The Goal Behind the Action: Toward Goal-Aware Systems and Applications

DIMITRA PAPADIMITRIOU, University of Trento, Italy
GEORGIA KOUTRIKA, HP Labs, USA
JOHN MYLOPOULOS and YANNIS VELEGRAKIS, University of Trento, Italy

Human activity is almost always intentional, be it in a physical context or as part of an interaction with a computer system. By understanding why user-generated events are happening and what purposes they serve, a system can offer a significantly improved and more engaging experience. However, goals cannot be easily captured. Analyzing user actions such as clicks and purchases can reveal patterns and behaviors, but understanding the goals behind these actions is a different and challenging issue. Our work presents a unified, multidisciplinary viewpoint for goal management that covers many different cases where goals can be used and techniques with which they can be exploited. Our purpose is to provide a common reference point to the concepts and challenging tasks that need to be formally defined when someone wants to approach a data analysis problem from a goal-oriented point of view. This work also serves as a springboard to discuss several open challenges and opportunities for goal-oriented approaches in data management, analysis, and sharing systems and applications.

Categories and Subject Descriptors: H. [**Information Systems**]: Goal-Aware Data Management

General Terms: Goal, Intention, Design, Models, Recognition, Exploitation

Additional Key Words and Phrases: Goal-aware, intention, actions, interaction, plans, operationalization, queries, recommenders, interactive systems, retrieval

ACM Reference Format:
Dimitra Papadimitriou, Georgia Koutrika, John Mylopoulos, and Yannis Velegrakis. 2016. The goal behind the action: Towards goal-aware systems and applications. ACM Trans. Database Syst. 41, 4, Article 23 (November 2016), 43 pages.
DOI: http://dx.doi.org/10.1145/2934666

1. INTRODUCTION

A fundamental aim of most information systems is to provide users with the information that best serves their needs. Retrieving items related to a keyword query [Bergamaschi et al. 2011], similar to an example [Mottin et al. 2014], related to existing results [Bi et al. 2015], proposing related queries [Cheema et al. 2014], and recommending items [Zhang and Wang 2015] are examples of alternative approaches serving this purpose. To be successful, the systems analyze user actions (e.g., clicks), user logs (e.g., past queries), or preferences explicitly stated (e.g., ratings), with the aim

Authors' addresses: D. Papadimitriou and Y. Velegrakis, University of Trento, Via Sommarive 9, 38123 Trento, Italy; emails: d.papadimitriou@unitn.it, velgias@disi.unitn.eu; G. Koutrika, HP Labs, 1501 Page Mill Rd, Palo Alto, USA; email: koutrika@hp.com; J. Mylopoulos, School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Avenue, Ottawa ON K1N 6N5, Canada; email: jmylopou@eecs.uottawa.ca.

DOI: http://dx.doi.org/10.1145/2934666

of deriving knowledge that will help understand the user needs and drive the selection of items, the recommendation of elements, and the actions of the system.

A limitation of such approaches is that they do not take into consideration the goals, that is, what the user wants to fulfill when performing an action. By action, we mean a specific query, a purchase, a selection of a document from a result set, a request for a recommendation, and so forth. It has already been recognized that behind every action in real life, there is a goal to be fulfilled [Newell 1982]. Goals rationalize and provide context and meaning to actions; thus, they have become a topic of extensive studies in sociology, psychology, and cognitive science [Austin and Vancouver 1996; Sadri 2014; Sen et al. 1986; Thompson and McEwen 1958]. A goal is fulfilled through a sequence of actions. A specific action may serve different goals, and a goal may be fulfilled through different action sequences. Thus, in information systems, characterizing an element as interesting or relevant to a user based only on the knowledge of past actions does not suffice. It is important to consider the reasons that these actions have been performed. In other words, to serve information that is a better fit to the user expectations, systems need to take into account user goals.

Goals have already been captured and applied to improve the effectiveness of a number of different applications [Papadimitriou et al. 2015]. Intelligent interfaces [Lieberman 2009] is one such example, in which the system tries to guess what the user intends to do from a single or from a number of user actions, and then adjusts the interface options accordingly so that the user can achieve the intended task faster and more conveniently. Story generators and computer games [Gold 2010; Meehan 1981] are extensively using techniques to guess the user goal and adjust the way the story is going to evolve in the future. Predictions of interactions and next actions, such as queries, likes, purchases, downloads, and comments [Chelmis and Prasanna 2012; Cheung and Lee 2010; Sadikov et al. 2010], are also benefiting from the consideration of goals. Goals have also been used in real environment applications, for instance, in applications that help the elderly. There, when the system recognizes a goal, it can suggest or even perform certain actions that will help the elder successfully fulfill the intended tasks [Geib and Goldman 2001; Jarvis et al. 2004].

There are few surveys on goal exploitation, and they focus mainly on goal recognition. One of these surveys studies ways through which a goal can be recognized by observing the actions that a system agent performs and provides an overview of techniques on how the recognized goals can be used in decision making [Anh and Pereira 2013]. Along the same lines, another survey emphasizes plan recognition and probabilistic methods [Armentano and Amandi 2007], while a third one approaches the challenge of goal recognition through logic-based formalisms [Sadri 2014]. All of these works are agent oriented; thus, they consider approaches typically employed by agents. They have an Artificial Intelligence (AI) flavor, ignoring issues like performance or usability. Furthermore, they have not considered works in areas like recommendation systems or information retrieval, and it is hard to see how the presented approaches can be adopted by other areas. Finally, they do not provide any generic formal definition of what a goal is, nor do they define related concepts.

We believe that goals can offer a great opportunity for improving the effectiveness of a number of data management systems [Papadimitriou 2016]. Many such systems nowadays are highly interactive, and the user actions can be recorded and leveraged so that the systems become more proactive, adaptive, and efficient. Traditional query processing, for instance, is based on the identification of the elements in the repository that satisfy the query conditions. Knowing the goals that a user is trying to fulfill, the system may restrict further the results to only those that are of real value to the user. This can be applied also in big data analytics and interactive data exploration, where results can be offered that better fit the user needs and expectations. As another

example, recommendation systems are based on past users' actions to identify and recommend possible elements of interest. By nature, these systems recommend items that are as close as possible to the user's history or the user's environment. Through goals, they can be adapted to also consider the future, by proposing items not because they are similar to something in the user's past, but because they help in fulfilling the user plans for the future. This same argument applies in information retrieval, where goals may be exploited to enhance the retrieved results with additional items that help fulfill a certain user goal.

The aim of this work is to provide an extensive and comprehensive study of the techniques and the ways that goals have been used in different fields, in order to provide the required background and framework for exploiting goals in building better data management systems. For this, we do not focus only on a specific area or discipline, but we study all the goal-related approaches together under a single prism. To the best or our knowledge, we are the first to provide such a complete and global study alongside a generic formal definition of goals and the related concepts. We intend to enable the exploitation of goal management methods in fields that have not considered goals in the past, significantly improving in that way their functionality. We show that this is possible and present the mechanisms for achieving it.

In a *goal-aware system*, there are three different tasks that the system should be able to perform. First, it has to be able to model and store user goals. Second, it should be capable of identifying the goals given a set of observed actions that the user has performed. Recall that goals are rarely mentioned explicitly or communicated, but instead exist in the users' minds. Last, but not least, the system should be able to exploit the recognized goals and adapt its functionality and output accordingly.

We organize our study around these three main axes as follows:

Goal modeling. To exploit goals, a system needs, first of all, to obtain a collection of goals alongside the knowledge of how these goals can be fulfilled. Goal modeling is challenging. Several approaches have been studied in applications and environments phenomenally disconnected. For example, we meet goals in (1) software for computers or other electronic devices (e.g., for providing intelligent interfaces); (2) the web as a collection of resources, where goals can be incorporated into query answering; (3) limited physical environments, where one or more sensor-based intelligent systems act (e.g., activity recognition sensors that send personalized activity reminders); and (4) physical locations monitored by sensors (e.g., an airport monitored for suspicious behavior).

We categorize goal modeling approaches according to the source of data used for the construction of the model. In particular, we consider those that are based on

—*complete records* of all the required information about goals, actions, and plans;
—*taxonomy records* containing actions matched to classes of a goal taxonomy;
—*corpuses* containing all the information for a *subset* of the goals, actions, and so forth; and
—*behavior theories* providing all the required information about human goals and actions within a specific environment.

Goal recognition. To recognize goals, one needs to observe user actions within an environment and identify patterns that lead to the satisfaction of user goals. User actions include queries, purchases, menu item selections, free-text input, preference statements, publishing multimedia objects, moves in a natural environment or moves of certain human parts, user interactions, and so forth. Data mining techniques, such as association rules that could capture knowledge in the form of "a set of actions X is followed by the set of actions Y with high probability," can provide useful correlations

among observed actions and system conditions. However, they cannot answer the question of "what the user wants to achieve with these actions," that is, identify and assign a goal to these actions [Wilcox and Bush 1992]. Goal representation is tightly coupled to goal recognition, and thus, we study them together.

Goal exploitation. Understanding a user's goal as the user interacts with the system can help the system to adapt its operation, personalize its responses, and facilitate the user in achieving his or her goal. For instance, imagine a user of a text editor, who opens the submenu for tables but changes nothing, then checks the printing settings a couple of times. The user is probably trying to print a table so that it fits the page. Knowing the user goal, the system can facilitate, for example, by highlighting related menu options. Exploitation of goals can save the user from performing irrelevant steps, or putting in extra effort, and the system from unnecessary operations and costly computations [Armentano and Amandi 2009; Carberry 1983; Gold 2010; Zhe et al. 2010]. Or it can make available knowledge, information, or any other type of response that is derived consider the current goal of the user [Broder 2002; Maragoudakis et al. 2007; Sadikov et al. 2010].

The exploitation of goals in practice is strongly dependent on the application scenario. We categorize existing approaches across two dimensions:

—*Exploitation through dynamic environment changes*, that is, changes in the environment states by taking into consideration the inferred goal(s)
—*Exploitation through system responses*, that is, responses to user requests performed by algorithms that embrace goals into their core functionality (i.e., the system performs its tasks considering the inferred goal)

Goals can be exploited at any given point of a system's lifecycle. During requirement specification, goal-oriented approaches aim at capturing the objectives a system should achieve [Mylopoulos et al. 1999]. Since the focus of this study is on systems, we do not consider such works. Furthermore, we do not consider studies on goals and human behavior from a psychological and sociological perspective.

The article is organized as follows. Section 2 provides a unified overview of goals and the related concepts. This section is meant to offer a formal concept map that can work as a reference point for any goal-aware system. Section 3 presents different techniques for goal modeling and recognition. Section 4 describes how the tasks and concepts described in Section 2 have been exploited so far by the existing systems and explains the benefits they offer to the functionality of the applications. Finally, Section 5 discusses several open challenges and opportunities for goal-enhanced approaches in data management and analysis.

2. GOAL-AWARE SYSTEMS

2.1. Key Concepts

Before studying the way goals have been used, it is necessary to establish a common terminology and formally define a number of concepts.

We assume the existence of a countable set $\mathcal{U}$ of *actors*, which can be persons or (software) agents. Actors live and operate in an environment, performing *actions* that affect it. *Environments* can be natural, such as a room monitored by sensors, or virtual, that is, created by a computer program. To realize the effects that actions have on an environment, we assume that an environment has a countable number of states. The states are specified by a number of factors that are modeled as variables. In particular, we assume the existence of a countable set $\mathcal{V}$ of variables. Each variable $v \in \mathcal{V}$ is associated with a domain D_v, the values of which are the possible instantiations of the variable and are referred to as the *states of the variable*.

Definition 2.1. An *environment* E is a finite set $\{v_1, v_2, \ldots, v_k\} \subseteq \mathcal{V}$. The variables $v_1, v_2, \ldots, v_k$ are referred to as the *environment variables* and the number k, denoted as $|E|$, is the *cardinality* of the environment. A *state*, or *instance*, of the environment is a set $\{S_{v_1}, S_{v_2}, \ldots, S_{v_k}\}$, with $S_{v_i} \in D_{v_i}$, for $i = 1..k$.

The symbol $\mathcal{S}^E$, or simply $\mathcal{S}$, will denote the set of all possible states that an environment E can be. The state of an environment is changed by actions performed by the actors or by external factors, for example, a variable that represents time changes independently of any actor actions.

Definition 2.2. An *action* is a function $act{:}\mathcal{S}^E \rightarrow \mathcal{S}^E$, expressed as a conjunction of "v=S_v" pairs, where v is an environment variable of E and $S_v \in D_v$.

For brevity, we will write $(v_1, \ldots, v_i) \overset{act}{\rightarrow} (S_{v_1}, \ldots, S_{v_i})$ to denote $v_1 = S_{v_1} \wedge \cdots \wedge v_i = S_{v_i}$. Furthermore, if for an action act, it holds that $act(S) = S$, where $S \in \mathcal{S}^E$, then the action will be said to have no effect on the environment state. Finally, the symbol $\mathcal{A}$ will be used to denote the set of all possible actions.

