Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web

Daniel Nyga, Moritz Tenorth, Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{nyga, tenorth, beetz}@in.tum.de

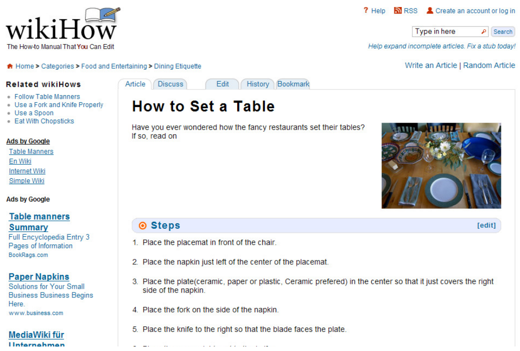
***Abstract*— Service robots will have to accomplish more and more complex, open-ended tasks and regularly acquire new skills. In this work, we propose to use the World Wide Web as a source of knowledge and to benefit from the large number of task descriptions on web sites like ehow.com. We present a system that converts instructions for complex household chores given in natural language into a formal, logic-based representation, resolves the word senses using the WordNet database and the Cyc ontology and exports the generated plans into the mobile robot's plan language RPL. Usually, these instructions lack important details as they are intended to be understood by humans who automatically add this information. We present approaches for adding different kinds of knowledge that is required for the successful plan execution. The system has been successfully tested with a set of about 140 natural language directives, of which up to 80% could be correctly transformed.**

I. Introduction

Consider autonomous personal robots that are to perform housework ([1], [2], [3]). Such a robot has to set the table. It has to cook spaghetti. In cases where children are visiting it should cook the spaghetti on the plates in the back to make the kitchen childsafe.

One of the key challenges for autonomous personal robots that are to perform everyday manipulation tasks in households is the openendedness of the task domain. It is an open challenge to generate the range of plans that contain such rich specifications of how actions are to be executed, what events to wait for before executing the next actions, which additional behavior constraints to satisfy, and which problems to watch for. The expressiveness of the necessary task instructions goes well beyond the expressiveness of planning problem descriptions used in the AI action planning area [4]. Even if the planning problems could be expressed the number of objects and actions including their possible parameterizations would cause search spaces that are not tractable by the search algorithms of these planning systems.

Luckily, plans for all the tasks listed in the introductory paragraph are already available. Webpages such as ehow.com and wikihow.com provide step-by-step instructions for setting the table (Figure I), cooking spaghettis, and making a kitchen childsafe. Both web sites contain thousands of directives for everyday activities: about 45,000 on wikihow.com and even more than 250,000 articles on `ehow.com`. The most interesting categories in our case, "Food & Drink" and "House & Garden", feature more than 38,000 and 36,000 descriptions respectively.


wikiHow
The How-to Manual That You Can Edit
? Help
RSS
Create an account or log in
Type in here
Search
Help expand incomplete articles. Fix a stub today!
Home > Categories > Food and Entertaining > Dining Etiquette
Write an Article | Random Article
Related wikiHows
Follow Table Manners
Use a Fork and Knife Properly
Use a Spoon
Eat With Chopsticks
Ads by Google
Table Manners
En Wiki
Internet Wiki
Simple Wiki
Ads by Google
Table manners Summary
Full Encyclopedia Entry 3 Pages of Information
BookRags.com
Paper Napkins
Solutions for Your Small Business Business Begins Here.
www.business.com
MediaWiki für Unternehmen
Article
Discuss
Edit
History
Bookmark
Steps
[edit]

How to Set a Table

Have you ever wondered how the fancy restaurants set their tables? If so, read on

1. Place the placemat in front of the chair.
2. Place the napkin just left of the center of the placemat.
3. Place the plate(ceramic, paper or plastic, Ceramic prefered) in the center so that it just covers the right side of the napkin.
4. Place the fork on the side of the napkin.
5. Place the knife to the right so that the blade faces the plate.

Fig. 1. Example task description from `wikihow.com`.

Thus, a promising alternative to generating plans from first principles as it is the standard approach in AI planning, which is still far away from generating plans with these types of complexities [5], is to look up the instructions for a new task from ehow.com and wikihow.com and translate the natural language instructions into executable robot plans. With such an approach working, the personal robots would already know how to perform thousands of household tasks that cover the whole range of everyday activity.

