Learning Models of Human Behaviour from Textual Instructions

Kristina Yordanova and Thomas Kirste
University of Rostock, 18051 Rostock, Germany

Keywords: Precondition-effect Rules, Models of Human Behaviour, Natural Language Processing, Activity Recognition.

Abstract: There are various activity recognition approaches that rely on manual definition of precondition-effect rules to describe human behaviour. These rules are later used to generate computational models of human behaviour that are able to reason about the user behaviour based on sensor observations. One problem with these approaches is that the manual rule definition is time consuming and error prone process. To address this problem, in this paper we propose an approach that learns the rules from textual instructions. In difference to existing approaches, it is able to learn the causal relations between the actions without initial training phase. Furthermore, it learns the domain ontology that is used for the model generalisation and specialisation. To evaluate the approach, a model describing cooking task was learned and later applied for explaining seven plans of actual human behaviour. It was then compared to a hand-crafted model describing the same problem. The results showed that the learned model was able to recognise the plans with higher overall probability compared to the hand-crafted model. It also learned a more complex domain ontology and was more general than the hand-crafted model. In general, the results showed that it is possible to learn models of human behaviour from textual instructions which are able to explain actual human behaviour.

1 INTRODUCTION

Assistive systems support the daily activities and allow even people with impairments to continue their independent life (Hoey et al., 2010). Such systems have to recognise the user actions and intentions, track the user interactions with a variety of objects, detect errors in the user behaviour, and find the best way of assisting them (Hoey et al., 2010). This can be done by activity recognition (AR) approaches that utilise human behaviour models (HBM) in the form of rules. These rules are used to generate probabilistic models with which the system can infer the user actions and goals (Krüger et al., 2014; Hiatt et al., 2011; Ramirez and Geffner, 2011). Such types of models are also known as computational state space models (CSSM) (Krüger et al., 2014). They treat activity recognition as a plan recognition problem, where given an initial state, a set of possible actions, and a set of observations, the executed actions and the user goals have to be recognised (Ramirez and Geffner, 2011). These approaches rely on prior knowledge to obtain the context information needed for building the user actions and the problem domain. The prior knowledge is provided in the form of precondition-effect rules by a domain expert or by the model designer. This knowledge is then used to manually build a CSSM. The manual modelling is however time consuming and error prone (Nguyen et al., 2013; Krüger et al., 2012).

To address this problem, different works propose the learning of models from sensor data (Zhuo and Kambhampati, 2013; Okeyo et al., 2011). One problem these approaches face is that sensor data is expensive (Ye et al., 2014). Furthermore, sensors are sometimes unable to capture fine-grained activities (Chen et al., 2012), thus, they might potentially not be learned.

To reduce the need of domain experts and / or sensor data, one can substitute them with textual data (Philipose et al., 2004). More precisely, one can utilise the knowledge encoded in textual instructions to learn the model structure. Textual instructions specify tasks for achieving a given goal without explicitly stating all the required steps. On the one hand, this makes them a challenging source for learning a model (Branavan et al., 2010). On the other hand, they are usually written in imperative form, have a simple sentence structure, and are highly organised. Compared to rich texts, this makes them a better source for identifying the sequence of actions needed for reaching the goal (Zhang et al., 2012).

According to (Branavan et al., 2012), to learn a model of human behaviour from textual instructions,

Yordanova, K. and Kirste, T.
Learning Models of Human Behaviour from Textual Instructions.
DOI: 10.5220/0005755604150422
In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) - Volume 2*, pages 415-422
ISBN: 978-989-758-172-4

the system has to: 1. **extract the actions' semantics** from the text, 2. **learn the model semantics** through language grounding, 3. and, finally, to **translate it into computational model of human behaviour** for planning problems. To address the problem of learning models of human behaviour for AR, we extend the steps proposed by (Branavan et al., 2012). We add the need of 4. **learning the domain ontology** that is used to abstract and / or specialise the model. Furthermore, we consider **computational models for activity recognition** as the targeted model format, as they represent the problem in the form of a planning problem and are able to reason about the human behaviour based on observations (Hiatt et al., 2011; Ramirez and Geffner, 2011).

