Recovering Plans from the Web

Andrea Addis[1a], **Giuliano Armano**[1b] **and Daniel Borrajo**[2c]
[1]Department of Electrical and Electronic Engineering, University of Cagliari, Italy
[2]Department of Computer Science, University Carlos III of Madrid, Spain
[a]addis@diee.unica.it, [b]armano@diee.unica.it, [c]dborrajo@ia.uc3m.es

Abstract

Planning requires the careful and error-prone process of defining a domain model. This is usually performed by planning experts who should know about both the domain in hand, and the planning techniques (including sometimes the inners of these techniques or the tools that implement them). In order planning to be widely used this process should be performed by non-planning experts. On the other hand, in many domains there are plenty of electronic documents (including the Web) that describe processes or plans in a semi-structured way. These descriptions mix natural language and certain templates for that specific domain. One such examples is the *www.WikiHow.com* web site that includes plans in many domains, all plans described through a set of common templates. In this work, we present a suite of tools that automatically extract knowledge from those unstructured descriptions of plans to be used for diverse planning applications.

Introduction

We are assisting to a continuous growth in the availability of electronically stored information. In particular, the Web offers a massive amount of data coming from different and heterogeneous sources. Most of it is in an unstructured format as natural language (blogs, newspapers) or semi-structured mixing some structure with natural language descriptions and predefined ontologies as in Wikipedia[1], eBay[2], Amazon[3], or IMDb[4]. In Wikipedia, a style template has to be filled in for each category belonging to a hierarchical structure of topics. In commercial sites as eBay, the online auction and shopping website, Amazon, an American-based multinational electronic commerce company website, a predefined list of mandatory attributes within its category is provided for each article. Also a more specialized knowledge base, IMDb, the Internet Movie Database, provides a list of standard attributes such as authors, director, or cast for each stored movie. In the spirit of Wikipedia, WikiHow[5] is a wiki-based web site with an extensive database of how-to guides. They are provided in a standard format (template) consisting of a summary, followed by needed tools (if any), steps to complete the activity, along with tips, warnings, required items, links to related how-to articles, and a section for sources and citations.

Currently, the most overshadowing and noteworthy web information sources are being developed according to the collaborative web paradigm, also known as Web 2.0 (O'Reilly 2005). It represents a paradigm shift in the way users approach the web. Users (also called prosumers) are no longer passive consumers of published content, but become involved, implicitly and explicitly, as they cooperate by providing their own content in an *architecture of participation* (Burdman 1999). Nowadays, thanks to advanced publishing tools the semi-structured knowledge base is more common, but not yet dominant. Therefore, is becoming a primary issue to support applications that require structured knowledge to be able to reason (as in the form of ontologies), in handling with this enormous and widespread amount of web information. To this aim, many automated systems have been developed that are able to retrieve information from the Internet (Addis, Armano, and Vargiu 2008a; Camacho et al. 2005), to select and organize the content deemed relevant for users (Addis, Armano, and Vargiu 2008b; Birukov, Blanzieri, and Giorgini 2005). Furthermore there has been some work on ontology learning (Zhou 2007; Manzano-Macho, Gmez-Prez, and Borrajo 2008) pointing out how it is possible to solve the problem concerning the lack of structure of which the web often suffers. Thus, the structured format of the extracted knowledge is usually in the form of hierarchies of concepts (see for example the DMOZ project[6]) and this can help on developing many different kinds of web-based applications, such as mainly specialized or general purpose search engines, and web directories. Other applications need information in the form of individual actions more than structured hierarchies of concepts, or in the form of plans.