The execution of web instructions can be performed as a three stage process:

1) *Translation of the natural language instructions into an almost working but buggy robot plan.* Due to the fact that web instructions are written to be executed by people with commonsense knowledge, the instructions may contain ambiguities, missing parameter information and even missing plan steps.
2) *Debugging of the plan.* In a second step the above plan flaws are to be detected, diagnosed, and forestalled using transformational planning based on mental simulations of the plans in a simulated environment [6]
3) *Plan optimization.* Web instructions also fail to specify how tasks can be carried out efficiently. Thus, transformational planning is applied to find out that the table setting task can be carried out more efficiently if the robot stacks the plates before carrying them, if it carries cups in each hand, and if it leaves the cupboard doors open while setting the table [7].

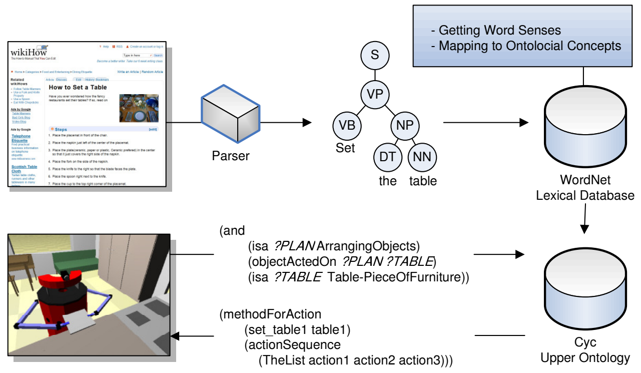
In this paper we design, implement, and empirically evaluate a system that performs the first computational task: the translation of the natural language instructions into an almost working but buggy robot plan.

Understanding these natural-language task descriptions is a hard challenge and part of the general natural-language understanding problem. The system has to deal with ambiguities on different levels and resolve the meaning of actions, objects and properties by mapping the words to ontological concepts. This task only becomes tractable as we do not target at the understanding of textual descriptions in general, but limit ourselves to "howtos" that mainly consist of a sequence of imperative statements with a rather similar structure.

The remainder of the paper is organized as follows: We start with an overview of of the system components (Section II), the parser (III-A), instruction recognition (III-B), resolution of word senses (III-C), the internal plan representation (III-D), some methods for adding information that is required for the plan execution but not contained in the howto (**??**), and finally the export into the RPL language (III-E). Afterwards, we describe our experiments and the evaluation results (IV). We finish with an overview of related work (V), a general discussion of the problem and our approach (VI) and our conclusions.

II. Overview

Figure 2 shows the overall structure of our system. We use a Probabilistic Context-Free Grammar (PCFG) parser for examining the sentence structure and resolve the meaning of words using the lexical database WordNet [8] and the Cyc ontology [9]. The robot's knowledge base is derived from the Cyc ontology and all things in the environment are identified by the Cyc class they belong to. In order to refer to the same things as the rest of the control program, this import system has to resolve the ontological concepts corresponding to the words.



```
(and
    (isa ?PLAN ArrangingObjects)
    (objectActedOn ?PLAN ?TABLE)
    (isa ?TABLE Table-PieceOfFurniture))

(methodForAction
    (set_table1 table1)
    (actionSequence
      (TheList action1 action2 action3)))
```

Fig. 2. General strategy for importing task descriptions from the WWW.

Based on the syntax tree that the parser creates, we create data structures describing objects, actions, pre- and postconditions, temporal constraints, quantifiers and prepositional relations. These data structures are combined to a first internal instruction representation in a bottom-up fashion. This representation already describes the structure of an instruction, but still uses the words from the original howto.

To resolve the meaning of the words, we use the mapping of a set of synonyms in WordNet (so-called "synsets") to a concept in the Cyc ontology. The relation of words and ontological concepts is not a one-to-one mapping as each word can have different meanings and several synonymous words can belong to the came concept.

When the meaning of words is resolved, the instructions are represented in Cyc in a way compatible to the OpenCyc planner, a logic-based plan representation which supports reasoning on the plans. This representation serves as an interlingua and is useful for examining the plans and detecting and inferring missing information. In order to execute the plans on our robots, we export them into the Reactive Plan Language (RPL) which is the basis of our planning system.

III. Understanding Instructions

In this section, we will present the different steps from the instruction in natural language to the executable plan with the example sentence "Place the cup on the table".