In this work we concentrate on the problem of (1) learning the precondition-effect rules that describe the human behaviour; (2) learning the domain ontology that describes the context information and its semantic structure; and (3) the ability of the learned models to explain real human behaviour in the form of plans.

2 RELATED WORK

There are various approaches to learning models of human behaviour from textual instructions: through grammatical patterns that are used to map the sentence to a machine understandable model of the sentence (Zhang et al., 2012; Branavan et al., 2012); through machine learning techniques (Sil and Yates, 2011; Chen and Mooney, 2011; Kollar et al., 2014); or through reinforcement learning approaches that learn language by interacting with an external environment (Branavan et al., 2012; Branavan et al., 2010; Kollar et al., 2014).

Models learned through model grounding have been used for plan generation (Li et al., 2010; Branavan et al., 2012), for learning the optimal sequence of instruction execution (Branavan et al., 2010), for learning navigational directions (Chen and Mooney, 2011), and for interpreting human instructions for robots to follow them (Kollar et al., 2014; Tenorth et al., 2010). To our knowledge, any attempts to apply language grounding to learning models for AR rely on identifying objects from textual data and do not build a computational model of human behaviour (Perkowitz et al., 2004; Ye et al., 2014). This, however, suggests that models learned from text could be used for AR tasks. AR here is treated as a plan recognition problem, thus the plan elements have to be learned. Existing approaches that learn human behaviour from text make simplifying assumptions about the learning problem, making them unsuitable for more general AR problems. More precisely, the preconditions and effects are learned through explicit causal relations, that are grammatically expressed in the text (Li et al., 2010; Sil and Yates, 2011). They however, either rely on initial manual definition to learn these relations (Branavan et al., 2012), or on grammatical patterns and rich texts with complex sentence structure (Li et al., 2010). They do not address the problem of discovering causal relations between sentences, but assume that all causal relations are expressed within the sentence (Tenorth et al., 2010). They also do not identify implicit relations. However, to find causal relations in instructions without a training phase, one has to rely on alternative methods, such as time series analysis (Yordanova, 2015a). Furthermore, they rely on manually defined ontology, or do not use one. However, one needs an ontology to deal with model generalisation problems and as a means for expressing the semantic relations between model elements.

Moreover, there have been previously no attempts at learning CSSMs from textual instructions. Existing CSSM approaches rely on manual rules definition to build the preconditions and effects of the models. For example, (Hiatt et al., 2011) use the cognitive architecture ACT-R, a sub-symbolic production system. It allows the manual description of actions in terms of preconditions and effects, while the state of the world is modelled as information chunks that can be retrieved from the memory of the system. Other approaches rely on a PDDL[1]-like notations to describe the possible actions (Ramirez and Geffner, 2011; Krüger et al., 2013). Then, based on a set of observations, the agent's actions and goals are recognised.

In this work we represent the learned CSSM in a PDDL-like notation and use the learned model to explain plans that describe actual human behaviour.

3 APPROACH

3.1 Identifying Text Elements of Interest

To extract the text elements that describe the user actions, their causal relations to other entities and the environment have to be identified. This is achieved through assigning each word in a text the corresponding part of speech (POS) tag. Furthermore, the dependencies between text elements are identified through dependencies parser.

[1]**P**lanning **D**omain **D**efinition **L**anguage

To identify the human actions, the verbs from the POS-tagged text are extracted. We are interested in present tense verbs, as textual instructions are usually written in present tense, imperative form.

After identifying the actions, we extract any nouns that are direct objects to the actions. These will be the objects in the environment with which the human can interact. Furthermore, we extract any nouns that are in conjunction to the identified objects. These will have dependencies to the same actions, to which the objects with which they are in conjunction are dependent.

Moreover, any preposition relations such as *in*, *on*, *at*, etc. between the objects and other elements in the text are identified. These provide spacial or directional information about the action of interest. For example, in the sentence *"Put the apple on the table."* our action is *put*, while the object on which the action is executed is *apple*. The action is executed in the location *table* identified through the *on* preposition.

Finally, we extract "states" from the text. The state of an object is the adjectival modifier or the nominal subject of an object. As in textual instructions the object is often omitted (e.g. *"Simmer (the sauce) until thickened."*), we also investigate the relation between an action and past tense verbs or adjectives that do not belong to an adjectival modifier or to nominal subject, but that might still describe this relation. The states give us information about the state of the environment before and after an action is executed.