Example 2.3. Consider an environment E={place, time, status} that describes the position of a person at some point in time, and whether he or she is alone or not. Assume that at the current moment it is night and the person is at the sea alone. The current state of the environment is S_{curr} = {"*Sea*", "*Night*", "*Alone*"}. An action $(time) \overset{act_1}{\rightarrow}$ ("*Day*") will bring the environment into the new state S_{new} = {"*Sea*","*Day*", "*Alone*"}, while the action $(place) \overset{act_2}{\rightarrow}$ ("*Mountain*") will lead into the new state S_{new} = {"*Mountain*", "*Night*", "*Alone*"}.

By definition, an action can always be executed. However, there are many practical scenarios in which it is important to restrict when this can happen. For this reason, an action may be associated to a set of *preconditions* for its execution.

The actions that the actors perform are not random but are performed for a reason; that is, the actor wants to achieve a *goal*.

The term *goal* has been defined in different contexts as the point that marks the end of a process, the purpose toward which an endeavor is directed, an objective,[1] or a desired state of affairs. All these definitions converge into a generic description of a goal as one or more desired states described through some common properties.

Definition 2.4. A *goal* g in an environment E is a Boolean expression of environment variables of E. The goal is *fulfilled* (or *achieved* or *satisfied*) in a state S of the environment E, and denoted as $S \models g$, if after the replacement of each environment variable in the Boolean expression g with its state in S, the expression evaluates to true. The set of all possible goals is denoted by $\mathcal{G}$.

Example 2.5. Consider the environment of Example 2.3 and the desire of the person to go to the mountains during the day. This desire can be modeled as the goal g: $((place = \text{"}Mountain\text{"}) \wedge (time = \text{"}Day\text{"}))$. Note that the goal is independent of the state of the variable *"status."*

In order to fulfill their goals, actors are typically making plans on what actions to perform. This is known in the literature as the *operationalization* of a goal [Dalpiaz et al. 2014].

Definition 2.6. A *plan* is a sequence $\langle act_1, act_2, \ldots, act_n \rangle$ of actions. The *operationalization* of a goal g in an environment E with a state S is a plan $\langle act_1, act_2, \ldots, act_n \rangle$ for

[1]http://dictionary.cambridge.org, http://oxforddictionaries.com.

which $S'=act_n(act_{n-1}(\ldots act_2(act_1(S))\ldots))$ and $S' \models g$. Such a plan is also referred to as a *successful plan* for this goal. Lack of a successful plan makes the goal *infeasible*.

Example 2.7. The plan that consists of the two actions mentioned in Example 2.3, in any sequence, is a successful plan for the goal g: ((*place* = "*Mountain*") $\wedge$(*time* = "*Day*")) since the final state of the system after the execution of these two actions is S' = {"*Mountain*", "*Day*", "*Alone*"}, which satisfies the goal. The plan with the additional action (*status*) $\stackrel{act_3}{\rightarrow}$ ("*WithCompany*") is also a successful plan since it brings the system in the state S" = {"*Mountain*", "*Day*", "*WithCompany*"}, which also satisfies the goal.

The *implementation* (or execution) of the plan is the execution of its actions. The *lifetime* of a goal consists of the states that the environment goes through from the time a goal was set to the time a state was reached in which the goal is satisfied.

People often talk about how close they are in achieving a goal, or to what degree a goal is fulfilled. Hence, there is a notion of proximity between the current state of the environment and the set of states that satisfy the goal, that is, the set $\mathcal{S}_g = \{S |\ S \models g\}$. To quantify this proximity, we assume the existence of a *fulfillment function*, $scr_g: \mathcal{S} \rightarrow [0, 1]$, which is a scoring function with $scr_g(S) = 1$, for every $S \in \mathcal{S}_g$, and $scr_g(S) \neq 1$ otherwise. Such a scoring function is typically goal and application specific, since it depends on what factors of the environment are considered important and how much. A goal is *partially fulfilled* with respect to a state S if the value of this goal fulfillment function for the specific state is in $(0, 1)$.

In some cases, actors set goals that have no clear specification on when they are fulfilled. These types of goals are called *soft goals*, a term originally proposed in the field of goal-oriented requirements engineering [Mylopoulos et al. 1999], to be distinguished from the "hard" goals introduced in Definition 2.4, which are based on a Boolean function.

Definition 2.8. A *soft goal* is a function $g: \mathcal{S}^E \rightarrow \mathbb{R}$.

Intuitively, a soft goal provides a way to quantify whether one state of an environment is preferable to another, but there is no state in which it can be said that the goal has been satisfied. Due to this fact, a fulfillment function cannot be computed. However, between two states S_1 and S_2, it is typically said that the state S_1 has better fulfilled the soft goal g than S_2 if and only if $g(S_1) > g(S_2)$.

Example 2.9. An example of a soft goal is web searching in which users search for resources in order to get informed about a topic. For this purpose, the actions they perform are to submit keyword queries and click on the available web resources. It is hard to say that at some point the goal has been fulfilled (i.e., that the user knows everything about the topic). However, in an environment defined by features such as the diversity of the web resources, the position of the keywords within a page, and so forth (refer to Section 3.2, especially Model Construction & Table I for more examples), it can be defined as a function having as parameters a subset of the environment variables, that is, the goal function, to return whether one environment state is preferable to another. The environment states are actually matched to a number of web resources. According to the selected state, the respective web resources become available to the user.

Setting goals means typically some commitment to perform a sequence of actions for achieving that goal. The term *intention* is used to capture that commitment.

2.2. System Components and Tasks

We consider systems that allow us to store, retrieve, and discover information and knowledge from a data repository. In the physical or virtual workspace defined by the system, actors perform actions, such as submitting searches or sending requests (e.g.,

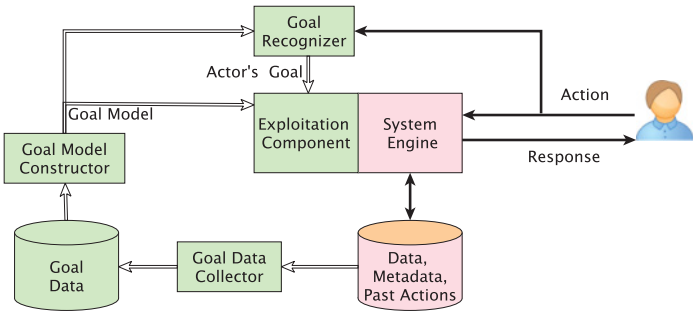

Fig. 1. A goal-aware system architecture.

for services), and the system responds by changes in the environment, services, or data in its environment. The *actors* are users or other systems/applications that interact with the system. Thus, the systems we consider are *interactive*.

The pink-colored part of Figure 1 illustrates the typical components of such a system. The bottom part is the data repository in which all the data is stored. This data is accessed by the main component of the system, the *System Engine*. The System Engine implements the main functionality of the system, that is, runs the main algorithms. A data analytics system would include in that component algorithms for data mining, text analytics, and statistical analysis that will be running over the stored data. A recommendation system would comprise different recommendation methods, while a traditional database system would entail methods for accessing the underlying data based on the specifications of the actor's queries.

Actors' actions support a goal that the actors want to achieve. For example, a user that poses a query "europabank, credit card, pay" is likely looking to pay the monthly statement balance rather than finding the cost for a new card. Respectively, the user would expect results that are different from the results when he or she tries to find the cost for a new card.

A system that ignores goals, that is, a *goal-agnostic* system, misses the big picture (i.e., "why is the actor doing this?"), and hence it cannot help the user as effectively as possible. In our earlier web search example, that would signify an increased user effort to achieve the goal. Knowing actors' goals can help the system understand their actions and adapt its behavior and functionality to a goal faster and more effectively, resulting in better system resource usage and user experience. A *goal-aware* system would require the components to record the goal-related information, analyze it, and then use the analysis results to recognize the goals of the actors and respond accordingly. These components are shown in green in Figure 1.

Goal Data Collector. In any data management system that allows interactions, it is common to track past user actions in the *data repository*. In a similar way, a goal-aware system also keeps information on the goal each such action serves (whenever it is available) and when these goals have been fulfilled. Consequently, the *data repository* is extended with the *goal repository* that contains information about the goals and the ways they can be operationalized, their actions, their preconditions, and their effects. Actions may be recorded directly in log files or indirectly, that is, via changes in the values of environment variables (e.g., temperature tracked by a sensor). Alternatively, goal data can be gathered from experts or user annotations. Figure 2(a) shows several alternatives for collecting goal data.

Goal Model Constructor. Goal data can be used to create the so-called *goal model*, which is a model used to recognize and subsequently to exploit the actor goals. The goal model construction can be done either in a top-down fashion, where experts or

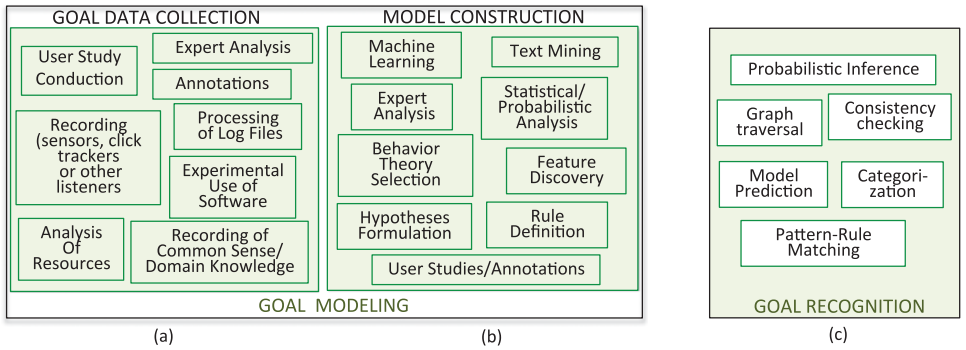

Fig. 2. Methods that may be employed during goal modeling and goal recognition.

actors themselves explicitly state their goals and actions, or in a bottom-up fashion, where models are constructed by *observing* and analyzing the actions of the actors in the system. The goal model is constructed offline and may be updated periodically as new actions and their goals are recorded in the goal repository. Figure 2(b) shows alternative approaches for building goal models. The goal data collector and the goal model constructor make up the *goal modeling* operations of a goal-aware system.

Goal Recognizer. With the goal model constructed, the next challenging task is the *goal inference* or *recognition*, that is, the ability to infer the goal(s) that an actor is currently pursuing by observing his or her actions [Sadri 2012]. The task is challenging because the actions provide only partial information. The idea is to recognize the goal way before all the actions that operationalize the goal have been completed. Depending on whether or not the actor wants to disclose the goals, the task can be characterized as *intended*, in which the actor tries to communicate his or her goals to the system; *keyhole*, in which the actor is unobtrusively observed and does not attempt to impact the recognition process; and *adversarial,* in which the actor is hostile and tries to hide his or her actual goal [Geib and Goldman 2001]. An example of the intended case are the natural language dialogues, where speakers explicitly try to communicate their goals. An adversarial example is cases of computer security and information warfare, where the actor tries to hide his or her intentions and the system tries to predict them in order to identify possible attacks. Finally, an example of the keyhole case is in query answering systems where users interact normally with the system without explicitly expressing their goals or hiding them from the system. Moreover, apart from the user's effort to keep his or her goals unrevealed, there are cases where we have *partial observability* of the environment for reasons such as sensor limitations, uncertain action logs, or privacy issues. In these cases, the matching between the actions that are actually performed and what is observed in the environment is not deterministic, making goal recognition more challenging [Hoelzl et al. 2012; Keren et al. 2016b]. For instance, an action may be matched to more than one set of effects on the environment. Figure 2(c) shows alternative goal recognition approaches for different goal models.

Plan recognition is an extension of goal recognition that aims at identifying not just the goal but also the plan followed by the observed actor in order to achieve his or her goal. In the basic case, it is assumed that an actor is pursuing a single goal using a deterministic set of actions; hence, a plan can be identified by matching these actions against the actions in a goal model such as a plan library representing the operationalization of the various goals. This decision cannot be done with complete certainty either, because the observations match partially and are not enough

to make a firm decision, or because the representations themselves are incomplete or uncertain.

Goal Exploitation Component. The recognized goal (and possible plan) can be exploited at runtime by the system algorithms. The module can select the responses to provide in order to drive the actor toward the fulfilment of the set goal. We highlight that the system "responses" are not necessarily returned data. They may be actions that the system takes even if not explicitly requested by the actor, for example, as performed by intelligent interfaces. This means that the goal exploitation module may provide different responses to the same request if the identified goals are different or if different plans are used for the operationalization of the goals. In a query answering system (e.g., a web search engine), the results retrieved with the existing techniques are all related to the query in general, but based on the goal that has been recognized, some may be more important than others. Thus, the results can be further processed to keep only those that will enable or facilitate the goal fulfillment. To achieve this functionality, the main system component (system engine) that performs data selection should be aware of the identified goal, its operationalization choices (i.e., the plans that lead to goal fulfillment), and the interaction history of the specific actor.

Note that goal exploitation is not necessarily an additional processing step. Existing algorithmic techniques can be adapted (or novel techniques can be designed) to take goals into consideration by performing the respective reasoning (that requires goal modeling and recognition) that has just been described.

