times n}$ are the connection weights used to model the contribution of observed and unobserved symbols, respectively, and $\mathbf{V} \in \mathbb{R}^{n \times 2n}$ is used for the contribution of ordering constraints;
- $\mathbf{b} \in \mathbb{R}^n$ is a bias vector;
- $\mathbf{c}_{j,\theta}(x) = [\mathbf{h}_\theta(x_{j_{\text{left}}}), \mathbf{h}_\theta(x_{j_{\text{right}}})]^\top$ is the vector representation of the $j^{\text{th}}$ ordering constraint obtained by concatenating the representations of the left and right symbols of the constraint; and
- $g$ is any sigmoid-like function, $w$ is the number of observed children symbols under $x$, $u$ is the number of unobserved children under $x$, and $v$ is the number of ordering constraints under $x$.

For example, obtaining the feature vector representation of all components in the parse tree rooted at $A$ in Figure 2 is illustrated in Figure 3 and is computed like so:

$$\mathbf{h}_\theta(A) = g\Bigg(\mathbf{b} + \frac{1}{1}\Big(\mathbf{W}\,\mathbf{h}_\theta(a)\Big) + \frac{1}{2}\Big(\mathbf{U}\,\mathbf{h}_\theta(B) + \mathbf{U}\,\mathbf{h}_\theta(c)\Big) + \frac{1}{2}\Big(\mathbf{V}\,[\mathbf{h}_\theta(a), \mathbf{h}_\theta(B)]^\top + \mathbf{V}\,[\mathbf{h}_\theta(B), \mathbf{h}_\theta(c)]^\top\Big)\Bigg)$$

$$\mathbf{h}_\theta(B) = g\Bigg(\mathbf{b} + \frac{1}{1}\Big(\mathbf{W}\,\mathbf{h}_\theta(d)\Big) + \frac{1}{1}\Big(\mathbf{U}\,\mathbf{h}_\theta(E)\Big)\Bigg)$$

$$\mathbf{h}_\theta(a) = \mathbf{R}_a, \;\; \mathbf{h}_\theta(c) = \mathbf{R}_c, \;\; \mathbf{h}_\theta(d) = \mathbf{R}_d, \;\; \mathbf{h}_\theta(E) = \mathbf{R}_E$$

Finally, to obtain the feature representation of the whole hypothesis $\eta$ (which can consist of $t \geq 1$ parse trees), we perform a similar merging transformation as for parse tree components, as follows:

$$\mathbf{h}_\theta(\eta) = g\left(\mathbf{d} + \sum_{i=1}^{t} \mathbf{Q}\,\mathbf{h}_\theta(root_i)\right), \tag{2}$$

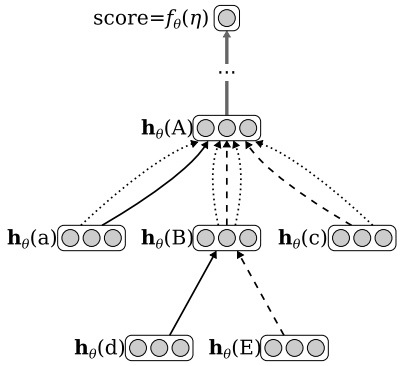



Figure 3: Computing the representation and the score of a hypothesis with the recursive neural network. Grey circles represent learned features of the hypothesis. Arrows between the representation of symbols indicate which parameter matrix is being used: solid for $\mathbf{W}$, dashed for $\mathbf{U}$, dotted for $\mathbf{V}$, and bold solid grey for the score vector $\mathbf{z}$. Note that matrix $\mathbf{V}$ is used twice between symbols $A$ and $B$, since $B$ is present in two ordering constraints under $A$ ($a \rightarrow B$ and $B \rightarrow c$).

where the $root_i$ are the root goal symbols of each of the $t$ parse trees in hypothesis $\eta$, and $\mathbf{Q} \in \mathbb{R}^{m \times n}$ and $\mathbf{d} \in \mathbb{R}^n$ are also parameters of the RNN.