3.2 Extracting Causal Relations from Textual Instructions

To identify causal relations between the actions, and between states and actions, we use an approach proposed in (Yordanova, 2015a). It transforms every word of interest in the text into a time series and then applies time series analysis to identify any causal relations between the series. More precisely, each sentence is treated as a time stamp in the time series. Then, for each word of interest, the number of occurrences it appears in the sentence is counted and stored as element of the time series with the same index as the sentence index.

Generally, we can generate a time series for each kind of word, as well as for each tuple of words. Here we concentrate on those describing or causing change in a state. That means we generate time series for all actions and for all states that change an object.

To discover causal relations based on the generated time series, we apply the Granger causality test. It is a statistical test for determining whether one time series is useful for forecasting another. More precisely, Granger testing performs statistical significance test for one time series, "causing" the other time series with different time lags using auto-regression (Granger, 1969). The causality relationship is based on two principles. The first is that the cause happens prior to the effect, while the second states that the cause has a unique information about the future values of its effect. Based on these assumptions, given two sets of time series x_t and y_t, we can test whether x_t Granger causes y_t with a maximum p time lag. To do that, we estimate the regression $y_t = a_o + a_1 y_{t-1} + \ldots + a_p y_{t-p} + b_1 x_{t-1} + \ldots + b_p x_{t-p}$. An F-test is then used to determine whether the lagged x terms are significant.

3.3 Building the Domain Ontology

The domain ontology is divided into argument (object) and action ontology. The argument ontology describes the objects, locations, and any other elements in the environment that are taken as arguments in the actions and predicates. The action ontology represents the actions with their arguments and abstraction levels.

To learn the argument ontology, a semantic lexicon (e.g. WordNet (Miller, 1995)) is used to build the initial ontology. As the initial ontology does not contain some types that unify arguments applied to the same action (see Fig. 1), the ontology has to be extended. To do that, the prepositions with which actions are connected to indirect objects are also extracted (e.g. *in*, *on*, etc.). They are then added to the arguments ontology as parents of the arguments they connect. In that manner the locational properties of the arguments are described (e.g. *water* has the property to be *in* something). During the learning of the action templates and their preconditions and effects (see Section 3.4), additional parent types are added to describe objects used in actions that have the same preconditions. Furthermore, types that are not present in the initial ontology, but which objects are used only in a specific action, are combined in a common parent type. Fig. 1 shows an example of an argument ontology and the learning process.

To learn the action ontology, the process for building action ontology proposed in (Yordanova and Kirste, 2015) is adapted for learning from textual data. More precisely, based on the argument ontology, the actions are abstracted by replacing the concrete arguments with their corresponding types from an upper abstraction level. In that manner, the uppermost level will represent the most abstract form of the action. For example, the sentence *"Put the apple on the table."* will yield the concrete action *put_apple_table*, and the abstract action *put_object_location*. This

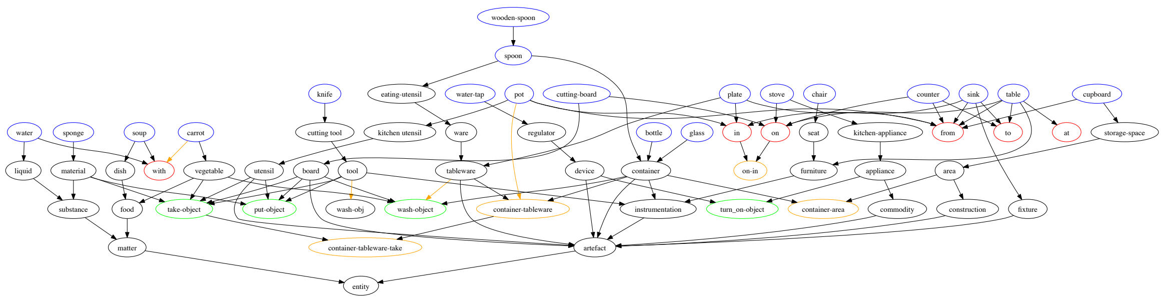

Figure 1: Learning the argument ontology. Step 1 (blue): objects identified through POS-tagging and dependencies; step 2 (black): hierarchy identified through WordNet; step 3 (red): types identified through the relations of objects to prepositions; step 4 (green): types identified based on similar preconditions; step 5 (yellow): types identified through action abstraction.

representation will later be used as a basis for the precondition-effect rules that describe the actions.