On the other hand, making planning a widely used technology requires its usage by non-planning experts. As

[1]http://www.Wikipedia.org
[2]http://www.eBay.com
[3]http://www.Amazon.com
[4]http://www.IMDb.com
[5]http://www.WikiHow.com
[6]http://www.dmoz.org

discussed in the abstract, this is something far from being a reality currently. So, there is a need for techniques and tools that either allow an interaction with domain experts in their usual language, or automatically (or semi-automatically) acquire knowledge from current sources of plans and actions described in semi-structured or unstructured formats. In the first case, there has been some work on knowledge acquisition tools for planning as GIPO (Simpson, Kitchin, and McCluskey 2007), techniques for domain models acquisition (Gil 1991; Wang and Veloso 1994; Yang 2005), or tools that integrate planning and machine learning techniques (Fernndez et al. 2007). In the second case, there has been very little work on building plans from human generated plans or actions models described in semi-structured or unstructured formats, as filling natural language descriptions on templates. Another field that could also benefit from this automatic (or semi-automatic) acquisition of plans is the goals/activities/plan recognition (Tapia, Intille, and Larson 2004), where most of its work assumes the existence of plan libraries that are manually coded. Examples are in the health environment (Sánchez, Tentori, and Favela 2008), helping aging persons to perform their daily activities (Pollack et al. 2003), or to assist a user on performing bureaucratic or tourism related actions (Castillo et al. 2008).

In this paper, we want to describe some work to bridge the gap between the lack of tools to automatically build plans and action models from semi-structured information, and the existence of this kind of knowledge in the Web, as is the case of WikiHow. This is similar to what is currently done in the Semantic Web, Information Retrieval or Ontology Learning fields. We have built a set of tools in a modular structured architecture, which automatically browses some specific category from the ones represented in WikiHow, analyzes individual plans in those web pages, and generates structured representations of the plans described in natural language in those pages. We believe this is an important step towards a massive usage of planning technology by users in that they can share plans as they are doing now through WikiHow in natural language, and then automatic tools build planning technology on top of those plans. This applies also to other domains as workflow applications, where most big organizations have written processes, or hospitals, where many standard procedures are described also in semi-structured formats. Also, we will encourage research in this topic by suggesting potential relevant tools to be built on top of our research for improving the construction of planning and plan recognition tools. The remainder of the paper is organized as follows: first the proposed architecture is depicted, and the constituting subsystems are separately analyzed. The experiments and their results are presented and evaluated. Finally, we draw some conclusions and outline future research.

From Unstructured Plans to Action and Plan Models

In this section we describe the Plan Acquisition Architecture (PAA) that performs the acquisition of plans and actions from semi-structured information. Then, we will describe the information source that we have used in this paper for showing how it works.

The PAA

Figure 1 highlights how the different subsystems, each wrapping different technologies, retrieve and transform the semi-structured information provided by web articles into a structured output. The output contains labeled and recognizable objects (e.g. work tools, ingredients), actions (e.g. cut, peel, fry) and plans (e.g. sequences of instantiated actions), so that they may be reused for planning purposes.

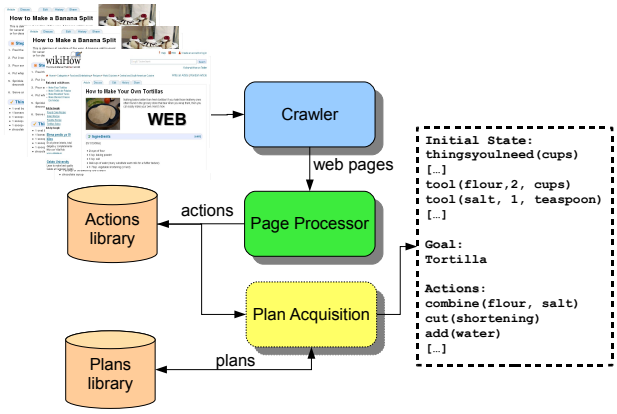

Figure 1: PAA at a glance.