A. Natural Language Parsing

The understanding of an instruction starts with the syntactic analysis of the sentence. This task comprises the part-of-speech (POS) tagging and the creation of the parse tree with phrases as the internal nodes and the POS tags as the leaves. An example of such a parse tree is shown in Figure 3 on the left side.

The POS tags in this example denote if a word is a verb in the base form (VB), a determiner (DT), a singular noun (NN) or a preposition (IN), and the phrase tags distinguish between noun phrases (NP), verb phrases (VP) and preposition phrases (PP). Quantifying phrases denoted by QP are missing in this example. Our implementation is based on the Stanford Parser [10] that automatically creates parse trees using a Probabilistic Context-Free Grammar.

B. Recognition of Instructions

The parse tree is the input for the instruction recognition algorithm which combines the parts of the instruction in a bottom-up fashion. The general idea is that each phrase node accumulates the data coming from its children to a new data structure. That way, complex descriptions are combined into one structured representation.

- Leaves of the parse tree are assigned an instance of the *word* data structure $w = (l, p, M)$ consisting of the label, POS and WordNet synset ID described in the next section.
- Quantifying phrases typically comprise a set or range of cardinal numbers Q and are combined with the measuring unit M to a *quantifier* $q = (Q, M)$.
- Noun phrases consist of a determiner and a list of nouns (one object), a noun phrase and a quantifying phrase (certain amount of something) or other noun phrases (a list of objects) and transform these children into a new *object* data structure. An *object* $o = (w, A, P, q)$ consists of a designator w and possibly a set of adjectives A, prepositional expressions P describing its location and a quantifier q specifying the amount of it to be used.

- Prepositional phrases consist of a preposition P and a noun phrase specifying an object O and create a *preposition* instance $p = (P, O)$.
- Verb phrases describe instructions and are transformed into an *instruction* instance $i = (a, O, P, t)$ with a being the action verb, O a set of objects, P a set of prepositional postconditions and t describing the duration of the action.

Figure 3 exemplarily shows how the parse tree is translated into two *object* instances, once *preposition* and one *instruction*.

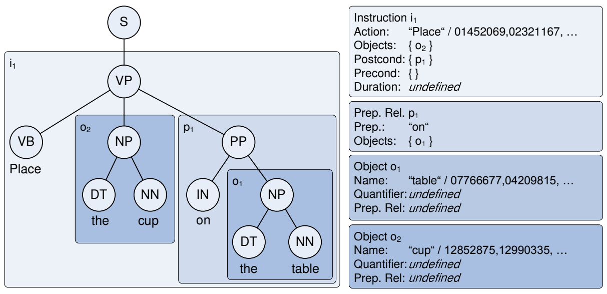


Fig. 3. Parse tree for the sentence "Place the cup on the table" (left) and the resulting data structures representing the instruction that are created as an intermediate representation by our algorithm (right).

Some automatic post-processing of the generated data structures is necessary to resolve issues that require more contextual information. Object names consisting of multiple words (like "stove top") are resolved as well as phrasal verbs, i.e. verbs that occur in combination with a preposition (for example "turn on").

These constructs are normally not recognized correctly by the parser but result in characteristic structures which can be corrected automatically. Quantifiers are finally resolved in this post-processing step by linking the cardinal numbers from the quantifying phrase to the corresponding measuring unit.

Finally, we add a simple method for resolving pronomial anaphora, references using pronouns as "it" or "them". In fact, other pronouns are hardly used in howtos which mainly deal with objects and not with persons. Our observation is that most of the instructions refer to the last mentioned object, so we resolve the anaphora using this heuristic.

Since we focus on understanding imperative clauses, we currently cannot recognize instructions which are hidden in declarative sentences ("The cup is to be placed on the table").

C. Retrieving and Disambiguating Word Senses

Once the structure of instructions is identified, the system resolves the meaning of the words using the WordNet lexical database and the Cyc ontology.

In WordNet, each word can have multiple senses, i.e. it is contained in multiple "synsets" which group synonymous words. When querying for a word, the WordNet database returns a set of IDs of synsets this word is part of, grouped into nouns, verbs and adjectives. The part-of-speech tag from the syntax parsing helps with choosing a category. "Place" for example can be a verb as well as a noun depending on the context.