3.4 Generating Precondition-effect Rules

Having identified the actions, their causal relations, and the domain ontology, the last step is the generation of precondition-effect rules that describe the actions and the way they change the world. The basis for the rules is the action ontology. Each abstract action from the ontology is taken and converted to an action template that has the form shown in Fig. 2. Ba-

```
(:action put
  :parameters (?o - object ?to - location)
  :precondition (and
          (not (executed-put ?o ?to)))
  :effect   (and
          (executed-put ?o ?to))
)
```

Figure 2: Example of an action template *put* in the PDDL notation.

sically, the action name is the first part of the abstract entity *put_object_location*, while the two parameters are the second and the third part of the entity. Furthermore, the default predicate *(executed-action)* is added to both the precondition and the effect, whereas in the precondition it is negated.

Now the causal relations extracted from the text are used to extend the actions. The execution of each action that was identified to cause another action is added as a precondition to the second action. For example, to execute the action *put*, the action *take* has to take place. That means that the predicate *executed-take ?o* has to be added to the precondition of the action *put*.

Furthermore, any states that cause the action are also added in the precondition. For example, imagine the example sentence is extended in the following manner: *"If the apple is ripe, put the apple on the table."* In that case the state *ripe* causes the action *put*. For that reason the predicate *(state-ripe)* will also be added to the precondition.

This procedure is repeated for all available actions. The result is a set of candidate rules that describe a given behaviour.

As it is possible that some of the rules contradict each other, a refinement step is added. This is done by converting the argument ontology to the corresponding PDDL format to represent the type hierarchy of the problem. Then a concrete problem is manually provided. It describes the initial state of the world and the goal state to be reached. Later, the problem as well as the rules and the type hierarchy are fed to a general purpose planner and any predicates that prevent the reaching of the goal are removed from the preconditions.

4 EXPERIMENTAL SETUP

To evaluate the approach, textual instructions describing a kitchen experiment were used to generate precondition-effect rules. The instructions were obtained from the annotation of an activity recognition experiment where a person prepares a carrot soup, then serves the meal, has lunch, then cleans the table (Krüger et al., 2015; Krüger et al., 2014). The instructions consisted of 80 sentences, with an average sentence length of 6.1 words, and an average of 1 action per sentence. The instructions were parsed with the Stanford Parser to obtain the POS-tags and the dependencies. They were then used as an input for identifying the model elements and for generating the time series. The time series were then tested for stationarity by using the Augmented Dickey–Fuller (ADF) t-statistic test. It showed that the series are already stationary. The generated time series are available at the University Library of the University of Rostock (Yordanova, 2015b). Using the Granger causality test, 18 causal relations were discovered. They were then used as an input for building the precondition-effect rules. Furthermore, WordNet was used to build the initial argument ontology. It was later extended by the proposed process and resulted in the ontology in Fig. 1. The initial state and the goal were manually defined and a planner (Yordanova et al., 2012) was used to identify any rules that contradict each other.

The resulting model $CSSM_l$ was compared to a hand-crafted model developed for the same problem $CSSM_m$. The reason for that was to evaluate the model complexity in comparison to that of a model built by a human expert.

Later, the models were used to recognise seven plans, based on the video log from the cooking dataset. Landmarks were used as action selection heuristic (Richter and Westphal, 2010). This allowed the computation of the approximate goal distance to the goal. The actions in the plans were represented according to the action schema learned in each of the models. The plans were between 67 and 86 steps long. Cohen's kappa was calculated to determine the plans' similarity. The mean kappa was 0.18, which indicates that the overlapping of the plans was low.