### 4.2 Scoring and Ranking Hypotheses

As stated earlier, we cast the problem of identifying the correct hypotheses as multinomial logistic regression. Our regression model works as follows.

First, for each hypothesis $\eta$ in a set of candidates $H$, we compute a score $f$ by multiplying the feature vector representation of the hypothesis by a score parameter vector $\mathbf{z}$ and applying a sigmoid nonlinearity:

$$f_\theta(\eta) = \text{sigm}\left(\mathbf{z}^\top \mathbf{h}_\theta(\eta)\right) \tag{3}$$

Then, we simply model the probability that $\eta \in H$ is the correct hypothesis as:

$$p(\eta \mid H) = \frac{\exp(f_\theta(\eta))}{\sum_{\eta' \in H} \exp(f_\theta(\eta'))} \tag{4}$$

After training, these probabilities form the observee's HTN decision model and can be used to rank hypotheses.

Training is performed jointly over both the RNN parameters $\theta$ and the score parameter vector $\mathbf{z}$. Specifically, given a training set consisting of pairs $(H, y)$, where $\eta_y \in H$ is the correct hypothesis (see Section 5.1 for how training sets are generated), we minimize the negative log-likelihood loss for each training example:

$$l(f_\theta, H, y) = -\log\left[\frac{\exp(f_\theta(\eta_y))}{\sum_{\eta' \in H} \exp(f_\theta(\eta'))}\right] \tag{5}$$

The model does not currently include a regularizer term to counter overfitting, but this does not appear to be an issue at the moment (see Section 5).

We train the model with stochastic gradient descent through backpropagation, a common method for training neural networks [Rumelhart *et al.*, 1986]. The gradient of the loss

with respect to the score function of each hypothesis $\eta_i \in H$ is simply the following:

$$\nabla_{f_\theta(\eta_i)} l(f_\theta, H, y) = -\left(1_{(i=y)} - f_\theta(\eta_i)\right) \quad (6)$$

The other gradients are straightforward to derive using the chain rule. Before training the model, all parameters $\theta$ and $\mathbf{z}$ are initialized with random values, save for bias vectors $\mathbf{b}$ and $\mathbf{d}$ which are initialized to zeroes.

# 5 Evaluation

We implemented our algorithm in Python using the NumPy library for efficient linear algebra operations, and compared the ranking results on three domains against Yappr [Geib *et al.*, 2008] and the inverse planning approach of [Ramírez and Geffner, 2010].

## 5.1 Datasets

We evaluated our algorithm on three synthetic benchmark domains, each with different characteristics:

- **Monroe plan corpus** This domain consists of plan sessions automatically generated by an AI planner in a disaster management domain [Blaylock and Allen, 2005]. We slightly altered the original domain to remove left recursion and epsilon productions in the grammar, as well as action parameters. The resulting plan library has 30 action symbols, 43 goal symbols, 10 root goal symbols, and 91 production rules.
- **StarCraft navigation (SCN)** This domain consists of plans in the StarCraft real-time strategy game, for example attacking an enemy base, or defending an allied location[1] [Kabanza *et al.*, 2013]. The observed actions correspond to groups of units moving around the game map. The plan library has 58 action symbols, 134 goal symbols, 7 root goal symbols, and 258 production rules.
- **Kitchen** This domain consists of activities of daily living that take place in a kitchen, for instance making tea and packing a lunch bag [Wu *et al.*, 2007]. The observed actions correspond to taking and using various objects of the kitchen to perform the activities (e.g., taking a kettle to make tea). The plan library has 25 action symbols, 16 goal symbols (all of them are also possible root goal symbols), and 29 production rules.

For each domain, we generated sequences of observations by randomly selecting the root goals the observee would take as well as the actions it would take to accomplish said goals, forming the ground truth. The sequences of observations as well as the ground truth were already provided with the Monroe plan corpus. We assumed uniform distributions to generate the SCN and Kitchen scenarios. The goals in the SCN domain are conjunctions of 3 randomly selected root goal symbols (for a total of 343 possible combinations), and only one root goal symbol in the Monroe and Kitchen domains.