The synset IDs are mapped to ontological concepts in Cyc via the *synonymousExternalConcept* predicate. "Cup" as a noun, for instance, is part of the synsets *N03033513* and *N12852875* which are mapped to the concepts *DrinkingMug* and *Cup-UnitOfVolume* respectively.

```
(synonymousExternalConcept
        DrinkingMug WordNet-Version2_0 "N03033513")
(synonymousExternalConcept
        Cup-UnitOfVolume WordNet-Version2_0 "N12852875")
```

Most queries return several synsets for each word, so a word sense disambiguation method has to select one. We propose a simple algorithm that infers the most probable action based on the prepositions in the instruction and the most probable object senses based on the action.

This algorithm is based on the observation that the word sense of the action verb is strongly related to the prepositions (e.g. "taking something from" as *TakingSomething* in Cyc up vs. "taking something to" as *PuttingSomethingSomewhere*). We assume that the prepositions occur independently which is not completely true but proved to be a reasonable approximation.

Let A_i be the ontological concept, i.e. one particular meaning of the action verb a_i in the instruction $i = (a_i, O_i, P_i, t_i)$, and let $P := \{on, in, to, from, of, next\_to, with, without\}$ be a set of binary random variables indicating if a preposition is present in the current instruction.

Then the conditional probability of the action A_i given the set of prepositions $P_i \subseteq P$ in i is:

$$P(A_i = t | \bigwedge_{p \in P_i} p = t) = \frac{P(A_i = t, \bigwedge_{p \in P_i} p = t)}{P(\bigwedge_{p \in P_i} p = t)} \quad (1)$$

$$= \frac{P(A_i = t, \bigwedge_{p \in P_i} p = t)}{\prod_{p \in P_i} P(p = t)} \quad (2)$$

Having the most probable action A_i, the most likely object word sense O can be calculated from the word O_i with

$$P(O = t | A_i = t) = \frac{P(O = t \land A_i = t)}{P(A_i = t)} \quad (3)$$

For calculating the above probabilities, it is necessary to know the frequency of the prepositions and actions and the joint probabilities of actions and prepositions as well as objects and actions. These probabilities were trained by supervised learning on the training set.

If there is no statistical evidence about any sense of a word, the algorithm chooses the meaning with the highest frequency rank in WordNet.

D. Plan Representation in the Knowledge Base

The imported plans are stored in the knowledge base until they are executed. This representation allows for reasoning on the sequence of actions, the objects and other parameters.

A plan comprises a list of actions which have to be executed in order to perform the task:

```
(methodForAction
  (COMPLEX_TASK ARG1 ARG2 ...)
  (actionSequence
      (TheList action1 action2 ...)))
```

Each step *action1*, *action2* etc. is an instance of an action concept in the ontology. Since the system knows which parameters are required for the respective concept, it can detect if the specification is complete. For instance, the action *PuttingSomethingSomewhere* needs to have information about the object to be manipulated and the location where this object is to be placed.

Action parameters are created as instances of objects or spatial concepts and are linked to the action with special predicates. In the example below, the *objectActedOn* relation specifies which object the action *place1* of type *PuttingSomethingSomewhere* is to be executed on. *purposeOf-Generic* is used to describe post-conditions; in this case, the outcome of the action *place1* shall be that the object *cup1* is related to *table1* by the *on-UnderspecifiedSurface* relation.

```
(isa place1 PuttingSomethingSomewhere)
(isa table1 Table-PieceOfFurniture)
(isa cup1 DrinkingMug)
(objectActedOn place1 cup1)
(purposeOf-Generic
      place1
      (on-UnderpecifiedSurface
            cup1
            table1))
```

Time constraints are translated into *timeSpan* relations, quantifiers are modelled with the *amountOfObject* property we introduced, for example

```
(amountOfObject tablesalt1 (Teaspoon-UnitOfVolume 1 2))
(timeSpan boilingFood1 (MinutesDuration 10 12))
```

E. Export to RPL Code

The imported plans are executed by our B21 robot acting in a kitchen environment. This scenario exists both in reality and in a realistic physical simulation (Figure III-E). A simulation allows the easy execution and evaluation of the plans and provides, since we focus on plan descriptions on a rather high level of abstraction, a sufficient level of detail for testing.