3. GOAL MODELING AND RECOGNITION

Goal models, and by extension goal recognition techniques, are not restricted to a certain application scenario. The same modeling method can be used in different goal exploitation scenarios. The available data based on which the goal models are constructed together with the needs of the system determine the goal modeling approaches that can/should be used. Based on the *main source of goal data* used for the construction of the models, goal models can be clustered into models derived from *complete records*, *taxonomies*, *corpuses* (training data), and *behavioral theories*. In these methods, different ways to collect the goal data are applied, as we will see (Figure 2(a)). We also consider another alternative, collecting goal data from *text corpuses*. Text analysis is not offering a complete solution for goal modeling and inference. However, as we will see, challenging issues in terms of data management get raised by the discovery of goal knowledge in text data.

For each goal model type, we present the most common techniques for constructing the model based on the goal data, and for inferring the goals based on the current observations (goal inference). As we will see in Section 5, there is ground for further study and development of techniques for data management needs.

3.1. Based on Complete Records

Approaches *based on complete expert records* assume that *experts* provide all the information needed for building a model sufficient to match any set of observations to a latent goal (and possibly to a plan) within the examined environment. Goal recognition proceeds as follows. Initially, all the goals that require the actions observed so far are considered as candidates. As the actor continues to perform more actions, some of the candidate goals become logically infeasible due to missing actions in their implementation plans, violated preconditions, or other observations. These goals are excluded from the candidate set reducing the search space. During the check for the goal infeasibility, logic-based reasoning (mainly logic abduction) can be employed to provide explanations of the observations [Sadri 2012] when this is possible. The goal recognition task does

Problem Domain	hacking
Goal notation	(theft), (vandalism)
Actions	**Description**
$(reconnaissance) \overset{recon}{\rightarrow} (\text{“}true\text{”})$	make a reconnaissance, scan the system to determine vulnerabilities
$(breakedIn) \overset{break-in}{\rightarrow} (\text{“}true\text{”})$	exploit the system weaknesses
$(root) \overset{gain-root}{\rightarrow} (\text{“}true\text{”})$	gain entry break in escalate privileges gain root
$(exportDataRoot) \overset{steal}{\rightarrow} (\text{“}true\text{”})$	export desired data root
$(exportData) \overset{mod-webpage}{\rightarrow} (\text{“}true\text{”})$	export desired data
$(deleted-logs) \overset{clean}{\rightarrow} (\text{“}true\text{”})$	hide traces of presence

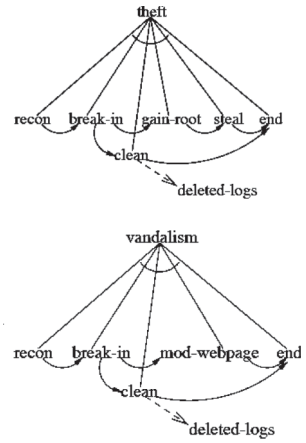

Fig. 3. Example of hierarchical plan library.

not always lead to some conclusion, but when it does, it returns one and only one goal. We identify three categories of approaches based on complete records: (1) plan libraries, (2) consistency graphs, and (3) action-centric representations.

3.1.1. Plan Libraries. Plan libraries could be characterized as a set of recipes describing alternative plans for implementing a set of goals in which the developers of the goal recognition system are interested. They contain information about (1) the set of actions that an actor is allowed to perform and (2) the preconditions of each action alongside its effect on the environment [Carberry 2001]. Plan libraries have to be *complete,* that is, to exhaustively describe all the actions that may be performed in the domain under study and all the alternative plans for all the possible goals. Moreover, they have to be *correct* since they lack mechanisms for handling inconsistencies [Sadri 2012].

Model Construction. The construction of a plan library demands a lot of effort by experts who master the problem domain. In general, experts try to organize the goals and actions into plans in a way that enables the generalization of the plans to cover new facts. Even psychological theories about how human observers understand the actions of others have been used in the task [Schmidt et al. 1978]. In many cases, the domain experts perform closed-world reasoning [Kautz 1991]; that is, they isolate a part of the world in which they are mainly interested and focus on the minimum sets of independent plans that explain the observations and are sufficient for fulfilling the goals of the observed agents.

Simple plan libraries may be represented as rules of the form $g \Rightarrow act_1, act_2, \ldots act_n$, where g denotes a goal, and $act_1 \ldots act_n$ is the sequence of actions that make up the operationalization of the goal. Actions in the literature may be represented as predicates; for example, *land*(*jet101*, *airbase1*) describes the action of *landing* of *jet101* to *airbase1* [Sadri 2012]. Respectively, in a simple environment E (refer to Section 2.1), the action $land \in \mathcal{A}$, for instance, would be defined as $(jet101_{Loc}) \overset{land}{\rightarrow}$ (“$airbase1$”), where $jet101Loc \in E$ and $D_{jet101Loc} = \{\text{airbase1, airbase2, airbase3}\}$.

Plan libraries can take a hierarchical form as well. Figure 3 depicts a simple hierarchical plan library capturing hacker goals in a web system [Geib and Goldman 2001]. On the left, the goals and the actions (together with a short description) are shown. On the right, the actual library is illustrated as two diagrams that correspond to two plans for the goals: (*theft*) and (*vandalism*). As we see, the actions in the plans

are not sequential; there exist links that determine *ordering constraints* among the actions.

Moreover, dashed lines represent the fact that a change in the environment is observed after the execution of an action (i.e., the action postconditions are illustrated). For instance, if *clean* is executed, a change in the environment will be observed; that is, the *deleted-logs* value will change from "false" to "true." This information about action effects can be critical to inferring the execution of unobserved actions (all actions except for clean are not observed). For example, in Figure 3, the ordering constraints allow us to conclude that in order for *clean* to be performed without being observed, an earlier unobserved *break-in* should also occur.

Goal Inference. Goal inference using plan libraries requires the detection of the plan that is consistent with the current observations. The correctness and completeness of the library is of major importance for the inference task. If the observations can be matched to more than one plan, the system should either wait until one or more actions exclude all the candidate plans but one, or return all the matching candidate plans. This approach is extremely sensitive to noise. One misidentified action may cause the exclusion of the real plan from the candidate set of solutions. However, inference in plan libraries is not always straightforward. In cases of adversarial recognition, in which hostile agents try to hide their actions from the system, there is no fully observable sequence of actions. Thus, probability distributions are introduced in the inference technique by algorithms such as Poole's PHA to first infer unobserved actions, based on the observed actions or the state of the system, and then infer the most probable goal [Geib and Goldman 2001]. The search space can be limited by considering the ordering of constraints and/or by excluding disabled actions. An action is considered to be disabled when in all the plans of the library, it is preceded by actions that have not been observed.

3.1.2. Consistency Graphs. Consistency graphs are graphs that consist of (1) proposition nodes that store the values of the environment variables, (2) nodes representing actions, (3) nodes representing goals, and (4) edges representing possible connections between nodes. Instead of constructing a complete plan library that includes all possible plans related to every possible goal, to build a consistency graph, one should *focus on defining what constitutes a valid plan*. In other words, focus on how the allowable actions can be combined to a plan that fulfills a goal according to the structure (i.e., the environment), the restrictions (i.e., action preconditions and postconditions), and the system functionality (i.e., the allowed actions and goals).

In contrast to goal recognition systems with complete libraries, where a goal is *consistent*, if there is a plan that starts with actions already observed leading to the goal, consistency graphs are able to recognize new plans as well [Hong 2000]. However, since the action restrictions are not modeled, consistency graphs cannot capture causal links among actions and goals. That makes them more appropriate for explaining past actions rather than making predictions.

Model Construction. Initially, all the actions and goals that are feasible in the examined environment are recorded by the domain experts and are inserted as nodes in the consistency graph. Then, all the action nodes and the goal nodes are fully connected to each other without checking for inconsistencies, that is, without checking whether the sequences of actions (plans) that are connected to a goal violate any conditions and if they actually lead to the fulfillment of the graph. Inconsistent goals are then repeatedly pruned from the consistency graph. The consistency control may be performed by the experts or automatically.

Initial State

(and(garbage)
(clean hands)
(quiet)
(not present)
(not dinner))

Goals

g1	(and(dinner)(present)(not garbage) (quiet))
g2	(and(dinner)(present)(garbage)(quiet))
g3	(and(dinner)(not present)(garbage)(quiet))

Actions

Preconditions	Effects	Short Explanation
(clean hands)	(dinner)	COOK. Before: your hands should be clean. After: dinner is ready.
(not present)	(present)	WRAP UP. Before: present is not ready After: present is ready.
(garbage)	(and (not garbage) (not cleanHands) (quiet))	CARRY GARBAGE. Before: garbage exists. After: there is no garbage, the agent's hands are unclean and there is silence

Fig. 4. An example of a state-variable representation of a simple goal recognition problem.

Goal Inference. The main idea is to reduce the set of candidate goals by eliminating the goals that cannot be explained by the actions that the actor performs. If more than one goal can be explained by the observed actions, consistency graphs cannot return a result since only one consistent explanation is possible.

3.1.3. Action-Centric Representations. Action-centric representations, originally proposed for classical planning problems, have recently been for goal and plan recognition by exploiting the progress in modern plan synthesis [Sun and Yin 2007; Ramırez and Geffner 2009].

Model Construction. In contrast to the construction of plan libraries, where domain experts record all actions possible, in action-centric representations, the modeled actions are the outcome of all the possible combinations of the environment variables to a set of pre- or postconditions. Thus, the size of the state space is exponential to the size of the set of environment variables. However, only the actions that have an important impact on the environment can be selected and stored. The modeling is done using propositional logic. Environment variables are modeled as propositions, the states of the environment as a set of propositions connected with logical symbols such as AND ($\vee$) and OR ($\wedge$), and action effects as two sets: the first set indicates which propositions will be removed from the environment state after the action is performed, and the second which propositions will be added. There are also models that instead of propositions use first-order literals.

An example is the STRIPS models that are expressed in the homonymous modeling language. STRIPS language has been initially suggested to represent planning problems for a specific software, a planner called STanford Research Institute Problem Solver (STRIPS) [Fikes and Nilsson 1971], but since then it has been used as a tool for representing the environment in planning and goal recognition problems independently of the STRIPS planner. In STRIPS models, often it is preferred to store only the action postconditions, that is, the changes that occur in the environment, instead of storing the complete outcoming environment states for reasons of efficiency. Moreover, other than states and goals, STRIPS includes operators. Operators represent the combination of two or more actions that cause a state transition that is considered important for the system.

Figure 4 shows a state-variable representation of a problem examined by Sun and Yin [2007] in which an actor may potentially prepare dinner, throw away the garbage,

and wrap up a present for his girlfriend. The representation consists of three actions, three goals, and the initial state. The actor goals are combinations of the propositions: (*dinner*), that is, dinner is prepared; (*present*), that is, present is wrapped up; (*garbage*), that is, garbage is not thrown away; and (*quiet*), that is, the agent's girlfriend is not woken up. The goal (*and*(*dinner*)(*present*)(*not garbage*)(*quiet*)), for example, describes that the actor wants to clean the room and prepare both the dinner and the present without waking up his girlfriend.

According to Section 2.1, the environment in the previous problem can be defined as $E = \{dinner, present, garbage, quiet, clean\ hands\}$. The domains of each of the respective environment variables $v_i \in E$ are $D_{v_i} = \{$"true","false"$\}$. Moreover, the goals g1, g2, g3 are determined by the respective environment states; that is, in g1 (dinner) will be translated into *dinner* = "true". Thus, g1 will consist of two environment states of $\mathcal{S}^E$: g1 = {{"true", "true", "false", "true", "true"}, {"true", "true", "false", "true", false}}. As the state of variable *clean hands* (which is the last variable) is not explicitly stated in g1, both environment states are desired.

The construction of the goal model starts with the initial state of the environment. After having produced all the possible actions (derived actions), the transition graph is constructed layer by layer, with the first layer being the initial state. Specifically, all the derived actions are examined, and if the preconditions of an action are satisfied or there are no other inconsistencies, the state is updated according to the effect of the current action. Inconsistencies include actions with inconsistent effects (effect-effect), actions where an action effect interferes with the precondition of another action (effect-precondition), or actions. The process is run recursively until the planning graph stabilizes.

Goal Inference. To infer goals from the derived transition graphs, search space algorithms are used, such as breadth-first search (BFS) and A^*, in combination with heuristics to boost performance. According to the BFS strategy, search is performed level by level; first, all the existing sibling nodes (nodes of the same level) are visited, then the next level of nodes is examined, and the procedure goes on until a node that represents a goal consistent with the observations is visited. The A^* strategy reaches the node representing the consistent goal by following the path of the minimum cost according to a cost function, such as minimizing the length of the path that contains all or some of the action nodes that have been observed. The starting search point is the current state of the system. Ramırez and Geffner [2009] showed how algorithms originally designed for planning can be slightly modified and used for plan recognition over a domain theory.

3.2. Taxonomy Based

Taxonomy-based approaches require the existence of a taxonomy of the possible goals, that is, a set of goal categories, within the system. The categorization is performed by experts and requires studying the existing actions (i.e., those that have already been posed), identifying the actor goals, and then building the taxonomy. In the area of goal-aware query answering, where they have been extensively used, the goal taxonomies were derived from *extended user studies* using questionnaires and *interactive tools* on web browsers tracking user moves such as clicks and form submissions in combination with *expert knowledge*. Examples of goal taxonomies [Broder 2002; Kang and Kim 2003; Lee et al. 2005; Rose and Levinson 2004] are briefly presented in Section 3.2.1.