We generated hypotheses consistent with the plan library and prefixes of these sequences of observations, and labelled them as correct or incorrect according to the ground truth. We then randomly selected sets of incorrect hypotheses and added the correct hypothesis to each set, forming ranking examples for our training, validation and test sets. Although all correct and incorrect hypotheses are consistent with the plan library, we could also envision different experiments involving incorrect hypotheses that are inconsistent with the plan library. We expect that the RNN would still be able to learn under these conditions. Such experiments are for future investigations. Table 1 gives the total number of hypotheses used in each set for each of the three domains. The low number of hypotheses in the Kitchen domain is simply a product of the plan library, which produces few hypotheses consistent with any given sequence of observations.

| Set | Monroe | SCN | Kitchen |
|---|---|---|---|
| Training | 50 275 | 70 900 | 13 263 |
| Validation | 17 225 | 23 125 | 4534 |
| Test | 17 100 | 21 850 | 4723 |

Table 1: Number of hypotheses in each dataset

We trained the RNN with several combinations of hyperparameter values for the number of features $n$, the learning rate $\alpha$, and the sigmoid or hyperbolic tangent ($\tanh$) functions as the $g$ nonlinearity in Equations (1) and (2). We used early stopping with a lookahead of 5 epochs to find the optimal number of training epochs based on the classification error of the validation set, measured as the ratio of incorrectly identified hypotheses. Table 2 reports the best combinations of hyperparameters we found empirically as well as the resulting classification error on the test set.

| | Monroe | SCN | Kitchen |
|---|---|---|---|
| $\alpha$ | 0.01 | 0.01 | 0.01 |
| $n$ | 25 | 15 | 25 |
| $g$ | $\tanh$ | sigm | $\tanh$ |
| Error | 0.101 | 0.052 | 0.031 |

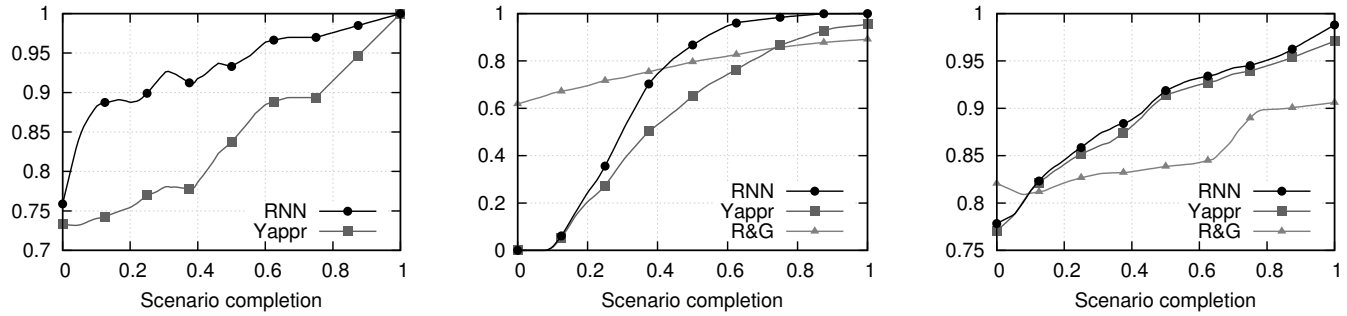
Table 2: Hyperparameter values and the classification error

Note that two hypotheses in the datasets could be labelled differently since one could be the correct hypothesis in a scenario but not in another, depending on context. Nevertheless, the recursive neural network model managed to obtain a good classification precision on the test set of each domain.

## 5.2 Ranking Experiment

We measured the rank of the observee's goal in all three algorithms. With the RNN and Yappr, we generated all hypotheses consistent with the observations in a number of scenarios (145 in the Monroe domain, 77 in the SCN domain, and 250

[1] Although this is an adversarial setting, we can test simplified scenarios in a non-interactive context by assuming that the observer uses a plan recognizer to infer the goals of the observee on very short horizons along which it can be reasonably expected that the observee will not adjust its decisions. Decision adjustments on longer horizons are somewhat implicitly accounted for by invoking the plan recognizer after every new observation, causing a reevaluation of the inferred goals. This of course only remains an approximation of a plan recognition problem which should explicitly take into account the interactions between the agents as some authors have attempted to do, albeit with limited efficiency so far [Lisý *et al.*, 2012].