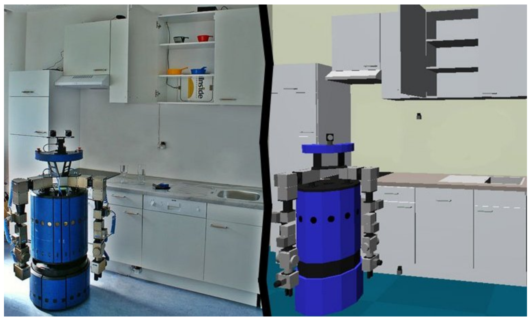


Fig. 4. B21 robot in our real and simulated kitchen environment.

The plans for our robot are implemented in extended RPL (Reactive Plan Language) [11] which provides a clear, expressive and extensible language for writing robot plans. RPL is an interpreted language, written in Lisp. For execution, the plans have to be transformed from the Cyc representation into a valid RPL plan.

In our planning system, objects and locations are described by designators, qualitative and possibly incomplete specifications that are resolved during the plan execution. Designators contain all information that is needed for selecting an object, for example a cup in the state "clean". Which object fulfills the requirements and which one is finally selected is not decided until the plan gets executed.

Single instructions, which are modeled as action instances in Cyc, are translated into achieve statements which get a list with the goal to be achieved as the first entry. Depending on the goal, additional parameters can be specified. For each goal, there exists a high-level plan that can be used to achieve it.

```
(pl:define-high-level-plan
:input-schema
      (:achieve (place1))
:output-plan (
      (with-designators (
            (drinkingmug1 '(an entity
                  (type cup)))
            (table1 '(an entity
                  (type table)))
            (location1 (a location
                  (on table1)))
)

(:achieve (entity-at-place
      drinkingmug1
      location1 ))
)))
```

IV. Evaluation and Discussion

The system performance depends to a high degree on the correctness and completeness of different, partly external software components. In order to give a more detailed evaluation, we did not only examine the complete system but also looked at individual modules.

As the syntax parser is one major source of error, we evaluate the system both with automatically parsed syntax trees and manually created ones. Another main issue affecting the recognition rate is missing mappings from synsets in WordNet to the corresponding ontological concept in Cyc. In the training set, we manually added 72 mappings for actions, objects and adjectives. Finally, we analyze how many instructions are correctly transformed into the internal data structures before being added to the knowledge base to show the dependency on lacking ontological concepts.

We tested the implemented system on approximately 100 instructions from a training set and another 40 from a test set of howtos which are taken from `ehow.com` and `wikihow.com`.

First, we trained the disambiguator on the training set with manually created parse trees. Afterwards, we ran the system including the syntax parser on the same set of howtos, the results are shown in Table I. With correctly parsed trees, the system achieves a recognition rate of about 80% before the ontology mapping and the transformation of the instructions into a logic representation.

	aut. parsed		man. parsed	
Training Set:				
Actual Instructions	**95**	**100%**	**95**	**100%**
Correctly Recognized	66	69%	76	80%
False Negative	29	31%	19	20%
False Positive	4	4%	2	2%
Test Set:				
Actual Instructions	**42**	**100%**	**42**	**100%**
Correctly Recognized	27	64%	37	88%
False Negative	15	36%	5	12%
False Positive	3	7%	4	9%

TABLE I

SUMMARY OF THE EVALUATION ON INSTRUCTION LEVEL

The remaining 20% have either been recognized incorrectly (missing object or preposition in the instruction) or not at all. The latter group also comprises instructions that are not expressed as imperative statements and are therefore not supported by the current system. False positive cases are phrases which have been erroneously recognized as instructions. A similar recognition rate could be achieved on the test set. In both test runs, a significant decrease in the recognition rate is caused by errors in the automatically generated parse trees (11 percentage points in the training set, 24 in the test set).

Table II shows the results of the import into the knowledge base. 74 of the 76 instructions which have been recognized in the previous step could successfully be imported, the two errors were caused by a mapping of a word sense to a concept that cannot be instantiated as an object in Cyc.

	aut. parsed		man. parsed	
Training Set:				
Actual Instructions	**95**	**100%**	**95**	**100%**
Import Failures	**31**	**33%**	**21**	**22%**
Incorrectly/Not recognized	29	94%	19	90%
Missing WordNet entries	0		0	
caused Import Failures	0	0%	0	0%
Missing Cyc Mappings	0			
caused Import Failures	0	0%	0	0%
Misc. Import Errors	2	6%	2	10%
Disambiguation Errors	0		0	
Correctly imported into KB	**64**	**67%**	**74**	**78%**
Test Set:				
Actual Instructions	**42**	**100%**	**42**	**100%**
Import Failures	**25**	**60%**	**24**	**57%**
Incorrectly/not recognized	15	60%	5	21%
Missing WordNet entries	3		3	
caused Import Failures	2	8%	2	8%
Missing Cyc Mappings	14		20	
caused Import Failures	8	32%	17	71%
Misc. Import Errors	0	0%	0	0%
Disambiguation Errors	2		2	
Correctly imported into KB	**17**	**40%**	**18**	**42%**