The *Crawler* subsystem is devoted to crawl and store pages and category sections of a web site. The *Page Processor* subsystem is currently the core of PAA. It is aimed at analyzing web pages, storing only the relevant semi-structured content into an action library after performing an initial preprocessing. In particular (i) the goal, (ii) the initial state (in terms of required tools), (iii) the actions, (iv) tips (to be exploited as heuristic hints), (v) warnings (to be exploited as plan build constraints), and (vi) articles related to the selected page/plan are stored for each input. The *Plan Acquisition* subsystem processes this information to extract plans. The tools belonging to each subsystem, depicted in Figure 2 will be separately analyzed.

WikiHow: a Knowledge Source for Extracting Plans

WikiHow is a collaborative writing project aimed at building the world's largest and highest quality how-to manual. WikiHow currently contains more than 56,000 articles written, edited, and maintained primarily by volunteers. Each article contains the necessary tools and describes the sequence of actions required to reach the goal the page is concerned with. As an example, let us take a look at the page *Make Your Own Tortillas*,[7] reported in Figure 3, to better understand how the Page Processor subsystem parses its different sections. The relevant ground items that can be contained in each WikiHow web page are actions, tools, and related

[7] http://www.wikihow.com/Make-Your-Own-Tortillas

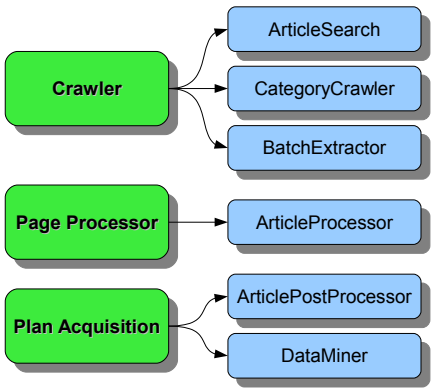

Figure 2: Subsystems of PAA and corresponding tools.

web pages (relatedwps), represented as A, T, and WP respectively. The Page Processor is the subsystem entrusted with processing the different sections of the page, their name being identified by a $< div\ id = NAME >$ HTML tag (e.g. $< div\ id = "ingredients" >$). Each section must be associated with a type, being one of the following:

- **actions**: a sequence of actions, representing the necessary steps to reach the goal. Examples of actions are *combine flour and salt*, *cut in shortening*;
- **tools**: a set of tools needed to reach the goal with different semantics depending on the selected category, e.g. ingredients for the *cuisine* category and mechanical tools for the *building stuff* category. They represent the initial state. Examples of tools are: *2 cups of flour*, or *1 spoon of salt*;
- **relatedwps**: other web pages related with the described task. Examples of related pages are *how to make Tortilla de Patatas*, *how to make Flour Tortillas*, *how to make Tacos*. This is usually not used within planning, but they open new possibilities for planning purposes, such as suggesting potentially relevant plans.

Since the Steps, Tips and Warnings sections, that are of type *actions*, the *ThingsYoullNeed* section of type *tools*, and the *RelatedWikiHow* section of type *relatedwps* are suggested by the WikiHow template layout, they occur in almost every page. They are parsed by default by the Page Processor, whereas further sections (e.g. *Ingredients* of type *tools* usually added by the person that compiles a recipe) have to be explicitly declared.

The Crawler

The Crawler subsystem includes tools devised to find an article using natural language and to find all the articles belonging to a specific category. Given the set of natural language queries, and the HowTo categories, the crawler can perform the following functions:

- ArticleSearch: given a user input stated as a natural language query, it finds all relevant articles, sorted by relevancy rank. Furthermore, it selects the highest ranking

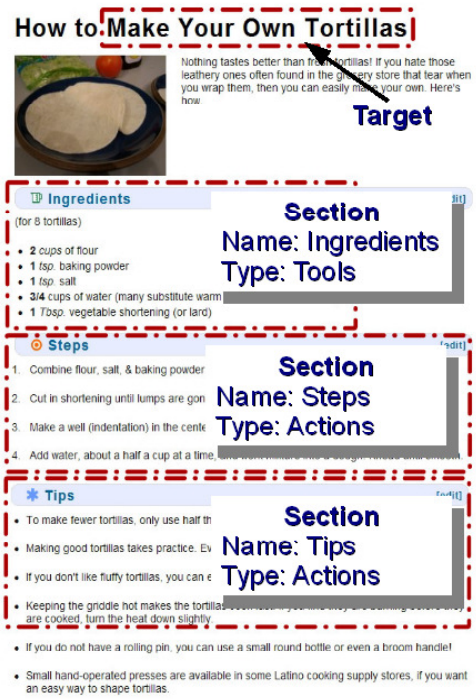

Figure 3: WikiHow sample web page

page and processes it. As an example, if the user enters the query *Make Tortilla*, the pages:

1. http://www.wikihow.com/Make-Your-Own-Tortillas
2. http://www.wikihow.com/Make-Tortilla-de-Patatas
3. http://www.wikihow.com/Make-Tortilla-Pizzas
4. http://www.wikihow.com/Make-Tortilla-Snacks
5. [...]

are suggested as relevant, and the first one (e.g. Make Your Own Tortillas) is automatically parsed by the ArticleProcessor (described later on)

- CategoryCrawler: permits to find all the articles belonging to a specific category. For instance, if the user enters the category recipes, *http://www.wikihow.com/Category:Recipes*, the crawler will find the current 3144 receipts belonging to its 167 sub-categories.
- BatchExtractor: applies the page processing to all the pages belonging to a category. It stores results in a file representing the category or in a database. Currently it can handle every JDBC-compliant database.

The Page Processor

The Page Processor subsystem includes tools for processing a web page, to facilitate the recognition of sections, together with the type of their content. Currently the Page Processor includes only the ArticleProcessor tool. The ArticleProcessor implements a function that, given as input a web page returns a tuple $< a, t, r >$ where a, t, and r, are a sequence

of actions, a set of tools, and related web pages, respectively. Each tuple can be seen as an augmented plan with information on its actions, a, initial state, t and related plans r. This processing phase tries to remove all noisy information while avoiding to lose the relevant one required for further processing. The ArticleProcessor embeds an HTML parser devoted to cope with several errors, such as the ones related to the `<div>` closure tag, incoherences with the `id` attribute declarations, changes on the main structure of the page, or bad formatted HTML code.

This subsystem incorporates the current standard tools for processing natural language, such as stemming procedures, which remove inflectional and derivational suffixes to conflate word variants into the same stem or root, or stopwording procedures which remove words with a low information content (e.g. propositions, articles, common adverbs) from the text. The semantic analysis is performed by using WordNet,[8] a lexical database considered the most important resource available to researchers in computational linguistics, text analysis, and related areas. Its design is inspired by current psycholinguistic and computational theories of human lexical memory (Fellbaum 1998).

The raw content of a sentence is also preserved to permit further tools to re-parse it. As for the sections of type *actions*, the action itself of each sentence is recognized by identifying the verb or the corresponding compound. Furthermore, a set of parameters related to the action are stored and separated from the redundant part of the sentence. More specifically, both actions, tools and relatedwps can have related parameters. Next, we define the most relevant parameters of each type of information.

The parameters related to actions are:

- **action**: the main action, represented by a verb or a compound
- **components**: the components of the action
- **components-st**: the stopwording+stemming of the *components* field
- **plus**: sentences related to the action considered redundant
- **plus-st**: the stopwording+stemming of the *plus* field
- **raw**: the raw content of the sentence

As an example of actions parsing, given two of the input sentences in the Tortillas Web page "Combine flour, salt, and baking powder in a large medium large bowl." and "Cut in shortening until lumps are gone.", the output of the parser would be:

ACTION:combine; COMPONENTS:flour; PLUS:salt, amp baking powder in a large medium large bowl; COMPONENTS-ST:flour; PLUS-ST:bowl larg bake powder amp medium salt; RAW:combine flour, salt, amp baking powder in a large medium large bowl. and *ACTION:cut; COMPONENTS:in shortening; PLUS:until lumps are gone; COMPONENTS-ST:shorten; PLUS-ST:lump gone until; RAW:: cut in shortening until lumps are gone.*