(a) Monroe plan corpus (b) StarCraft navigation (c) Kitchen

Figure 4: Average rank of the correct goal by scenario completion on three selected domains

in the Kitchen domain). We then measured the rank of the correct goal after each observation, where the score of a goal is the maximum probability of all hypotheses with this exact goal. The scenarios used in this experiment did not serve to generate the datasets mentioned in the previous section. The comparative results in each of the three domains are plotted in Figure 4.

We can see from the graphs that the RNN performs best on all three domains, with the smallest gap in the Kitchen domain. We can expect this gap to grow larger with more training data (see Table 1). We also assumed a uniform decision model to generate the SCN and Kitchen scenarios, which explains why Yappr works well since it assumes a uniform decision model by default. In future experiments, we will use nonuniform decision models to generate scenarios to better highlight the advantages of learning the model with an RNN without making any prior assumptions[2].

The shape of the RNN and Yappr curves on the SCN domain is symptomatic of the way the parser generates hypotheses incrementally: the score (or probability) of the correct goal will always be zero until there are 3 root symbols in the hypothesis (forest of partial parse trees). R&G does not have this problem, so it initially fares much better than the RNN and Yappr, but falls short of the ranking accuracy we obtain past the first 40% observations. R&G performs the least well on the Kitchen domain, in part because it failed to find a plan before the 5-minute timeout on several scenarios (other scenarios would only take a few milliseconds to find plans for).

We did not provide results for R&G on the Monroe domain because a PDDL domain was not available. Although there exist methods to translate HTNs into PDDL [Alford *et al.*, 2009], the modelling effort required to convert and optimize the resulting PDDL domain is beyond the scope of this paper.

Note that we did not compare against DOPLAR [Kabanza *et al.*, 2013] because the only thing it improves over Yappr is timeliness by limiting the number of generated hypotheses. Although we do not report timeliness results in this paper, scoring and ranking hypotheses with the RNN is not computationally costly (see Equations (1)–(4)). We also did not compare against the ranking approach of [Wiseman and Shieber, 2014] simply because we would have had to manually specify and develop structural features of hypotheses modelled as parse forests, which is exactly what we set out to avoid by learning the features automatically with an RNN.

[2]It is important to note that the probability distributions underlying the HTN decision model are unique to Yappr and DOPLAR, and have nothing to do with R&G. Only the probability distribution for goal predictions have the same interpretation in both frameworks, i.e., the probability of a goal given a sequence of observations.

## 6 Conclusion

In this paper we presented a recursive neural network model that learns the HTN decision-making model of the observed agent for plan recognition. Our model learns the feature vector representation of hypotheses as well as their score, which are then used for ranking. We evaluated our approach on three domains and compared it against Yappr and R&G, two approaches among state-of-the-art plan recognition algorithms, and the results are very promising.

We intend to pursue investigating this approach by comparing it against other plan recognition algorithms, such as those based on Markov logic [Song *et al.*, 2013], and on more complex scenarios in real-world domains. We believe that our approach is general in that a recursive neural network could learn other hierarchical representations than our hypotheses, such as the abductive proofs of [Wiseman and Shieber, 2014], or the hypotheses generated by the SBR algorithm [Avrahami-Zilberbrand and Kaminka, 2005].

We will also investigate relaxing some of the assumptions we have made, in particular the full observational capabilities. Since we use the same hypothesis generation framework as PHATT, Yappr and DOPLAR, we can use the extension proposed by [Geib and Goldman, 2005] as a starting point, although it is not scalable in practice. Finding better solutions to deal with this problem is one of our objectives.