In taxonomy-based approaches, there exist two types of actions: (1) the actions that initially trigger the functionality of the system and (2) the actions that become available after the system response. The first types of actions are the *user requests* or *queries*. For instance, in a web search engine, the requests are the *keyword queries*, while the actions

are the clicks on the web resources (e.g., pages, snippets) that the system returns to the user after the query is posed. The actor requests are used for goal inference, while the actions performed after the system response may be tracked by the system to evaluate the actor's satisfaction. Ideally, the actor would be satisfied by clicking a single web resource. Thus, the plan would consist of a single action. However, more actions, that is, clicks on the top related resources returned, may be required, creating longer plans.

Model Construction. Once the categories of the taxonomy are decided, the model that will classify a user query into one of the goal classes should be built. To build the model, experts select a number of environment variables that are considered appropriate for grouping the requests into the classes of the *goal taxonomy*. Manual classification can be used to understand the user goals and whether it is feasible and meaningful to incorporate them in the existing system. To automate (at least partially) this laborious task, the analysis of the involved resources can be employed; for example, in Web Information Retrieval, query logs and snippets have been used.

The selected environment variables are then used in rule-based annotators [Jansen et al. 2008; Lee et al. 2005; Li et al. 2006] or to train automatic classifiers, such as Support Vector Machines (SVMs) with RBF (Radial Basis Function) kernel [Baeza-Yates et al. 2006; Herrera et al. 2010].

The selection of environment variables (features) has to be performed very carefully, since for different domains the accuracy of the classification may increase or decrease significantly depending on the features. This is particularly evident in web searching [Herrera et al. 2010]. For web queries, the variables more widely used fall into five categories: (1) anchor-text-based features, (2) features regarding URLs, (3) query-based features, (4) features based on user's past clicks, and (5) page-content features (refer to Section 3.2.1 and Table I). They are typically extracted directly from the resource collection or from snippets retrieved via classical retrieval techniques or query logs. The selected environment variables constitute the main features. However, for each goal class, there may exist additional features that are essential for the definition of the goals to be inferred (e.g., the number of different involved web resources) or their diversity for informational queries for instance.

Goal Inference. Goals in taxonomy-based approaches are *soft goals*(refer to Section 2, Definition 2.8). Thus, goal inference is about defining a function over the environment variables. Considering the set of environment variables selected in the model construction step, the constructed model (i.e., the rule-based annotator or classifier) matches every new request in the system to one (or possibly more) of the goal categories. The goal class sketches (or else partially defines) the goal, since for each goal class, there is a set of conditions on the environment variables. The goal takes its complete form by extra conditions on the environment variables based on the request itself and the goal class. Hence, goal inference requires two tasks: (1) categorization of the request to one of the categories of the taxonomy and (2) definition of a function that captures conditions on environment variables based on the goal class and the request.

Some works have treated goal inference from user queries as a problem of query reformulation. They define goals in web search as sets of semantic concepts [He 2010] or as sets of "verb-object" pairs derived from the sentences that are implied by the queries [Chang et al. 2006]. Although these approaches showcase interesting results, the focus of this work is on goals that can (even approximately) describe a desired state of the environment, so we do not consider them further.

[*User satisfaction*]. An important aspect in taxonomies has been the evaluation of goal inference. Even if the model is accurate, it may not correctly classify a *request*,

mainly because of the subjective nature of soft goals. In contrast to hard goals (refer to Definition 2.4), soft goals (refer to Definition 2.8) are defined by a function $g{:}\mathcal{S}^E \rightarrow \mathbb{R}$. To define this function, one needs to know whether the actor will characterize a plan as successful, which is not possible to know in advance. To cope with this challenge, analysis of action patterns (e.g., sequences of queries or clicks within user sessions) have been employed by the information retrieval community. These analyses have been based on Markov Models [Hassan et al. 2010] or on Hierarchical Conditional Random Field techniques [He 2010].

3.2.1. Examples in Taxonomy-Based Approaches. In the area of goal-aware answering, to come up with the goal taxonomy, experts actually studied the information needs (sometimes called user intentions) that drive users to search on the web [Baeza-Yates et al. 2006]. Examples of *information needs* are *to be informed*, *to navigate to a site*, *to execute a transaction*, or *to get advice*. There has also been work on converting their textual descriptions, which are actually the labels of the goal classes, to a set of features for automatically assigning an incoming query to a goal class [Baeza-Yates et al. 2006; Herrera et al. 2010; Jansen et al. 2008; Lee et al. 2005; Li et al. 2006].

The Broder taxonomy [Broder 2002] was the first created. It was implemented by examining whether it is feasible to identify what users expect from a web search engine so as to consider their search successful and stop submitting similar queries. The outcome of that work was a taxonomy of user queries that reflects a categorization of latent user goals. The taxonomy describes three types of web queries: informational, navigational, and transactional. *Informational* queries consist of terms that describe or capture vague notions, for example, "*bugs,*" or consist of specialized terms, for example, "*Peruvian Dubia cockroach*." Both types of queries indicate that the desired target is a collection of links that will enlighten the user on the subject. *Navigational* queries consist of query terms that describe a specific URL, for example, the query "*american airlines home*" indicates that the user's most probable target is http://www.aa.com. Finally, *transactional* queries contain terms that indicate that the target URL enables a transaction such as downloading a file, buying an item, or watching a video. For instance, with the query "*Athens photo,*" the user most probably expects to get direct access to image files related to Athens.

User surveys proved that the consideration of Broder's query taxonomy in the selection of the query results has a positive impact on user satisfaction [Broder 2002]. Consequently, further research was triggered toward this direction and differentiations of the taxonomy have been developed toward more detailed ones [Jansen et al. 2008; Rose and Levinson 2004] or more abstract ones [Baeza-Yates et al. 2006]. For instance, Rose and Levinson [2004] elaborated Broder's taxonomy by dividing informational queries into five subcategories: (1) directed queries that express specific questions, (2) undirected queries that aim at retrieving all the available information about a topic, (3) list queries aiming at getting a list of candidates, (4) find-queries aiming at locating real-world services or products, and (5) queries aiming at getting advice, ideas, suggestions, or instructions. Going back to our earlier example of informational queries,"*bugs*" is an undirected query, while "*Peruvian Dubia cockroach*" is a directed query. Similarly, transactional queries are divided into four categories that express what exactly the users want to do. In particular, the user may want to (1) "download," (2) "view an item such as a video," (3) "interact via another program or service," or (4) "obtain a resource" (video file, text file, etc.).

The aforementioned taxonomies were *evaluated by* user studies, and the classification of the sample queries has been performed by experts based on information about the queries, on the results returned by commercial search engines, on the user clicks on the result list, and on other actions performed by the user before and after the submission

Table I. Environment Variables for Query Answering Systems

Env. Variable Types	Examples of Env. Variables
Anchor text [Herrera et al. 2010; Lee et al. 2005; Fujii 2008]	Similarity of the query with the top similar anchor texts
URLs of the web collection [Lee et al. 2005]	Similarity of the query with the URL (important for navigational queries)
Query formulation [Herrera et al. 2010; Jansen et al. 2008]	Num of query terms $\geq 2 \rightarrow$ informational queries
Past user clicks for the same query [Jansen et al. 2008; Lee et al. 2005]	Skewness of click distribution, for example, for navigational queries: only one click
Cue query terms and "important" terms of the web collection [Herrera et al. 2010; Jansen et al. 2008]	Domain suffixes (e.g., "edu"), terms related to pictures, games, and so forth, to interactions such as buy, chat $\rightarrow$ transactional queries; terms such as "how to," "ways of," and general terms $\rightarrow$ informational queries

of the query [Rose and Levinson 2004]. The reasons that determined the experts' classification decisions in some cases remain unclear, but there have been efforts to clarify and record them [Jansen et al. 2008].

Based on the goal class of a query (action), there are different answering policies expressed in the goal definition by conditions on the environment variables. For instance, for informational queries, goals should be satisfied by diverse web resources that cover the query from different perspectives, contain complementary and controversial information, and offer various levels of comprehension (for broad, deep, or quick understanding). Thus, when a query is matched to this goal class, the inferred goal should be defined as a function on variables such as resource diversity, content diversity, or size of resource. On the other hand, for navigational queries, goals should contain conditions on features describing click streams from logs, since click distributions reveal whether users consider a site to be the "expert" for certain queries. The function that defines the goal can be next used to reorder the results of a web search engine (refer to Section 4.1.2).

3.3. Corpus Based

In *corpus-based* methods, the *goal data* contains a set of alternative plans for a set of goals: the *plan corpus*. The plan corpus is used as a training set for statistical models that can make inferences for future observations. There is no ground truth about the environment; uncertainty expressed in probabilities is the factor that rules the outcome of the recognition process [Russell and Norvig 2003]. The trust on systems with probabilistic output under difficult critical circumstances is still an open issue [Atkinson and Clark 2013]. Thus, corpus-based approaches have been criticized when used in real-life domains such as health, defense, and transportation and are delegated to difficult and critical safety tasks, tasks of high cost in time or money, or in general tasks that have high impact on human lives. Nevertheless, these methods enable goal inference in environments where it is too expensive or infeasible to gather complete and certain knowledge about all the goals and the potential corresponding plans that may be followed by the observed actors. They only require a *plan corpus*, which will constitute a sufficient training dataset for developing an efficient statistical model. The two most widely used classes of probabilistic models in these cases are *Markov models* and *Bayesian networks*.

3.3.1. Markov Models. Markov models in their general form consist of nodes representing random (stochastic) variables and edges modeling conditional dependencies among the variables. The values of the random variables may be observed (known values) or may be inferred (unknown values). In the context of goal-aware systems, the values

of the *random variables* describe the *environment state*. The main inference task of Markov models used for goal inference is to compute the conditional probability of a sequence of observations given some evidence, that is, to check to what extent the current observations would be justified, if we assumed that the variable to be inferred had a specific value.

In Markov models, it is assumed that the probability that a random variable (i.e., an environment variable) will have a certain value in the future can be computed by observing only the recent past of a set of observations, that is, that an observed action act_i is only dependent on the current goal g and the n precedent observations (i.e., observed actions). This assumption is known as the *Markov assumption*. The number of previous observations is called the *order* of the model.

Model Construction. Learning a Markov network requires statistical analysis of the plan corpus to define the following probability distribution functions: (1) the distribution of prior probabilities $P(g)$ indicating the expectancy that a goal $g \in \mathcal{G}$ is being pursued by the observed actor; (2) the state transition function $P(S_i|S_{i-1}, g)$, where S_i,$S_{i-1} \in \mathcal{S}^E$ returns the probability that the system will move from the environment state S_{i-1} to S_i given that goal g is pursued, and (3) the observation function $P(act_j|S_i, g)$ or $P(S_{i+1}|S_i, g)$ that returns the probability that an observation will occur (either act_j will be performed or environment state $P(S_{i+1}$ will be observed) given that the system is in state S_i and that goal g is pursued.

Learning the probability distribution functions is typically done by performing a global search in order to figure out *which combinations of environment variables and weights* would give more accurate predictions within the plan corpus [Della Pietra et al. 1997]. This methodology is not efficient and is prone to make only locally optimal choices. To build a consistent and efficient probabilistic model, a two-step methodology has been suggested. First, a model (e.g., decision tree or logistic regression model [Lowd and Davis 2010; Wainwright and Jordan 2008]) is built for predicting the value of each variable of the domain with respect to the other variables. Then, the separate models are converted into a single Markov model.

Goal Inference. To infer goals in Markov models, first the goal probabilities are initialized by considering the prior probabilities function $P(g)$. Then every time an observation occurs, the goal probabilities are updated by taking into consideration the conditional probabilities functions defined when the model was created, and the goal g with the maximum probability is selected.

In cases in which the Markov assumption is valid, the probability of goal g can be estimated using the formula (Markov chain rule) $\prod_{i=1}^{n} P(act_i|g)$ (or $\prod_{i=1}^{n} P(S_i|g)$). This rule has the nice feature of composeability: new observations produce conditional probabilities, which are simply multiplied with the previous predictions.

In cases in which the Markov assumption is not valid, the complexity of the problem becomes very high (#P-complete) and approximate solutions are required. One widely used method is the Markov chain Monte Carlo (MCMC), such as the Gibbs sampling. MCMC performs probabilistic queries and provides answers by counting the number of samples that satisfy each query over the total number of samples [Wainwright and Jordan 2008]. By query is meant a sequence of observations that is answered given another sequence of observations that is called evidence. The sampling is not performed on the data but is calculated based on the joint probability of the random variables. In contrast to Markov networks that have been learned by methods such as probabilistic decision tree learners (DTSLs) [Lowd and Davis 2010], MCMC allows inference by standard techniques such as loopy belief propagation [Murphy et al. 1999] because its models represent consistent probability distributions.

3.3.2. Markov Model Variations. There are a number of interesting variations of Markov models that have been used in goal modeling.