[8] http://Wordnet.Princeton.edu/

As for the elements of type tools, the information concerning *Quantity*, *Unit of measure* (e.g. units, grams, centimeters, cups, spoons) and *name of the ingredients* is stored. So their parameters are:

- **quantity**: the quantity/measure of the tool
- **type**: the unit of measure of the tool
- **tool**: the name of the tool
- **tool-st**: the stopwording+stemming of the *tool* field

As an example, given the sentence "$< b > 2 < /b >$ cups of flour", taken from the ingredients section of the *How To Make Your Tortillas* web page, the parser would generate: *QUANTITY:2; TYPE:cups; TOOL:of flour; TOOL-ST:flour; RAW:: $< b > 2 < /b >$ cups of flour.*

In the case of the relatedwps, only the name and the HTTP URL are stored. They serve as indexes in our plan data base for accessing other plans.

As an example of related web pages for the article Make Your Own Tortillas, it would generate the following two relations:

- URL:http://www.wikihow.com/Make-Your-Own-Tortillas; NAME:Make Your Own Tortillas;
- URL:http://www.wikihow.com/Make-Tortilla-de-Patatas; NAME:Make Tortilla de Patatas;

The Plan Acquisition

The plan acquisition subsystem includes tools that allow to create plans from web pages; in particular a post-processor that integrates semantic tools (i.e. the ArticlePostProcessor) and a suite of statistical tools (i.e. the DataMiner).

The ArticlePostProcessor

Given a web page, the ArticleProcessor builds its corresponding plan. For each action and tool, the ArticlePostProcessor uses the information retrieved by the ArticleProcessor, encompassing in particular the semantical annotation, in order to define the end result in the form of an augmented plan. The plan representation contains information about

- The goal represented by the name of the web page
- The initial state in the form of needed tools represented as a tuple <name of the section, quantity/measure, unit of measure, name of the tool>
- The actions to reach the goal represented as a tuple <name of the section, ordinal number of the action, action name, action tools (if any)>

As an example, the following is an extracted plan for the web page How To Make Your Own Tortillas:

- **Goal**: make tortilla
- **Initial state**:
  - tool(ingredients,2,cup,flour):
  - tool(ingredients,1,tsp,salt):
  - tool(ingredients,1,cup,water):
  - [...]

- **Plan**:
  - action(steps,1,combine,{flour,salt});
  - action(steps,2,cut,{shorten});
  - action(steps,3,make,{indentation});
  - action(steps,4,add,{water});
  - action(steps,5,work,{mixture});
  - [...]

The DataMiner

The DataMiner is a tool containing data mining and statistical algorithms. Statistics extracted by this tool are useful to understand which component or action is most likely to be used or applied in a specific context, in order to build new plans, or which are the most common subsequences of actions. Experiments have been performed exploiting actions, goal, tools frequency tables, and goal → action, goal → tool correlation tables. If we express the correlation between X and Y as $C(X, Y) = F$, where F is the value of the frequency of how many times the object X appears in the context Y, an example of correlation between goal components and actions performed in the context of the receipts category is:

- C(cake, pour) = 19.12934%
- C(sandwich, put) = 19.01585%
- C(cream, add) = 17.94737%
- C(cake, bake) = 14.81189%

This highlights that, for instance, it's likely (with probability above 19%) to perform the action *put* when the goal is to make a *sandwich*. It will be also useful to analyze particular subsequences of plans in specific contexts (e.g. what are the most likely actions to be performed on an onion in receipts needing oil?).

Experiments

PAA has been developed in Java using the version 1.6.0.11 of the Sun Java Development Kit. NetBeans 6.5.1[9] has been used as IDE. For the experimental phase, a GUI and a Glassfish WebService integrating the architecture have been deployed.