TABLE II

SUMMARY OF THE EVALUATION ON KNOWLEDGE BASE LEVEL

Results of the import of the test set into the knowledge base show that most of the errors are caused by lacking WordNet entries or mappings to Cyc concepts: Of the 27 correctly recognized instructions in the automatically parsed case, only 17 could be imported because of 2 failures caused by nonexistent WordNet synsets and 8 failures due to missing Cyc mappings.

V. Related Work

To the best of our knowledge our work is the first to mine complex household chores from the web and translate them into executable robot plans. However, many efforts have been made in building natural language interfaces to computer systems in the past.

Perkowitz et al. [12] proposed to mine web sites like `ehow.com` for activity models, but not with the intention of executing them. Instead, they extracted a set of objects involved in each step of a task and learned Hidden Markov Models in order to match sequences of RFID tag readings and visual object detections against the task descriptions. An important difference to our system is that we aim at understanding the whole instruction in a way that it can be executed instead of just looking for potentially relevant objects.

Various approaches exist for building speech interfaces to robots, but normally, they are limited in terms of vocabulary or allowed grammatical structures ([13], [14], [15]). As we cannot influence the creation of the howtos, this simplification is not possible in our case.

Related work on web information extraction usually targets at the recognition of single pieces of information. Wu and Weld [16] generate info boxes in Wikipedia containing information like the area or population of a state using a CRF classifier from the dictionary entries. Alani et al. [16] extract more information and create biographies from web documents with an approach similar to ours. They are also using a syntax parser, WordNet and ontologies to obtain the desired information. Mueller [17] presents a system for understanding texts involving dining in a restaurant and applies scripts [18] to infer missing information.

VI. Discussion

The system we present shows good performance on both the training set and an unknown test set of howtos. Problems that remain are mainly caused by the external software components we use: The syntax parser is not perfect, especially when the sentences become longer. Missing mappings from WordNet to Cyc are the second main source of error. Currently, only 11,000 of the 300,000 Cyc concepts are mapped to a WordNet synset. Anyway, we assume that the number of the number of mappings that have to be added will decrease over time as more and more of the relevant household items are covered.

At the moment, our system only understands single imperative instructions. The understanding of all possible ways an instruction can be formulated is beyond the scope of this project, but we plan to add support for directives starting with modal verbs (“You should place the cup on the table.”) and for understanding relations between instructions (and, or, if-then etc.).

As we already mentioned, most howtos lack important information that prevents them from being executed by a robot. We are currently extending our planning system so that it can detect if information which is required for an action is not sufficiently specified and can then trigger a plan debugger. One part of the debugging is about resolving references like "add salt" [to the pasta] or "pour the contents into the colander" that depend on the current execution context. Another part is to infer different types of missing knowledge from external sources.

One type is common-sense knowledge like the fact that only clean cups and dishes are used when setting a table. These facts can probably be inferred most efficiently using scripts for household activities. Orignally proposed by Schank [18], they describe which actions in which order are involved into an activity. For resolving commands like "use the eating bowl" or "select a pot", we are currently investigating if knowledge from the Open Mind Indoor Common Sense project [19] can be used.

Spatial properties like the distance between objects or their orientation are also frequently missing. Interestingly, many of these spatial properties are determined by external factors (like a knife being placed in a certain orientation inside a drawer because of the cutlery tray), but these factors do not have to be fully understood: Often, it is sufficient to remember the orientation of an object depending on its x,y position in the environment.

With this simple approach, we successfully learned several different concepts: The orientation of silverware on the table, a common pairwise distance between for instance a plate and a knife or the fact that items in a certain region are most probably put down on top of the table. For that statement, we combined the learned z-coordinate with a heuristic that objects which are located just above a horizontal surface are probably on top of it. The locations of pieces of furniture are provided by our 3D environment map [20].