5 RESULTS

Table 1 provides information about the model dimensions of both $CSSM_l$ and $CSSM_m$. The model designer in $CSSM_m$ identified 16 action classes. The learning method in $CSSM_l$ discovered 15 action classes. That is due to the fact that the action *wait* was introduces in $CSSM_m$. As this action was not present in the textual instructions, it was not discovered in $CSSM_l$. The action *wait*, however, is causally unrelated to any of the other actions so its presence did not change the causal structure of the model.

Furthermore, $CSSM_l$ discovered 18 arguments in the textual instructions, while the designer modelled 17 in $CSSM_m$. This is due to the fact that in the textual instructions after the carrots are cooked, they are transformed into a soup. Thus, *soup* is also an argument in the model. On the other hand, the system designer decided to use the argument *carrots* also for describing the soup. This shows that the approach is able to learn also context information that is discarded in the manual model.

$CSSM_l$ learned less operator (action) templates and predicates than those modelled in $CSSM_m$. On the other hand, the rules resulted in smaller number of ground operators and predicates in $CSSM_m$ than in $CSSM_l$. This indicates that $CSSM_l$ is more general than $CSSM_m$. This can be explained by the fact that there are less restrictions in the form of predicates in $CSSM_l$ than in $CSSM_m$.

Table 1: Model parameters of $CSSM_m$ and $CSSM_l$.

parameters	$CSSM_l$	$CSSM_m$
action classes	15	16
operator templates	22	28
predicate templates	17	36
ground operators	189	110
ground predicates	204	160
arguments	18	17
max. arg. per template	2	3
ontology elements	66	41
ontology levels	7	5
states	$46,845,389$	$10,312$

The model designer implemented action templates with maximum of three arguments in $CSSM_m$. On the other hand, in $CSSM_l$ templates with maximum of two arguments were learned. This indicates that the learned templates in $CSSM_l$ are of lower complexity than those in $CSSM_m$. Furthermore, the model designer in $CSSM_m$ introduced several additional predicates aiming at reducing the model size and increasing the action probability. This made the preconditions and effects in $CSSM_m$ more complex than those in $CSSM_l$[2]. Moreover, the proposed approach was

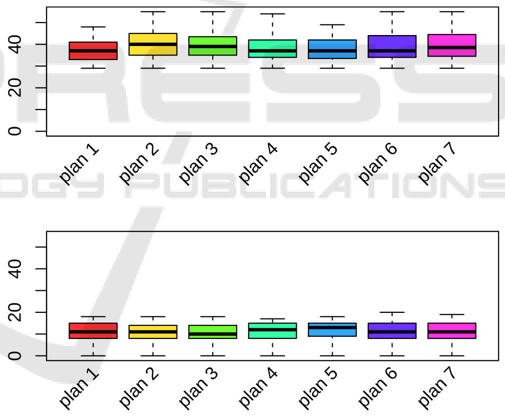

Figure 3: Median number of possible actions in $CSSM_l$ given the plan (top), and in $CSSM_m$ (bottom).

able to learn an argument ontology with seven levels of abstraction and with 66 elements in total (See Fig. 1). The model designer in $CSSM_m$ modelled a simpler ontology with five levels of abstraction and with 41 elements in total. This indicates that the learning approach is able to discover more complex semantic structures, given the same problem domain. Furthermore, an iteratively deepening depth first search with maximum depth of seven was applied to analyse the

[2]For example, the action *wash* in $CSSM_l$ consists of three rules in the form of predicates in the precondition and one rule in the effect clause. On the other hand, the same action in $CSSM_m$ consists of seven rules in the precondition and three rules in the effect.

state space graph. It discovered over 46 million states in $CSSM_l$ and 10,312 states in $CSSM_m$. Due to the size of the models, the complete state space graphs were not traversed. The larger number of discovered states in $CSSM_l$ once again stands to show that the learned model is more general than the manually developed one. The number of plans in both models was not computed because the whole state space was not completely explored.