We have applied PAA to three WikiHow categories:

- Receipts: http://www.wikihow.com/Category:Recipes
  - 3144 receipts parsed/acquired plans
  - 167 sub-categories found
  - 24185 different individual actions
- Sports: http://www.wikihow.com/Category:Team-Sports
  - 979 team sports parsed/acquired plans
  - 22 sub-categories found
  - 6576 different individual actions
- Travel destinations: http://[...]/Category:Destinations
  - 230 travel destinations parsed/acquired plans
  - 16 sub-categories found
  - 2017 different individual actions

The error on unrecognized actions (meaning that the current version of PAA could not be able to semantically parse some sentences and recognize their structure) is about 2%. Let us point out in advance that it is difficult to assess PAA, mainly due to the fact that correct representations in terms of plans are not currently available. Hence, with regard to the acquired plans, we did some ad-hoc analysis by manually inspecting some output plans. The analysis shows that the system performed rather well on plan extraction, considering the complexity of the semantical analysis tasks and the need to handle many outliers. In fact, parsing an HTML page, even if automatically generated from a php engine, is not trivial due to *code injection* during the compiling of the predefined structure, and to the addition of different sections depending on the context (e.g. *ingredients* in receipts, or *work tools* in machinery). Besides, sentences are structured in different ways and filled with different kinds of contents more than "simple" steps. For instance, some people add a lot of non descriptive text (e.g. from a step for the "Make Vodka" how-to: *Column stills produce purer alcohol because they essentially redistill the alcohol in a single pass*). Others add playful goals with facetious suggestions (e.g. from the "Microwave a Peep" how-to: *Don't hurt yourself with the fork. You will be a slave to the Peeps if you eat it at all*). Moreover, somebody slangs or adds actions not related to the goal (e.g. *Stir and enjoy!*). The preprocessor has to handle all this, other than trying to manage compound forms, exploiting redundant descriptions of the action into the sentence and attempting to discover tools not explicitly cited.

Clearly, it is impossible to perform perfectly well. However, we reached our goal to have a good tradeoff between information retrieved from a web page and information loss during the filtering process. Also, we obtained a reasonably good tradeoff between semantical comprehension and introduction of errors. Thus, even if the integrated technologies that compose our tools are subject to future improvements, they already gave us a base on which to work and play on collected information in the next future.

Conclusions and Future Work

In this paper, we described a work aimed at bridging the gap between the need of tools for automatically building plans and action models from semi-structured information and the existence of this kind of knowledge in the Web (e.g. WikiHow). We believe this work can encourage further research in this topic that can greatly help the massive application of planning to many real-world human related tasks. Experimental results show that the tools performed well on extracting plans, thus establishing a preliminary base on which to work.

As for future work, the *analysis of common subsequences* on multiple plans in the same domain will be performed. We will base this analysis on previous work on planning as macro-operators (Fikes, Hart, and Nilsson 1972), n-gram analysis of natural language tools (Dunning 1994), or asso-

[9] http://www.Netbeans.org

ciation rules learning (Kavsek, Lavrac, and Jovanoski 2003). This can be further used for performing case-based planning by recovering subsequences that include same specific object. For instance, subsequences of recipes that use a particular ingredient or tool in general. Also, it could be used by tools that help on inputing recipes on the WikiHow by suggesting previous plans subsequences for the ingredients. Furthermore, we plan to include a Planner subsystem that contains tools necessary to exploit the collected knowledge base (e.g. the plans library) *to build new plans*. Other uses of this library will be aimed at performing *plan recognition* from data coming from sensors and at matching such data against the plans recovered from the web, or at *acquiring complete action models* with preconditions and effects from the input plans (Yang, Wu, and Jiang 2007). We will also exploit different knowledge bases as *eHow*[10] an online knowledge resource offering step-by-step instructions on *how to do just about everything*. eHow content is created by both professional experts and amateur members and covers a wide variety of topics organized into a hierarchy of categories; or *HowToDoThings*[11] another hierarchically organized knowledge base of how-to manuals, in order to make our architecture even more general and independent from the particular web site.