When manually updating the imported plans with these learned concepts, some of them already become executable. The accompanying video shows the robot setting a table as specified in the howto which can be found at `http://www.wikihow.com/Set-a-Table`.

VII. Conclusions

The most important conclusion is that the automatic translation of natural-language instructions into executable robot plans is certainly feasible. This is important because websites like `ehow.com` cover huge ranges of everyday manipulation tasks at a level of detail and interaction that supersede those plans that can be generated by AI plan methods.

We present a system that creates valid robot plans for high-level household activities based on natural-language task descriptions from the World Wide Web. The howto-style descriptions from web sites like `ehow.com` and `wikihow.com` are transformed into a formal, logic-based representation using a natural language parser, the lexical database WordNet and the Cyc ontology. This formal representation can be exported as an executable robot plan.

Our evaluation shows that the system is able to recognize up to 80% of all instructions in an unknown set of instructions. Import failures are mainly caused by errorneous parsing results or missing ontological mappings.

References

[1] M. Beetz, F. Stulp, B. Radig, J. Bandouch, N. Blodow, M. Dolha, A. Fedrizzi, D. Jain, U. Klank, I. Kresse, A. Maldonado, Z. Marton, L. Mösenlechner, F. Ruiz, R. B. Rusu, and M. Tenorth, "The assistive kitchen — a demonstration scenario for cognitive technical systems," in *IEEE 17th International Symposium on Robot and Human Interactive Communication (RO-MAN), Muenchen, Germany*, 2008, invited paper.

[2] F. Gravot, A. Haneda, K. Okada, and M. Inaba, "Cooking for humanoid robot, a task that needs symbolic and geometric reasonings," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 462–467, 15-19, 2006.

[3] C. Burghart, R. Mikut, R. Stiefelhagen, T. Asfour, H. Holzapfel, P. Steinhaus, and R. Dillmann, "A cognitive architecture for a humanoid robot: A first approach," *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, pp. 357–362, 2005.

[4] M. Fox and D. Long, "Modelling mixed discrete-continuous domains for planning," *Journal of Artificial Intelligence Research*, vol. 27, pp. 235–297, 2006.

[5] M. Beetz, M. Buss, and D. Wollherr, "Cognitive technical systems — what is the role of artificial intelligence?" in *Proceedings of the 30th German Conference on Artificial Intelligence (KI-2007)*, J. Hertzberg, M. Beetz, and R. Englert, Eds., 2007, pp. 19–42, invited paper.

[6] M. Beetz, *Plan-based Control of Robotic Agents*, ser. Lecture Notes in Artificial Intelligence. Springer Publishers, 2002, vol. LNAI 2554.

[7] A. Müller, A. Kirsch, and M. Beetz, "Transformational planning for everyday activity," in *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, Providence, USA, September 2007, pp. 248–255.

[8] C. Fellbaum and I. NetLibrary, *WordNet: an electronic lexical database*. MIT Press USA, 1998.

[9] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira, "An introduction to the syntax and content of Cyc," *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, pp. 44–49, 2006.

[10] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2003, pp. 423–430.

[11] D. McDermott, "A Reactive Plan Language," Yale University," Research Report YALEU/DCS/RR-864, 1991.

[12] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson, "Mining models of human activities from the web," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 573–582.

[13] J. Zelek, "Human-robot interaction with minimal spanning natural language template for autonomous and tele-operated control," in *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1997. IROS '97.*, vol. 1, Sep 1997.

[14] S. Tellex and D. Roy, "Spatial routines for a simulated speech-controlled vehicle," in *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. New York, NY, USA: ACM, 2006.

[15] N. Mavridis and D. Roy, "Grounded situation models for robots: Where words and percepts meet," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 4690–4697.

[16] F. Wu and D. S. Weld, "Autonomously semantifying wikipedia," in *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. New York, NY, USA: ACM, 2007, pp. 41–50.

[17] E. Mueller, "Modelling Space and Time in Narratives about Restaurants," *Literary and Linguistic Computing*, vol. 22, no. 1, p. 67, 2007.

[18] R. Schank and R. Abelson, *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1977.

[19] R. Gupta and M. J. Kochenderfer, "Common sense data acquisition for indoor mobile robots," in *In Nineteenth National Conference on Artificial Intelligence (AAAI-04*, 2004, pp. 605–610.

[20] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D Point Cloud Based Object Maps for Household Environments," *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge)*, 2008.