N-Order Markov Models. In n-order models, the Markov assumption applies for n observations before the current observation while the evidence is the goal g. Hence, to check which goals explain better the observations, the n last observations are compared with subsequences of observations in the plans of domain goals in the plan corpus. Observations may be either actions or environment states, that is, $act_1 \dots act_n \in \mathcal{A}$, or $S_1,S_2 \dots S_n \in \mathcal{S}^E$. Respectively, plans are sequences either of actions (actions are directly recorded) or of environment states (actions are observed through environment state transitions) from a set of predefined actions or states. The value of n, that is, the order of the model, should be carefully chosen so as to create an expressive model (larger values of n) and at the same time keep the size of the search space traversable with low cost (smaller values of n). Due to their simplicity, n-order Markov models are very efficient but at the same time they may be ineffective in recognizing goals in cases of complex environments [Blaylock and Allen 2003].

Variable-Order Markov Models. In contrast to the n-order Markov models, in variable-order Markov models (VOMs), the probability of the current goal g is not defined by the same fixed number of previous observations. In other words, the order of the model varies based on the *specific observed realization* in the training data, known as *context*. Therefore, the use of VOM models can increase the accuracy of goal recognition by capturing longer regularities than n-order models, while controlling the size explosion of the search space caused when the n-order is increased. VOMs are learned over a finite alphabet consisting of all the available actions $\mathcal{A}$. States instead of actions are also able to be used. Thus, as in n-order models, the plans record either the performance of actions directly or the state transitions before the desired state, that is, the goal g. Armentano and Amandi [2009] suggested the use of Probabilistic Suffix Trees [Ron et al. 1994] to represent VOMs. For each domain goal, one PST is built to store the subsequences of variable length (plans) that are necessary and sufficient for modeling the corpus plan. Hence, a forest of PSTs is created. In order to optimize space and time efficiency, only the minimal subsequences of observations are preserved. In this case, *goal inference* becomes a classification problem of the sequence of observations to the most probable PSA. However, it is not a common classification problem since early predictions are very important [Armentano and Amandi 2009].

Hidden Markov Models. Hidden Markov Models (HMMs) are adequate for problems in which the current state of the system is not visible or cannot be identified with certainty, as in computer games and activity detection systems. This is because they do not require complete knowledge of the state of the system. Some or all of the environment variables of the partially observable or hidden states may be estimated based on probability distribution functions over a set of observed variables. These functions are called *output or emission probabilities*. Briefly, to define an HMM, the following probabilities have to be specified: (1) the initial probabilities that a state S_i, where $S_i \in \mathcal{S}^E$, may occur in the first place; (2) the probabilities that the system may transit from one state S to another S_{i+1} (transition matrix); and (3) the output probabilities.

HMMs may be used as *Hierarchical Activity Models* for activity recognition. First, an ontology of high-level composed activities is built, that is, the goals $\mathcal{G}$. One of the composed high-level activities g is chosen to be the recognition purpose of the system. Then all low-level primitive actions (where primitive actions refer to actions from the action set $\mathcal{A}$ in Section 2.1) that are related to the activity are recorded and are organized in different ensembles, that is, groups, so as to fulfill the recognition goal of

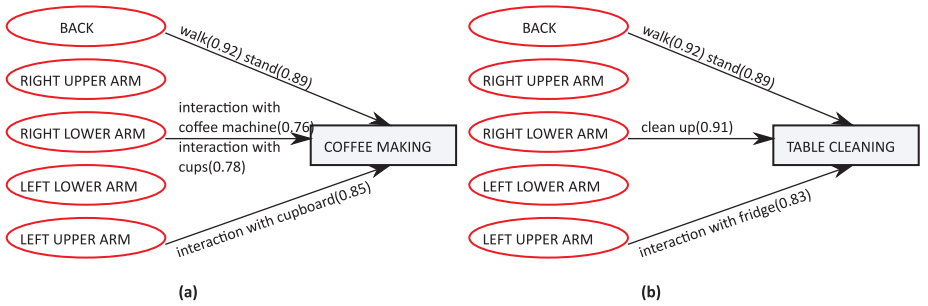

Fig. 5. Ensembles for goals: (a) Coffee Making and (b) Table Cleaning.

the system in different ways. Finally, the HMM is formed with the high-level activity modeled as the hidden state (i.e., the goal g) and all the related primitive actions as possible emissions of this state according to their probability [Hoelzl et al. 2012].

For example, Figure 5 illustrates the Hierarchical Activity Models for two activities: "Coffee Making" and "Table Cleaning" in a physical environment monitored by sensors. To recognize "Coffee Making," three sensors are involved: *back*, *right upper arm*, and *left upper arm*. Each sensor is monitoring the respective human body part and has been associated with a number of actions. For instance, sensor *back* is selected for inferring action "Walking" based on a measure, called degree of fulfillment, that reflects whether the system trusts the respective sensor to infer the action. The value of the sensor is 0.92 for "walking" and 0.89 for "standing." After actions are inferred, the main activity, that is, the goal g, can be inferred next according to the HMM that is constructed based on the plan corpus. Note that "Table Cleaning" can be detected by the same sensors with "Coffee Making," though associated with other actions: "clean up," "interaction with fridge," "walking," and "standing."

The use of HMMs requires deep understanding of the problem domain and usually requires very large training samples [Singer and Warmuth 1996].

Input Output Hidden Markov Models. A variation of HMMs considers additional context information, for example, the previous satisfied goal. This additional information modifies the state transition function and the observation probabilities when it is considered necessary. The Input Output Hidden Markov Models, or IOHMMs for short, manage to capture causalities in a similar way Bayesian networks do and are capable of updating their hidden states in a similar way HMMs do. IOHMMs are good models to be used in domains where there is abundant context information that can be exploited, for example, in computer games [Gold 2010].

Context information can be taken into consideration also in plain HMMs, by increasing the number of observation categories. However, this choice increases the training time and in addition causes a conceptual mixing of the known variables, that is, the variables showing whether a previous goal has or has not been already achieved with the hidden variable of the model, that is, the current goal.

Markov Logic Networks (MLNs). These Markov networks use Markov Logic (ML), a statistical-relational language that extends finite first-order logic (FOL) to a probabilistic setting. Specifically, they use a set of pairs (F_i, w_i), where F_i is an FOL formula, and $w_i \in R$ is a weight reflecting the significance of the constraint expressed by w_i, to calculate the conditional dependencies between pairs of nodes. The joint probability function is represented as the product of the potential functions. In the context of goal recognition, the MLNs represent the ambiguous causality between actions and goals in the dataset [Ha et al. 2012; Kautz 1991; Mott et al. 2006]. Kautz [1991] was the

first to introduce a formal theory of plan recognition in the context of Markov logic by suggesting a representation that may be transformed to an MLN by adding a binary node (binary variable) for each predicate and by considering the ground logic formulae as features that will determine the transitions into the network.

An example of logic formulae representing constraints in an interactive narrative system in a computer game environment [Ha et al. 2012] is the following set:

(1) $\forall t, a : action(t, a) \Rightarrow |\exists g : goal(t, g)| = 1$
(2) $\forall t, a : action(t, a) \Rightarrow goal(t, g)$
(3) $\forall t, a, s, g : action(t, a) \wedge state(t, g) \Rightarrow goal(t, g)$
(4) $\forall t, a, g : action(t - 1, a) \Rightarrow goal(t, g)$

The implicated parameters are (1) the player (i.e., actor) actions, such as moving to a particular location or opening a door; (2) the narrative states that represent the player's progress in solving the narrative scenario; and (3) the player's locations in the virtual environment. A narrative state is encoded as a vector of four environment variables, each one representing a milestone event within the narrative. Constraints in ML are divided into hard and soft. Hard constraints have to be always satisfied, while soft may be violated. The first formula represents a hard constraint, which defines that for each action at each time step t, the player has to pursue a single goal g. The second formula represents the prior probability distribution of the domain goals, while the rest of the formulae represent the goal g at time step t based on the values of the three parameters: time step t, action type a, and narrative state s. Each formula is assigned a weight that has been learned automatically from the plan corpus using a technique called Cutting Plane Inference (CPI) [Riedel 2012]. CPI limits the complexity of large-scale problems by focusing on a subset of constraints.

3.3.3. Bayesian Networks. Bayesian networks (BNs) have been widely used for goal recognition tasks because they manage to capture causality among actions and goals [Horvitz et al. 1998; Huber and Simpson 2003]. A BN is a directed acyclic graph in which nodes represent the constituent variables of the problem domain and edges the causal relationships or conditional dependencies between pairs of nodes, that is, between BN variables. The entire network can be understood as a representation of the joint probability distributions of all the random variables of its nodes. In a goal-aware system, the constituent variables may be observable or latent environment variables (unknown parameters or hypotheses), certain actions, or goals. The variables may represent observable quantities, latent variables, unknown parameters, or hypotheses. The strength of the connections between the variables is encoded in conditional probability tables. Independent variables are not connected.

For instance, consider a simple narrative virtual environment $E = \{p_1, \ldots p_m, l\}$, where each p_i ($1 \leq \mathrm{i} \leq \mathrm{m}$) represents an element that determines the plot of the story, and l represents the location of the user in the environment. Then, the BN variables could be variables representing sets of the plot elements (narrative states), the variable l indicating the location (e.g., D_l = {"locA", "locB", "locC"}), and a variable m that represents the user moves, that is, the user actions $\mathcal{A}$, within the virtual environment. Moreover, the links would capture the dependencies among the variables, for example, a move that results in the change of the narrative state [Mott et al. 2006]. Such a model in the context of an adventure game, for instance, can capture goals that involve different locations and plot elements such as being in location "locC," with the plot element p_1: *interaction with the story investigator* being true and the mystery considered as solved (plot element p_k is *solved*). To succeed, the actor may have performed several moves/actions such as collecting evidence and exploring different locations.

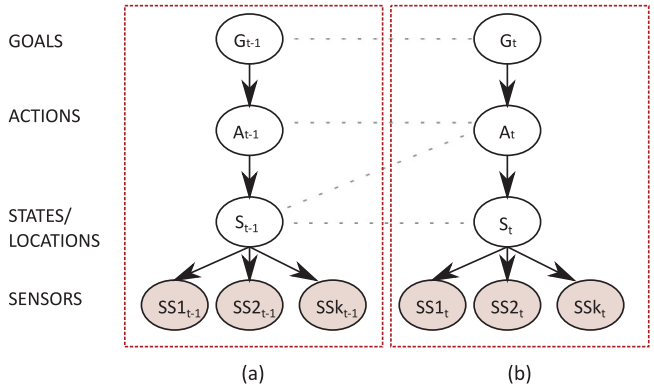

Fig. 6. An example of a Dynamic Bayesian Network.

Model Construction. For building the model, that is, learning the network, the conditional probabilities can be learned from real data (training corpus). The structure of the network can also be learned to some extent [Heckerman 1996]. However, most of the time, the structure is manually specified by domain experts. In case of an evolving dataset (continuous introduction of new evidence), the probabilities at each node may be recomputed by propagating the evidence through the edges.

Goal Inference. Bayesian networks can compute the conditional probabilities of the random variables of interest given a set of variables with known values, called evidence. Probabilistic inference using belief networks is NP-hard. To simplify the procedure, it is assumed that the variables are ordered for the independency probability assumption to be valid, that is, for a variable to be conditionally independent of its nondescendant variables given its parents. In this case, as in Markov networks, the Markov chain rule can be applied.

Inference may be efficiently performed using filtering to reduce the number of variables that are taken into consideration such as Rao-Blackwellised particle filtering. The latter is a combination of exact and stochastic inference; that is, sampling is used to reduce complexity but some of the variables (that are considered of greater importance) are excluded from the sampling procedure for higher accuracy [Doucet et al. 2000].

3.3.4. Bayesian Network Extensions. An extension of the Bayesian Network that also includes a temporal dimension is the Dynamic Bayesian Network (DBN). A DBN [Yin et al. 2008] is actually a sequence of Bayesian networks, each one modeling the dependencies among the variables in a specific time slice. Except for the causal links among the BN variables, there are also intraslice connections that represent temporal dependencies in consecutive time slices (refer to Figure 6). A plan in a DBN is a sequence of actions starting from an action node *act* with an incoming edge from a goal node *g*. In other words, the links from a node *g* to a node *act* point out a sequence of actions that implement the goal *g*.

By using DBNs, more complex models of sequential data, which are hopefully closer to reality, can be represented and learned. The price to be paid is the increased algorithmic and computational complexity. Parameters must remain the same across time slices so as to model sequences of unbounded length. The simplest way to do exact inference in a DBN is to divide the DBN into slices and then apply some inference algorithm to the resulting static Bayes net.

Figure 6 illustrates an example of a Dynamic Bayesian Network. The two dashed squares frame two hypothetical Bayesian networks (one for each time slot) that will

occur supposing that the dynamic network is unrolled. The dashed lines represent the connections between two consecutive time slots t-1 and t.

Works that use GPS data or data collected from sensors usually have different levels of inference. The lowest level corresponds to "raw" sensor data. In a DBN, the first level models the transitions at intersections and changes of modes of transportation (states), while the transitions at higher levels represent meaningful movements from one location to another (actions) [Patterson et al. 2003]. Moreover, a number of actions leading to certain locations constitute the plans toward the fulfillment of the corresponding goals.