$CSSM_l$ and $CSSM_m$ were also applied to seven plans produced by observing the video log of the kitchen experiment and by using the action schema from the corresponding model. Fig. 3 shows the median branching factor in each plan. There the number of possible actions from a given state in $CSSM_l$ (Fig. 3 (top)) was relatively high (between 30 and 55 executable actions). This is explained by the model generality. In other words, $CSSM_l$ does not have many restrictions which results in high behaviour variability. It can also be seen that the first plan had a slightly smaller branching factor than the rest of the plans. This is due to the fact that the instructions describing the experiment were compiled based on the execution sequence in the first experiment. This means that the learned model was overfitted for the first plan. Still, it was able to successfully interpret the remaining plans. In comparison, $CSSM_m$ had a median branching factor of 10 (Fig. 3 (bottom)). This is due to the additional modelling mechanisms applied by the designer to reduce the model complexity (Yordanova et al., 2014).

Regardless of the lower branching factor, having ten choices from a given state reduced the action probability. This was reflected in the probability of executing an action in the plan given the model. Fig. 4 (top) shows that the probability was very low for both models. Surprisingly, with nearing the goal state, the probability in $CSSM_l$ increased (Fig. 4 (top left)), while the probability in $CSSM_m$ stayed low until the goal state was reached (Fig. 4 (top right)). This can be explained by the shorter goal distance in $CSSM_l$ (Fig. 4 (bottom left)) compared to that in $CSSM_m$ (Fig. 4 (bottom right)). The distance to the goal in $CSSM_l$ starts with 11 states to the goal and generally decreases with each executed action. On the other hand, the goal distance in $CSSM_m$ is 50 at the beginning of the problem and it stays relatively long during the execution of the plan. As the estimated goal distance is used as an action selection heuristic, the long distance decreased the action probability in $CSSM_m$.

6 DISCUSSION

In this paper we presented an approach to learning models of human behaviour from textual instructions. The results showed that the approach is able to learn a causally correct model of human behaviour. The model was able to explain the behaviour of seven experiment participants that executed kitchen tasks.

In difference to existing approaches, the proposed method was able to learn a complex domain ontology. It was then used for generalising the model. This was reflected in the large state space and branching factor of the resulting model.

It also applied a new method for causal relation discovery, that was previously not applied to model learning problems. The method yielded good results without the need of a learning phase.

The model was compared to a hand-crafted CSSM model. The results showed that the learned model is more general than the hand-crafted model. On the other hand, they also showed that the learned model has smaller estimated goal distance. This resulted in the higher action probability when executing a plan, compared to the hand-crafted model.

Some of the limitations, the learned model had, are as follows. The model was able to learn actions with simple predicates. This resulted in the model generalisation. However, if applied to activity recognition, general models tend to decrease the model performance due to the high branching factor. This can be solved through reliable action selection heuristics, or through strategies for reducing the model complexity trough more complex predicates (Yordanova et al., 2014). In the future, we intend to include mechanisms for utilising these strategies during the learning process.

The model was unable to learn repeating actions, such as repetitively eating or drinking. This is due to the fact that the model learned that the precondition for executing the action is that it still has not been executed[3]. Then to repeat the action, the precondition that the action was not executed is violated. To solve this problem, a mechanism has to be introduced for identifying repeating actions in the text.

Another aspect of model learning is how to learn the initial and goal states. In this work we defined them manually. In the future we intend to investigate methods for learning the initial and goal state based on possible predicate combinations and reinforcement learning techniques.

Furthermore, in this work we only addressed the problem of learning the model structure through its domain ontology and the preconditions and effects

[3]This is not a problem for action pairs like *open* and *close* as they negate each other's effects. However, for actions that do not have negating action, the *executed* predicate cannot be negated, rendering the action impossible.