References

Addis, A.; Armano, G.; and Vargiu, E. 2008a. WIKI.MAS: A multiagent information retrieval system for classifying Wikipedia contents. *Communications of SIWN* 3(June 2008):83–87.

Addis, A.; Armano, G.; and Vargiu, E. 2008b. From a generic multiagent architecture to multiagent information retrieval systems. In *AT2AI-6, Sixth International Workshop, From Agent Theory to Agent Implementation*, 3–9.

Birukov, A.; Blanzieri, E.; and Giorgini, P. 2005. Implicit: an agent-based recommendation system for web search. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 618–624. New York, NY, USA: ACM Press.

Burdman, J. 1999. *Collaborative Web Development: Strategies and Best Practices for Web Teams*. Addison-Wesley Longman Ltd.

Camacho, D.; Aler, R.; Borrajo, D.; and Molina, J. 2005. A multi-agent architecture for intelligent gathering systems. *AI Communications, The European Journal on Artificial Intelligence* 18(1):1–19.

Castillo, L.; Armengol, E.; Onaindía, E.; Sebastiá, L.; González-Boticario, J.; Rodriguez, A.; Fernández, S.; Arias, J. D.; and Borrajo, D. 2008. SAMAP: An user-oriented adaptive system for planning tourist visits. *Expert Systems with Applications* 34(34):1318–1332.

Dunning, T. 1994. Statistical identification of language. Technical report.

Fellbaum, C. 1998. *WordNet An Electronic Lexical Database*. Cambridge, MA ; London: The MIT Press.

Fernndez, S.; Borrajo, D.; Fuentetaja, R.; Arias, J. D.; and Veloso, M. 2007. PLTOOL. A KE tool for planning and learning. *Knowledge Engineering Review Journal* 22(2):153–184.

Fikes, R. E.; Hart, P. E.; and Nilsson, N. J. 1972. Learning and executing generalized robot plans. *Artificial Intelligence* 3:251–288.

Gil, Y. 1991. A domain-independent framework for effective experimentation in planning. In *Proceedings of the Eighth International Workshop (ML91)*, 13–17.

Kavsek, B.; Lavrac, N.; and Jovanoski, V. 2003. *Lecture Notes in Computer Science*, volume 2810. Springer Verlag. chapter APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery, 230–241.

Manzano-Macho, D.; Gmez-Prez, A.; and Borrajo, D. 2008. Unsupervised and domain independent ontology learning. combining heterogeneous sources of evidence. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

O'Reilly, T. 2005. *What is Web 2.0, Design Patterns and Business Models for the Next Generation of Software*. O' Reilly.

Pollack, M.; Brown, L.; Colbry, D.; McCarthy, C.; Orosz, C.; Peintner, B.; Ramakrishnan, S.; and Tsamardinos, I. 2003. Autominder: An intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems* 44:273–282.

Sánchez, D.; Tentori, M.; and Favela, J. 2008. Activity recognition for the smart hospital. *IEEE Intelligent Systems* 23(2):50–57.

Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. L. 2007. Planning domain definition using gipo. *Knowl. Eng. Rev.* 22(2):117–134.

Tapia, E. M.; Intille, S. S.; and Larson, K. 2004. *Pervasive Computing*. Springer Berlin / Heidelberg. chapter Activity Recognition in the Home Using Simple and Ubiquitous Sensors, 158–175.

Wang, X., and Veloso, M. M. 1994. Learning planning knowledge by observation and practice. In *Proceedings of the ARPA Planning Workshop*, 285–294.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence* 171(2-3):107–143.

Yang, H. C. 2005. A general framework for automatically creating games for learning. In *Proceedings of the fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, 28–29.

Zhou, L. 2007. Ontology learning: state of the art and open issues. *Inf. Technol. and Management* 8(3):241–252.

[10] http://www.eHow.com/

[11] http://www.howtodothings.com/