3.4. Behavioral Theories

In environments such as those defined in social networking applications where users through their actions and interactions exhibit behaviors similar to those in the real world, the environment states can be determined by a number of environment variables $\{v_1, \ldots v_k\}$: the *motivations* or motivational factors (where $\{v_1, \ldots v_k\} = \mathcal{V}$). Intuitively, a motivation is a factor that drives someone in performing an action. Motivations preexist in the environment and they rule consciously or subconsciously user actions and inductively user goals.

Motivational factors and their interdependencies have been defined in theoretical *behavior models* by sociologists [Ajzen 1991; Fishbein and Ajzen 1975]. Models that reflect human behavior have also been developed independently by computer scientists [Chelmis and Prasanna 2012; De Choudhury et al. 2007; Perugini and Bagozzi 2001].

Two behavioral theories have been mostly used in computer science, the Theory of Reasoned Action (TRA) [Fishbein and Ajzen 1975] and the Theory of Planned Behavior (TPB) [Ajzen 1991]. TRA determines two main motivational factors (1) the *attitude* of a person *toward a behavior,* that is, his or her beliefs toward this behavior, and (2) the *subjective norm,* that is, the opinions of the persons that are important to the person under study (who will approve or disapprove the person in case he or she follows this behavior). TPB extends TRA by considering also whether the important persons to the user consider this behavior easy and trivial or important and worthwhile; this motivational factor is called *perceived behavioral control*.

Since the aforementioned theories are abstract and general, they can only be used to sketch the initial draft of the model, which is then enriched by motivational factors/variables that researchers consider important for a specific problem. In other words, these theories constitute the framework within which researchers express their hypotheses.

For example, Figure 7 illustrates a model for estimating the intention of a user to share knowledge within a business social network suggesting a number of motivational factors. In the graph representation, the nodes represent the environment variables, while their dependencies are determined by the stated hypotheses (i.e., each edge corresponds to a hypothesis). For instance, with collective/shared goals, organizational members tend to believe that other employees' self-interest will not affect them adversely and they all contribute their knowledge to help achieve their mutual goals.

Model Construction. To build a *theoretical model* for explaining user behavior, and by extension predicting the *user intention* to act toward a goal, the following steps are typically followed: (1) selection or formulation of a behavioral theory, (2) formulation of a set of assumptions (hypotheses) for each one of the factors that determine human behavior according to the selected theory (these hypotheses are the variables that define the suggested theoretical model), (3) conduction of a survey on real users to test these hypotheses, and (4) performance of statistical analysis to check the validity and reliability of the model.

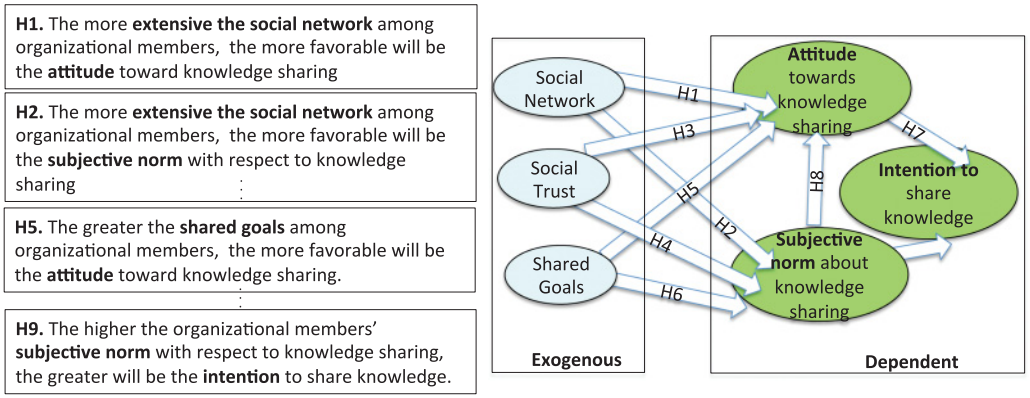

Fig. 7. Behavioral model for intention prediction.

The involved variables have to be determined based on the selected behavior model and then a set of causal assumptions or hypotheses that determine the dependencies among the variables are made. *One of the variables always expresses the user intention* regarding a certain behavior, that is, *the behavioral intention*.

The value estimation of some of the variables can be done *by regression relations*. These variables are called *dependent variables,* while the variables for which it is impossible to predict their values by other variables are called *exogenous*. For instance, in Figure 7, the extensiveness of the social network, the existence of social trust, and shared goals are exogenous; that is, they do not depend on other variables but they impact the dependent variables. The impact one variable has on another is determined by the stated hypotheses that form the behavior model.

Moreover, the in-degree of the nodes in the graph representation reveals whether a variable is exogenous or dependent. Exogenous variables are nodes with in-degree 0 because they are not pointed by other variables/nodes.

[User survey.] The stated hypotheses, which correspond to the variables of the model, are transformed into one or more questions usually of multiple choice that are answered by a representative sample of users. The results from the user study constitute the ground truth for the analysis. The performance of a user study is inherently problematic for very large datasets, such as data from Facebook (more than 1 billion of users), though. Consequently, it is feasible only for interest groups of users, such as university students or teenagers in a geographic region [Hsu and Lin 2008]. By offering free online services, such as simple game applications, companies manage to gather a significant amount of answers by questionnaires that either are part of a game or are required in order to use the service.

[Model accuracy.] In some cases, the consistency of the undirected model is controlled first, that is, the correctness of the selected variables/factors (confirmatory factoring analysis) [Chow and Chan 2008]. Then, the accuracy of the whole model is controlled including its dependencies (structural analysis) [Hsu and Lin 2008].

Goal Inference. The model is used to evaluate the commitment of the actor toward the behavior and the underlying goal. This is done by estimating the value of the variable *behavioral intention* based on the regression relations of the model. The known variables are the motivational factors that constitute the environment state (i.e., the input), while the latent variable is the behavioral intention (i.e., the output). Larger values of behavioral intention indicate more committed actors.

3.5. Based on Text Corpus

Text analysis does not offer a complete solution for goal modeling and recognition. However, information about goal achievement is available in multiple sources online with user-generated content, such as social networking sites, forums, blogs, and online guides. This large amount of data can be systematically analyzed to offer a wealth of information related to various aspects of goal achievement.

Thus, text analysis *can be used for gathering data regarding goals and goal achievement*, a task that is usually accomplished, as previously seen in this section, either directly by experts or through user studies, experimentation software, and annotation tasks. This way, experts' workload is reduced. Furthermore, since human effort cannot be compared in terms of time efficiency and cost to automatic text mining and NLP methods, the volume of gathered goal data can be significantly larger. *The data volume in combination with the diversity and the special characteristics of goal data makes goal modeling significantly challenging*.

Specifically, instead of (or in combination with) using expert knowledge, aspects such as goals [Castellanos et al. 2012; Smith and Lieberman 2010], motivations [Strohmaier et al. 2009], intentions [Castellanos et al. 2012], and actions [Strohmaier et al. 2009] *can be extracted from text*. Then, for the construction of the actual model and the inference process, an existing approach can be used, for example, plan libraries [Smith and Lieberman 2010] or rule-based annotators [Strohmaier et al. 2009; Louvigne et al. 2012]. Furthermore, social data can be mined to discover "recipes" of successful or failed implementations of various goals, to detect user sentiments during the life cycle of a goal, and to investigate the impact that goals have on social interactions.

Moreover, text analysis is important when users describe explicitly their goals using text [Carberry 1983], for example, "I want to eat Italian food in a restaurant nearby." Goals need to be inferred from the user input. For example, a keyword query such as "Michael Jackson, songs" can be transformed to either "I want to listen to M.J.'s songs" or "I want to download M.J.'s songs" based on analysis of verb phrases on web pages or result snippets [He 2010]. Overall, systems that use user-generated content can benefit from NLP and text mining techniques.

3.5.1. Text Corpus Examples. A plan library in the form of a *hierarchy of goals* is an example of a goal model built from text data. The data were taken from a social networking site designed to allow users to share their goals [Smith and Lieberman 2010]. The textual descriptions of goals can be analyzed, and connections between goals can be extracted by looking into common verb phrases. For example, when a similar verb phrase is found in the description of a goal and in a post explaining how the user accomplished another goal, the two goals can be connected accordingly.

Another example of the use of text analysis is the construction of a complete *plan library*, this time based on an existing taxonomy of human goals built by psychologists and sociologists [Strohmaier et al. 2009]. The *taxonomy* contains goals described by verb phrases such as "get married" and "become happy." The knowledge about how a goal can be implemented was derived from verb phrases that co-occur frequently in web pages with the textual description of the goal.

Another valuable source of data is Twitter. Twitter posts contain extracts related to goals. For instance, messages consisting of motivational messages such as "*Moi-lolita makes me want to learn some french #mangolanguages just to sing along to it*" and "*Getting ready for our trip in France, time to learn some french!*" contain information about learning goals [Louvigne et al. 2012]. Every such message consists of a set of textual features, for example, keywords such as "*because,*" "*so that,*" "*having,*" and a set of conceptual features. The latter features are motivational factors that reflect the difficulty or the engagement of the user to the respective goal.

To extract knowledge from such a corpus, *statistical models with rule induction* based on natural language processing patterns and other text mining techniques can be used. For example, if we consider a pattern *Verb+Infinitive* and a phrase derived from a post about Disneyland: "*would like to see the princesses,*" the phrase will be first characterized as goal (since it matches the pattern) and then will be disintegrated into the intention verb "*would like to*," the action "*see,*" and the object of the action "*princesses.*" Furthermore, a deeper understanding can be achieved by determining, for example, the intention (referred to in the specific work as the *level of intention*) of the user toward the goal. The intention inference can be made using classification methods on intention verbs; for example, "thinking of going" expresses weaker intention than "want to stay." The knowledge derived from this kind of analysis can be exploited from companies for providing better products and services to consumers and for personalized target marketing.

3.6. Discussion

The previous subsections have presented generic goal modeling and recognition methods, leaving out unnecessary application-specific details and keeping them under the common prism introduced in Section 2.1. There are, of course, methods that have been intentionally left out of this discussion, for example, works in the area of planning for stimulating human reactions, mainly because these works focus on nondata management issues, and hence are out of the scope of this study. Figure 2 provides a condensed overview of the alternative approaches, splitting goal modeling into its two components: goal data collection and model construction.

Table II summarizes the goal models focusing on these important features: the *main source* of goal data used for building the model, the *type* of the model, the *observation* (i.e., what is observed for finding the user goal), and the *model elements* (i.e., which goal-related concepts (refer to Section 2.1) are captured by the specific model).

Table III shows the methods used for building different goal model types. The main methods are (1) *expert analysis*, typically used for goal modeling based on complete records; (2) *statistical analysis,* typically used with corpus-based models; (3) *feature discovery and rule definition* in cases of taxonomy-based goal modeling; and (4) *behavior theory selection* in cases of modeling based on behavioral theories. Of course, these methods have been used in other cases too. For instance, expert analysis is also often used in corpus-based models, where experts may determine the structure of a model, for example, the connections among the variables of an HMM [Hoelzl et al. 2012]. We also often see different methods employed for constructing a model. For example, probabilities have been combined with expert analysis in plan libraries [Geib and Goldman 2001]. Moreover, machine learning, feature discovery, and expert analysis are often met in taxonomy-based approaches [Baeza-Yates et al. 2006; Herrera et al. 2010; Jansen et al. 2008; Lee et al. 2005; Li et al. 2006].

A number of observations can be made regarding the approaches and their uses.

Approaches that rely on complete records give a result if and only if there is a single candidate goal (or plan) matching the current observations. Such behavior is strict and may be sometimes considered inflexible. It may also increase the system response time.

Therefore, if goal exploitation is tightly integrated in the system workflow, the system may be blocked or delayed by the inference step. A solution to this problem is to use goal exploitation complementarily, as an add-on component. In this case, the system works properly when no knowledge about goals is available; when the pursued goal of the actor is successfully recognized, it can be exploited to improve the functionality of the system or the results returned to the actor. Moreover, even when additional time is required, if the inferred goal is adequately exploited, the effort required by the user can be also significantly limited, reducing the overall needed time.