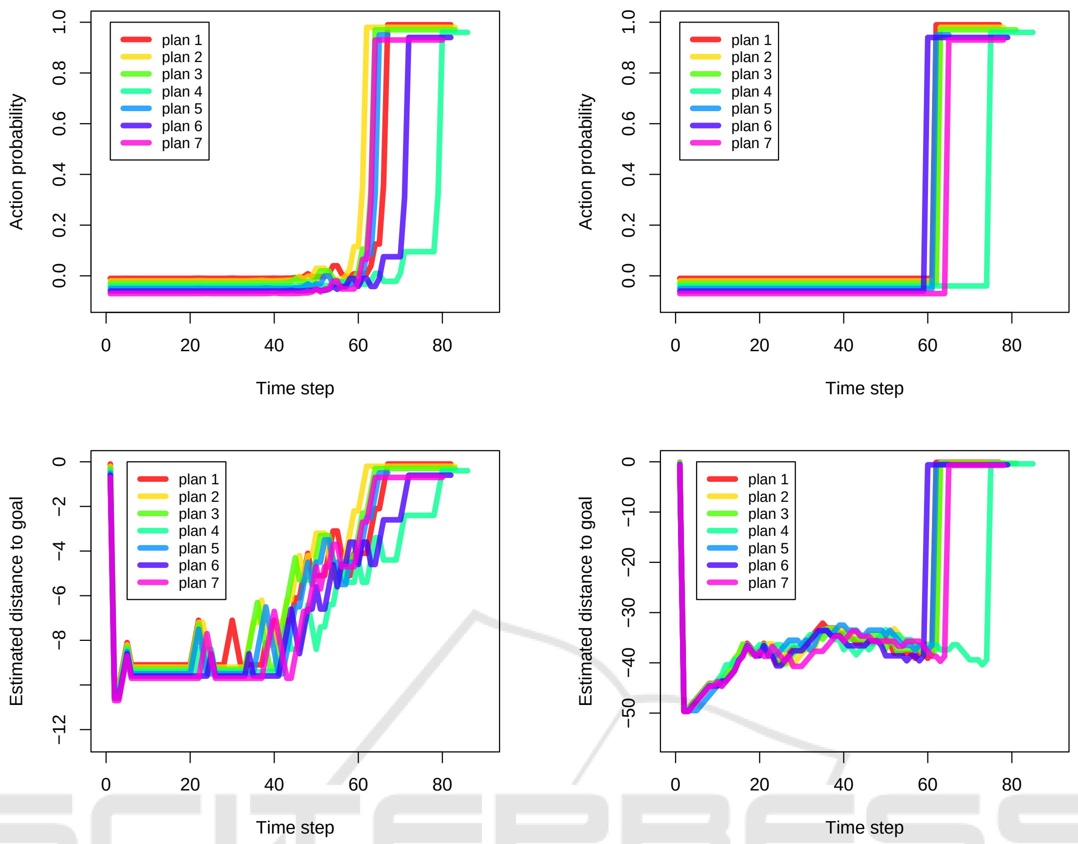

Figure 4: Probability of selecting the action in the plan, given $CSSM_l$ (top left) and given $CSSM_m$ (top right), and estimated distance to goal, given the action executed in the plan for $CSSM_l$ (bottom left) and for $CSSM_m$ (bottom right). To improve the visibility of overlapping lines, each line was shifted with 1% from the preceding.

describing the actions. However, if we want to apply the models to activity recognition tasks, the model needs to be optimised to increase the probability of selecting the correct action. This can be done by employing reinforcement learning techniques based on observations similar to those proposed in (Branavan et al., 2012).

In the future, we intend to extend our approach to learning models for AR problems based on sensor observations. To achieve that, methods for optimising the model structure will be investigated.

REFERENCES

Branavan, S. R. K., Kushman, N., Lei, T., and Barzilay, R. (2012). Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 126–135, Stroudsburg, PA, USA. Association for Computational Linguistics.

Branavan, S. R. K., Zettlemoyer, L. S., and Barzilay, R. (2010). Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1268–1277, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865.

Chen, L., Hoey, J., Nugent, C., Cook, D., and Yu, Z. (2012). Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(6):790–808.

Granger, C. W. J. (1969). Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica*, 37(3):424–438.

Hiatt, L. M., Harrison, A. M., and Trafton, J. G. (2011). Accommodating human variability in human-robot teams through theory of mind. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, IJCAI'11, pages 2066–2071, Barcelona, Spain. AAAI Press.

Hoey, J., Poupart, P., Bertoldi, A. v., Craig, T., Boutilier, C., and Mihailidis, A. (2010). Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. *Com-*

puter Vision and Image Understanding, 114(5):503–519.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2014). Grounding verbs of motion in natural language commands to robots. In Khatib, O., Kumar, V., and Sukhatme, G., editors, *Experimental Robotics*, volume 79 of *Springer Tracts in Advanced Robotics*, pages 31–47. Springer Berlin Heidelberg.