## Acknowledgements

This work was supported by the Natural Science and Engineering Research Council (NSERC) of Canada and the *Fonds de recherche du Québec – Nature et technologies* (FRQNT). We are grateful to the reviewers whose comments helped improve the paper. We also thank Julien Filion for answering questions on the DOPLAR algorithm.

## References


[Alford *et al.*, 2009] R. Alford, U. Kuter, and D. Nau. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI)*, pages 1629–1634, 2009.

[Avrahami-Zilberbrand and Kaminka, 2005] D. Avrahami-Zilberbrand and G. A. Kaminka. Fast and complete symbolic plan recognition. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 653–658, 2005.

[Baker *et al.*, 2009] C. L. Baker, R. Saxe, and J. B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113:329–349, 2009.

[Blaylock and Allen, 2005] N. Blaylock and J. Allen. Generating artificial corpora for plan recognition. In L. Ardissono, P. Brna, and A. Mitrovic, editors, *User Modeling 2005*, volume 3538 of *Lecture Notes in Computer Science*, pages 179–188. Springer, July 2005.

[Charniak and Goldman, 1993] E. Charniak and R. P. Goldman. A bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, November 1993.

[Geib and Goldman, 2005] C. W. Geib and R. P. Goldman. Partial observability and probabilistic plan/goal recognition. In *Proceedings of the IJCAI Workshop on Modeling Others from Observations (MOO)*, 2005.

[Geib and Goldman, 2009] C. W. Geib and R. P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 117(11):1101–1132, 2009.

[Geib *et al.*, 2008] C. W. Geib, J. Maraist, and R. P. Goldman. A new probabilistic plan recognition algorithm based on string rewriting. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 91–98, 2008.

[Kabanza *et al.*, 2013] F. Kabanza, J. Filion, A. R. Benaskeur, and H. Irandoust. Controlling the hypothesis space in probabilistic plan recognition. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[Kautz and Allen, 1986] H. A. Kautz and J. F. Allen. Generalized plan recognition. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI)*, pages 32–37, 1986.

[Kott and McEneaney, 2006] A. Kott and W. M. McEneaney, editors. *Adversarial Reasoning: Computational Approaches to Reading The Opponent's Mind*. Chapman & Hall/CRC, 2006.

[Lisý *et al.*, 2012] V. Lisý, R. Píbil, J. Stiborek, B. Bošanský, and M. Pěchouček. Game-theoretic approach to adversarial plan recognition. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, pages 546–551, 2012.

[Min *et al.*, 2014] W. Min, E. Y. Ha, J. Rowe, B. Mott, and J. Lester. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of the 10th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 37–43, 2014.

[Ramírez and Geffner, 2010] M. Ramírez and H. Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, pages 1121–1126, 2010.

[Ramírez and Geffner, 2011] M. Ramírez and H. Geffner. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2009–2014, 2011.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[Schmidt, 1976] C. F. Schmidt. Understanding human action: Recognizing the plans and motives of other persons. In J. S. Carroll and J. W. Payne, editors, *Cognition and Social Behavior*, Carnegie-Mellon University Cognition, pages 47–68. Lawrence Erlbaum Associates, 1976.

[Socher *et al.*, 2011] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 129–136, 2011.

[Socher *et al.*, 2014] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

[Sohrabi *et al.*, 2011] S. Sohrabi, J. A. Baier, and S. A. McIlraith. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*, pages 261–267, 2011.

[Song *et al.*, 2013] Y. C. Song, H. Kautz, J. Allen, M. Swift, Y. Li, J. Luo, and C. Zhang. A Markov logic framework for recognizing complex events from multimodal data. In *Proceedings of the 15th ACM International Conference on Multimodal Interaction (ICMI)*, 2013.

[Sukthankar and Sycara, 2005] G. Sukthankar and K. Sycara. A cost minimization approach to human behavior recognition. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1067–1074, 2005.

[Wiseman and Shieber, 2014] S. Wiseman and S. Shieber. Discriminatively reranking abductive proofs for plan recognition. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

[Wu *et al.*, 2007] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg. A scalable approach to activity recognition based on object use. In *Proceedings of the 11th International Conference on Computer Vision (ICCV)*, 2007.