Table II. Goal Model Type Descriptions

Main Source	Type	Short Description	Observations	Model Elements
Complete records	Plan Libraries	Set of all plans represented by action sequences, or action hierarchies with links showing post-pre-conditions; goals explicitly stated in each plan	Actions, or actions and env. variables	Actions, or actions and pre/postconditions
Complete records	Consistency graphs	Graph structure with nodes: all actions and goals and links representing restrictions on action performance pre/postconditions	Actions	Actions, goals
Complete records	Action centric	Transition graph (starting from initial state) with two types of levels: levels of observed actions and level of environment variable states; the links reflect action	States	Env. variables, actions
Taxonomies	Taxonomies & classifiers	Env. variables (features), functions capturing latent interdependencies among env. variables; a set of goal classes; no predefined set of goals	Actions	Env. variables, goal classes
Taxonomies	Taxonomies & annotators	Env. variables (features), patterns/rules involving env. variables (possibly with weights); a set of goal classes; no predefined set of goals	Actions	Env. variables, goal classes
Corpus	N-order Markov	A set of plans represented by action sequences, or a sequence of state transitions leading to a goal	Actions or states	Actions or states
Corpus	VOMs	Suffix tree for each goal; with nodes: actions that belong to plans for the goal; and links connect actions to get plans of different lengths	Actions or states	Actions or states
Corpus	HMMs	Graph structures; with nodes: observed variables; links expressing emission probabilities; state transition matrices/functions; goals explicitly stated	Observed variables	Observed variables, output variables
Corpus	IHMMs	Graph structures; with nodes: observed variables; links expressing emission probabilities; state transition matrices/functions; goals explicitly stated	Context (e.g., previous pursued goal), observed variables	Env. states, env. variables, goals
Corpus	MLNs	Graph structure; with nodes: formula predicates; and links: expressing a set of weighted logic formulae involving actions and goals	Actions	Actions and goals
Corpus	Bayes Networks	Graph structure; with nodes: actions, goals, and states; and links expressing causality among them	Actions, states, goals	Actions, states, goals
Corpus	DBNs	Graph structure with levels (for different time slots); nodes: actions, goals, and states; and links expressing causality among them	Actions, states, goals	Actions, states, goals
Behavior Theory	Behavior models	Graph structure; with nodes: env. variables (i.e., motivational factors) plus intention variable; with interdependency links expressing hypotheses of behavior models	Env. variables	Env. variables, intention variables

Table III. Methods Used During Goal Modeling

Models	Machine Learning	Text Mining	Behavior Theory Selection	Hypotheses Formu lation	Stati stical Analysis	Feature/ Rule Definition	Expert Analysis	User Studies Anno tations
Plan libraries /graphs		Smith and Lieberman [2010]	Schmidt et al. [1978]	Kautz [1991]	Geib and Goldman [2001]		Sadri [2012], Carberry [2001], Schmidt et al. [1978], Kautz [1991], Geib and Goldman [2001], and Hong [2000]	
Action-centric					Sun and Yin [2007]		Sun and Yin [2007] and Ramírez and Geffner [2011]	
Taxonomies	Baeza-Yates et al. [2006], Herrera et al. [2010], He [2010], Jansen et al. [2008], Lee et al. [2005], and Li et al. [2006]	He [2010] and Strohmaier et al. [2009]			Hassan et al. [2010]	Baeza-Yates et al. [2006], Herrera et al. [2010], He [2010], Jansen et al. [2008], Lee et al. [2005], and Li et al. [2006]	Baeza-Yates et al. [2006], Broder [2002], Herrera et al. [2010], Jansen et al. [2008], Lee et al. [2005], and Li et al. [2006]	Broder 2002], Kang and Kim [2003], Lee et al. [2005], and Rose and Levinson [2004]
N-order Markov	Sadikov et al. [2010]				Blaylock and Allen [2003 and Sadikov et al. [2010]			
VOMs	Armentano and Amandi [2009]				Armentano and Amandi [2009]		Armentano and Amandi [2009]	

(Continued)

Table III. Continued

(IO) HMMs					Hoelzl et al. [2012] and Gold [2010]		Hoelzl et al. [2012] and Gold [2010]	
MLNs					Ha et al. [2012], Kautz [1991], and Mott et al. [2006]		Ha et al. [2012], Kautz [1991], and Mott et al. [2006]	
BNs					Horvitz et al. [1998], Huber and Simpson [2003], and Patterson et al. [2003]		Horvitz et al. [1998], Huber and Simpson [2003], and Patterson et al. [2003]	
Behavioral Models			Bagozzi and Dholakia [2002], Chow and Chan [2008], Cheung and Lee [2010], Hsu and Lin [2008], and Parra-Lpez et al. [2011]	Chelmis and Prasanna [2012], De Choudhury et al. [2007], and Perugini and Bagozzi [2001]				Bagozzi and Dholakia [2002], Cheung and Lee [2010], Chow and Chan [2008], Hsu and Lin [2008], and Parra-Lpez et al. [2011]

However, there exist scenarios where the system should only take into account goals that are certain so as not to confuse the user or deteriorate the system operation. The latter is especially true in critical domains such as security. For instance, plan libraries have been used in the domain of web security [Geib and Goldman 2001]. However, approaches that perform consistency checking are in general not recommended in cases of adversarial recognition since hostile actors are not expected to follow ordinary plans. For these cases, design tools can be used to minimize the maximal number of observations before a goal is recognized, a task known as goal recognition design [Keren et al. 2016a].

Whether the knowledge of goals will benefit a system or not depends on the correctness and completeness of the data used for constructing the model. The creation of the complete records is generally a hard task, first due to the time and effort required by the domain experts, and second due to the existence of unknown plans. These techniques are more appropriate for problems where the focus is on a small number of goals.

Choosing the right model based on complete records for an application scenario depends on the construction and consistency checking mechanisms used. In plan libraries, allowed actions are combined to form all the possible valid plans for a set of goals of interest. In action-centric approaches, experts follow a different approach for gathering the knowledge. They predefine goals of interest but not actions. The transition graph in action-centric representations is formed while the environment is being perceived. The final plan synthesis is performed during the goal inference, making action-centric models appropriate for cases where gathering exhaustively all the plans is considered very costly or is not possible. Automatic consistency checking mechanisms are used while new observations become available to remove inaccuracies and revised judgments made earlier [Yin et al. 2007].

On the other hand, in plan libraries, when a plan does not contain an action that has been observed, it becomes inconsistent no matter if the previous observed actions agree with the plan. One could say that plan libraries include the most common plans that actors may follow in order to achieve some goal. In a closed world (closed-world assumption), this collection is considered complete. This observation is the reason for both the advantages (certainty and clear knowledge) and disadvantages (static, no new plans) of this model. On the flip side, in consistency graphs, there is no predefined number of plans; every action is connected with every goal unless an inconsistency is caused by this connection. This way, a set of observations that have not been met before can be associated to a goal. But at the same time, the more actions and goals exist in a consistency graph, the more expensive and complicated the inconsistency checking becomes. Furthermore, it becomes challenging to do plan recognition, since the plans are not encoded in the model. Consistency graphs are in general a better fit for providing causality relationships between actions and goals. *Approaches that rely on a corpus* deal with the problem of gathering goal knowledge by introducing probabilities into the goal models. Action sequences in Markov chains, for instance, could be seen as an incomplete plan library. Since not all the plans are recorded, the latest observations are used as evidence to predict the goal, and by extension the plan, that an actor is following.

There is a high-level connection between hierarchy plan libraries and Variable-Order Markov Models (VOMs). They both use a graph representation to encode alternative sequences of actions. In hierarchy plan libraries, this graph representation can reveal whether a goal is pursued by checking whether it is consistent with the environment; in VOMs, it is used to compute the most probable goal.

VOMs and n-order Markov models (chains) handle the actions as black boxes. In order to capture directly the environment variables or state transitions that occur

while a goal is being fulfilled, models such as HMMs or Bayesian Networks should be employed.

Bayesian networks capture causalities among actions and goals, since goals are a part of the network. Dynamic Bayes Networks allow observing goal and plan evolution over time. HMMs focus on the environment variables and allow the inference of unobserved environment variables from observed ones. In the case of HMMs, the goal is identified by defining the hidden state of the environment. This is why they have been extensively used in cases where sensors are involved (e.g., activity recognition [Hoelzl et al. 2012]), but they can also be a choice for any environment with variables with latent connections that can reveal the values of other variables.

Latent relationships among environment variables are also captured by *statistical models built on behavioral theories*. These approaches explain and predict user goals (and behavior in general) considering the user as part of a community. The methodology (i.e., the formulation of hypotheses and the performance of statistical analysis to check their validity and reliability) could be exploited for exploring connections of environment variables in different domains as well.

Approaches based on taxonomies that capture latent goals embrace uncertainty, specifically in the stage of the inference of the goal class to which the actions belong. To introduce goals in the core algorithmic procedures of a system based on a taxonomy, the most important task is the discovery of intrinsic features and properties within the environment that can be used as environment variables. This kind of approach can be more naturally used by applications such as online recommendations, social media applications, and web retrieval systems, where the environment consists of nonmonolithic objects. Such objects can be analyzed and represented in the context of an environment in a goal-aware system. For instance, web pages, forum posts, or products in an online store can be analyzed through user studies and experimentation to discover the right features that will be used for modeling environment variables and that can be associated to different (soft) goals. For web pages, we have seen in Section 3.2.1 which features have been used (and how) to form an environment in the context of which user goals can be inferred and exploited for effectively performing the retrieval of the web pages by satisfying the latent user goal.

4. GOAL EXPLOITATION

Given the goal model that captures the information about the operationalization of goals within a system and the inferred actor's goal, the system may exploit this knowledge to the benefit of the actor (in nonadversarial cases) or itself. In the literature, the exploitation of goals has been closely linked to the application scenario. Since the purpose of our study is to bring light to the challenges and the practical value of a goal-aware system, we categorize goal exploitation cases according to the *system behavior* once the user goal is known.

Systems using goal models with full plans, for example, plan libraries or n-order Markov models, can select which plan will be used to fulfill the goal based on a number of criteria, for example, computational cost, plan length, and so forth. Then, either the system executes the plan automatically or it guides the user through further interaction toward the fulfillment of the goal.

Systems that contain goal models where plans or goals are not fully determined (e.g., HMMs or models based on taxonomies) do not replicate a certain plan. The goal can be input to one or more algorithms that support the main functionality of the system. For instance, in web search engines, the inferred goal can be used by the web retrieval algorithms to reorder or filter the web sources in the result list.

Table IV. Goal Exploitation

Exploitation Cases		Indicative References
Exploitation Through Dynamic Environment Changes		
Acting in Anticipation of the Actor's Actions	By automatic action execution	Armentano and Amandi [2009], Geib and Goldman [2001], Lieberman [2009], and Lesh [1998]
	By state transitions	Charniak and Goldman [2013], Ha et al. [2012], Meehan [1981], Riedl [2004], Schank [1995], Han and Pereira [2010], and Roy et al. [2007]
Promoting/ Facilitating Actions	By interface adaptation	Dragunov et al. [2005], Armentano and Amandi [2009], Lesh [1998], and Lieberman 2009]
	By exposing actors to other actors' plans	Louvigne et al. [2012]
Exploitation Through System Response		
Adjusting the system response	By posteriori adjustment of the initial response of the system	Broder [2002], Lu et al. [2006], Li et al. [2006], and Herrera et al. [2010]
	By returning information about actions	Carberry [1983], Blaylock and Allen [2005], Crook and Lemon [2010], and Maragoudakis et al. [2007]
Side Services		Castellanos et al. [2012], Ajzen [1991], Chelmis and Prasanna [2012], De Choudhury et al. [2007], and Perugini and Bagozzi [2001]
Object Modeling		Carpineto et al. [2009] and Sadikov et al. [2010]

Overall, we see that goal exploitation can take two forms: (1) *exploitation through dynamic environment changes*, where the system provokes changes in the environment that lead directly or indirectly to goal fulfillment while the actor is interacting with the system, and (2) *exploitation through system response*, where the system responds to the actor request(s) using algorithms that embrace goals into their core functionality (i.e., algorithms implementing the tasks intended by the system that respond to actor requests considering the inferred goal). Table IV offers a hierarchical organization of the different cases.

4.1. Exploitation Through Dynamic Environment Changes

We further consider two subcategories of goal exploitation in this category: (1) acting in anticipation of the actor's actions and (2) promoting/facilitating actions.

4.1.1. Acting in Anticipation of the Actor's Actions. In this case, the system automatically performs actions (instead of the actor) or it changes the environment state (before the actor performs the actions he or she has in mind).

By Automatic Action Execution. In principle, every type of application that allows interaction with the user can leverage user goals. A system that "understands" the objects of interest to an actor and monitors the user operations and interactions in the environment of the system can automatically change its operation and behavior according to the actor's goal. For instance, it can *take actions on its own by invoking the commands provided by the system interface* (i.e., the commands that the user can perform through the interface) toward the fulfillment of the user goal. Such goal-aware interfaces are known as *intelligent interfaces* and have been used in applications like web browsers, text editors, and search engines [Armentano and Amandi 2009; Lesh et al. 1999; Lieberman 2009]. Another example is computer or web security

systems (e.g., Geib and Goldman [2001]), where the system can automatically change its operation to prevent user attacks.

In a goal-aware text editor, for instance, by observing a sequence of menu selections and clicks (i.e., actions), the editor may infer that the user aims to disable autocorrection. To save the user from browsing the various menu options, the editor can directly fulfill the goal (i.e., the deactivation of auto-correction). To avoid misinterpretations, a confirmation question may be posed to the actor before the system starts performing the actions that will fulfill the goal.

By State Transitions. Apart from goal-aware systems that act proactively in anticipation of the user actions to fulfill user goals, there exist systems that trigger *environment state transitions* when the goal is inferred in order to offer a different user experience, for example, a more interesting, amusing, or unexpected experience.

A well-known example in this category is the *interactive virtual environments or narrative managers*. Interactive virtual environments have been suggested for education and training environments [Mott et al. 2006; Louvigne et al. 2012], as well as for entertainment (game playing) [Gold 2010; Ha et al. 2012; Kabanza et al. 2010]. In *education*, interactive virtual environments engage students in learning procedures that serve educational purposes being at the same time amusing and pleasant. For example, in a virtual environment for microbiology, the laboratory changes according to the learner's goal, while he or she is trying to resolve a science mystery [Mott et al. 2006].