Krüger, F., Hein, A., Yordanova, K., and Kirste, T. (2015). Recognising the actions during cooking task (cooking task dataset). University Library, University of Rostock. http://purl.uni-rostock.de/rosdok/id00000116.

Krüger, F., Nyolt, M., Yordanova, K., Hein, A., and Kirste, T. (2014). Computational state space models for activity and intention recognition. a feasibility study. *PLoS ONE*, 9(11):e109381.

Krüger, F., Yordanova, K., Hein, A., and Kirste, T. (2013). Plan synthesis for probabilistic activity recognition. In Filipe, J. and Fred, A. L. N., editors, *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART 2013)*, pages 283–288, Barcelona, Spain. SciTePress.

Krüger, F., Yordanova, K., Köppen, V., and Kirste, T. (2012). Towards tool support for computational causal behavior models for activity recognition. In *Proceedings of the 1st Workshop: "Situation-Aware Assistant Systems Engineering: Requirements, Methods, and Challenges" (SeASE 2012) held at Informatik 2012*, pages 561–572, Braunschweig, Germany.

Li, X., Mao, W., Zeng, D., and Wang, F.-Y. (2010). Automatic construction of domain theory for attack planning. In *IEEE International Conference on Intelligence and Security Informatics (ISI), 2010*, pages 65–70.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Nguyen, T. A., Kambhampati, S., and Do, M. (2013). Synthesizing robust plans under incomplete domain models. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 26*, pages 2472–2480. Curran Associates, Inc.

Okeyo, G., Chen, L., Wang, H., and Sterritt, R. (2011). Ontology-based learning framework for activity assistance in an adaptive smart home. In Chen, L., Nugent, C. D., Biswas, J., and Hoey, J., editors, *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 237–263. Atlantis Press.

Perkowitz, M., Philipose, M., Fishkin, K., and Patterson, D. J. (2004). Mining models of human activities from the web. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 573–582, New York, NY, USA. ACM.

Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H., and Hahnel, D. (2004). Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57.

Ramirez, M. and Geffner, H. (2011). Goal recognition over pomdps: Inferring the intention of a pomdp agent. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, volume 3 of *IJCAI'11*, pages 2009–2014, Barcelona, Spain. AAAI Press.

Richter, S. and Westphal, M. (2010). The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177.

Sil, A. and Yates, E. (2011). Extracting strips representations of actions and events. In *Recent Advances in Natural Language Processing*, pages 1–8.

Tenorth, M., Nyga, D., and Beetz, M. (2010). Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1486–1491.

Ye, J., Stevenson, G., and Dobson, S. (2014). Usmart: An unsupervised semantic mining activity recognition technique. *ACM Trans. Interact. Intell. Syst.*, 4(4):16:1–16:27.

Yordanova, K. (2015a). Discovering causal relations in textual instructions. In *Recent Advances in Natural Language Processing*, pages 714–720, Hissar, Bulgaria.

Yordanova, K. (2015b). Time series from textual instructions for causal relations discovery (causal relations dataset). University Library, University of Rostock. http://purl.uni-rostock.de/rosdok/id00000117.

Yordanova, K. and Kirste, T. (2015). A process for systematic development of symbolic models for activity recognition. *ACM Transactions on Interactive Intelligent Systems*, 5(4).

Yordanova, K., Krüger, F., and Kirste, T. (2012). Tool support for activity recognition with computational causal behaviour models. In *Proceedings of the 35th German Conference on Artificial Intelligence*, pages 561–573, Saarbrücken, Germany.

Yordanova, K., Nyolt, M., and Kirste, T. (2014). Strategies for reducing the complexity of symbolic models for activity recognition. In Agre, G., Hitzler, P., Krisnadhi, A., and Kuznetsov, S., editors, *Artificial Intelligence: Methodology, Systems, and Applications*, volume 8722 of *Lecture Notes in Computer Science*, pages 295–300. Springer International Publishing.

Zhang, Z., Webster, P., Uren, V., Varga, A., and Ciravegna, F. (2012). Automatically extracting procedural knowledge from instructional texts using natural language processing. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association.

Zhuo, H. H. and Kambhampati, S. (2013). Action-model acquisition from noisy plan traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2444–2450, Beijing, China. AAAI.