In *game playing*, narrative managers, depending on the player's goal, change the environment states to introduce unexpected events, for example, unlock new powers or traps, or to prevent the player from repeating the same strategies, that is, *plans* [Ha et al. 2012]. Such goal exploitation mechanisms *differentiate the player's experience periodically* and enhance the player's loyalty to the game. Games often start with a trial session, during which the game directs players to follow specific goals in order to familiarize themselves with the environment. In this way, the game can keep track of the actions players follow for achieving their goals and can enhance an existing goal model (that expresses *the average player*) or create a model *for the specific player*. The latter can offer a more personalized experience.

Another example of systems that perform state transitions based on the inferred goals of the actors is that of assistance technologies or intelligent homes [Han and Pereira 2010; Roy et al. 2007]. The practical value of these systems is especially high in social groups that deal with problems such as mobility difficulties or vision or memory problems causing them difficulties in performing everyday tasks. The environment in these cases consists of variables that indicate mainly sensor values: locations, moves of certain body parts, sounds, and so forth. Furthermore, medical or other personal data may be captured. When the actor's goal is inferred, certain environment variable states change before the actor performs any further actions. For instance, the room temperature, the volume of the television, or the location of an object may change. The state transitions may directly cause the goal fulfillment, or some additional actions may be expected by the actor. Easiness and effectiveness are the desired qualities for such systems, while personalization elements may be desirable.

4.1.2. Promoting/Facilitating Actions. In this case of goal exploitation, the system somehow suggests actions to the actor by making them more "obvious" and easily accessible. It is, however, up to the actor to perform them.

By Interface Adaptations. To ensure the fulfillment of a goal, a system may facilitate or guide the actor to perform certain actions through various adaptations. These adaptations include the addition of graphics or animations, the use of vocal input/output means, or communication via other sensor channels [Lieberman 2009]. Intelligent

interfaces additionally to the automatic execution of actions may perform interface adaptations as well [Armentano and Amandi 2009; Lesh. 1998; Lieberman 2009]. Other features of adaptation or personalization are pointing out alternative paths and reordering or highlighting interface elements related to the user goals. Another important issue that the system needs to deal with is the changes in actor goals and to trigger the performance of additional actions whenever they are needed.

An example of such a system is TaskTracer [Dragunov et al. 2005], a research tool that consists of a number of intelligent interfaces that were designed to be on top of every desktop activity regarding Microsoft Office applications, Visual Studio, and Internet Explorer in Windows XP.

By Exposing Actors to Other Actors' Plans. Another way to exploit goals is by exposing the users to information about the goals of other actors. Specifically, in *educational interactive virtual environments*, exposing learners to information regarding the operationalization of goals of their peers was found very successful [Louvigne et al. 2012]. There is a theory behind this, called *observational goal setting theory*, that suggests that information about the goals of others may help the current learner (actor) to stay committed to his or her goals.

4.2. Exploitation Through System Responses

We further consider these subcategories of goal exploitation in this category: (1) adjusting the system response, (2) side services, and (3) object modeling.

4.2.1. Adjusting the System Response. In contrast to goal exploitation through dynamic changes in the environment, there are cases in which the system does not respond unless the actor makes an explicit request. Practically, the actor's goal is fulfilled through a plan containing actions that become available to the actor via the response of the system. The final selection of the actions is made by the actor though. The actions that the actor performs after the system's response may optionally be tracked and used to adapt the response. *Web Information Retrieval* is the most well-studied application domain.

Although the use of search engines has been significantly expanded in the last decades, users may not know exactly what they are searching for, or they may not know how to actually express it in a query language even if the query is expected in a form as simple as a set of keywords. Since users aim at discovering information (i.e., resources) and they keep browsing the results returned from a search, one can consider as an *action* the fact that the user poses a query and sees the results (the change of the information that a user has seen is reflected in the state of the environment) and this task stops when the user finds what he or she needs. If the user is not satisfied with any of the results, he or she has to pose another query and continue the same process. Thus, a *goal* can be modeled also as a set of environment variable states, and the clicking on the results returned for one or more queries the user has posed can be seen as the *actions*.

Goal-aware information retrieval allows users to cope with information overload and reach the information of interest faster. Clearly, understanding what the user intends to do (refer to Figure 8(b)) with the data and *promoting results* that help fulfill that goal can significantly improve the quality of the results and increase user satisfaction. This has led *search engine* providers into the study of ways to understand what the user had in mind. Over the last two decades, there have been *efforts to make search engines able to recognize the goal of the user*, that is, what the user wants to do with the retrieved information [Broder 2002; Herrera et al. 2010], and *act accordingly*. For instance, consider three users that pose the following queries: (1) "*Michael Jackson songs,*" (2) "*Michael Jackson,*" and (3) "*Michael Jackson site,*" respectively (refer to Figure 8(a)).

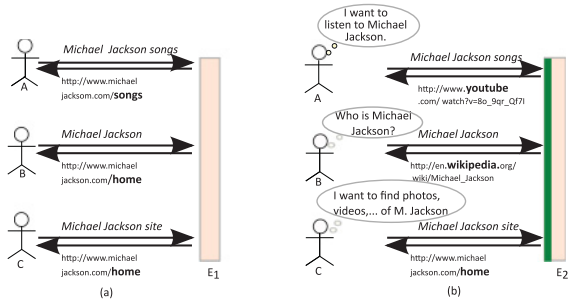

Fig. 8. Answering web queries (a) using only IR techniques or (b) in combination with goal recognition and exploitation.

For the first user, the search engine returns the link www.michaeljackson.com/songs containing the list of all the songs the artist has sung. For the second and third user, it returns the link to the artist's web page. At first sight, all the results seem to be satisfactory to the users. However, the first user does not actually want to see the list of songs, but wants to listen to them. Thus, a preferable answer would have been a link to the artist's songs on YouTube. The second user was actually interested in learning about Michael Jackson's life; thus, a link to his Wikipedia page would have been a more useful resource than his home page. Finally, the third user is interested in finding photos, videos, and filmography of the actor. For that user, the actor's website is indeed the best resource.

The goal taxonomies of queries presented in Section 3 have been plugged into existing web search engines to adapt the retrieved results to the goal behind the current user query. For example, retrieval algorithms that consider goals ensure that queries matching the class of *informational* goals should be answered by diverse web resources that cover the query from different perspectives, contain complementary and controversial information, and offer various levels of comprehension. All *these requirements are guaranteed in the respective goals by conditions on environment variables* such as resource diversity, size of resource, and so forth.

Queries that match the class of *navigational* goals should be answered by specific web resources that are characterized by perfectness, uniqueness, and authority [Lu et al. 2006]. In this case, users may be certain of the existence of the site because they have accessed it before, they have been informed about its existence by an external resource, or they assume it exists. For instance, a user may assume that there is a website for a scientific laboratory even if he or she has no information about it. In order for a goal-aware engine to detect authoritative sites for navigational queries, it exploits the *inferred goal that contains conditions on environment variables (features)* describing click streams from query logs or on features regarding the URL of a resource, such as the length of the longest substring of the query that can be matched to the URL [Lu et al. 2006].

Another example is the answering of queries given one query that expresses a *transactional* goal. In this case, goal-aware web search engines answer the queries using a specific fraction of the web collection, since according to studies, transactions can be performed only in a small number of websites that can be possibly distinguished from common pages. Indeed, there have been efforts to spot "transactional" pages and create a collection of web resources that will be used only for transactional queries. An example is an annotator based on regular expressions and gazetteer lookups that was built for queries related to two types of user activities on the web: software downloads and entry forms [Li et al. 2006].

By Returning Information About Actions. There also exist systems where after the goal is inferred, the system tries to *return information about the plan, that is, about the related actions and action preconditions needed for fulfilling the goal*. This is met in dialogue systems for instance, which are computer systems intended to converse with a human with a coherent structure. They support a broad range of applications in enterprises, education, government, healthcare, and entertainment, such as customer care and helpdesk services, technical support, and informational services about news, entertainment topics, the stock market, or any other type of information stored in a knowledge base [Carberry 1983; Crook and Lemon 2010; Maragoudakis et al. 2007]. Respectively, in Linux systems dialogue systems have been considered for goals that can be described in a high level as tasks such as sorting the files and subfolders in a folder, or renaming the files based on a pattern. Such goals are operationalized by command line commands [Blaylock and Allen 2005]. The dialogue systems first infer lower-level goals and then gradually build a *complete plan* as the dialogue progresses [Carberry 1983]. Thus, goal inference is performed hierarchically.

4.2.2. Side Services. Goals can also be exploited *by side services for marketing purposes*, for example, targeted offers, market analysis, and so forth. Online user-generated content offers great opportunities for extracting and encoding knowledge about human goals. For instance, goals expressed in phrases such as *I want to visit France* in Twitter or travel websites constitute valuable information for marketing in the travel and leisure industry [Castellanos et al. 2012].

Several personalized services benefit from the prediction of user behavior in social platforms [Ajzen 1991; Chelmis and Prasanna 2012; De Choudhury et al. 2007; Perugini and Bagozzi 2001]. For example, in online social virtual games, one can meet motivations capturing notions such as "entertaining" or "being challenged by others." A user may be interested in a recommendation of a game that offers entertaining missions (i.e., by trying to satisfy related goals) motivated by a tendency to entertain himself or herself, while another user may be willing to buy a game to pursue missions of great difficulty motivated by the fact that he or she is challenged by other players.

There are currently no algorithms capturing goal knowledge that manage to perform in an automatic way, that is, without the interference of a marketing team, tasks such as recommendations in a way that will help users to fulfill their goals, making it a challenging open direction.

4.2.3. Object Modeling. When an object (e.g., a query or a web page) is involved in the fulfillment of one or more goals, either in an action, in an environment state, or in a goal, the information about operationalization can be used for the modeling of the object. Two clustering algorithms have exploited such information for object representation. The first is designed for *clustering the results retrieved by a web search engine for a certain query*. The web pages in the result set are clustered into hierarchies that reflect the different *aspects of the query* [Carpineto et al. 2009] to allow the user to access the results that are only related to a specific goal. Such goal-aware techniques are especially useful in *mobile search* because mobile users are typically not willing to pose more than one query per session nor to scroll through long lists of results, and may be overwhelmed by a large volume of unrelated data. The other algorithm is designed for query clustering in order to suggest to the users those *related* to a query at hand [Sadikov et al. 2010]. A corpus-based goal modeling approach was followed. The used corpus consists of anonymized logs from the Google search engine containing user queries and clicks on pages from the respective result lists within a user session. Query refinements are considered to be the actions, and the documents the potential user goals. A graph for each query is created. The nodes representing the goals are

absorbing nodes; that is, once the user visits a document, it is assumed that he or she stops having the same searching goal in mind unless he or she further refines the same query again. Subsequently, the goal model is constructed, a *Markov chain* model with a transition matrix that reflects the probability with which a user may end up to each goal within a number of steps (actions). Given the model, every query can be transformed into a discrete probability distribution. Finally, a clustering algorithm is performed on these representations capturing the knowledge about goals and intentions of the "average" user.

4.3. Discussion

In all the aforementioned cases of goal exploitation, the goal of the current actor is recognized and exploited to give certain qualities to the usage of the system such as personalization, effectiveness, serendipity, and so forth. Effectiveness can have several interpretations; herein, it is used to show that the actor's goal is fulfilled through the system usage. Despite effectiveness, which is a quality desired by all goal-aware systems, there exist systems that consider goals to ensure that the actor will accomplish his or her goal easily and as soon as possible, for example, intelligent interfaces and dialogue systems. Other systems, on the other hand, embrace goals to make the usage of the system (until the goal is satisfied) more interesting, with surprising and pleasant elements or personalized features. These qualities, as we have seen, are very important for applications such as game playing and target marketing, but at the same time they have been proved beneficial for query answering, for instance. In fact, web search engines want to accomplish the right balance between the two; they want to pleasantly surprise the user with their responses and direct him or her in an efficient fulfillment of goals. Another advantage of goal-aware systems is that the included goal models capture data about the operationalization of goals that allows the discovery of knowledge during goal exploitation.

The different needs and desired qualities of certain applications can be met by diverse goal model types. Goal-aware systems have used different types of models for the same application domain. For instance, Interactive Narrative Managers have employed Markov Logic Networks [Baikadi et al. 2012; Ha et al. 2012; Mott et al. 2006], IOHMMS [Gold 2010], and Bayes Networks and N-order Markov Models [Mott et al. 2006]. Similarly, dialogue systems have employed Markov Models [Maragoudakis et al. 2007] and Bayes Networks [Raux and Ma 2011].

5. RESEARCH DIRECTIONS AND CONCLUDING REMARKS

Goals can provide a formal foundation for going beyond the traditional modeling and analysis of data, and can help build better systems and services for the end-user. Goal-aware approaches will and should gain more ground in the scientific community and industry due to the reinforcement of user engagement and sense of satisfaction, the direct and indirect economic implications (i.e., advertisements, e-commerce, and marketing campaigns), and the usefulness of the derived knowledge in a variety of applications. With a clear picture of what is meant by a goal-aware system, the different goal models from the literature, the ways they are constructed, how they are used for goal recognition, and how they are exploited in certain application scenarios, we can discuss additional situations in which goals can play a significant role and identify possible research directions and challenges. Table VI summarizes the potential benefits we have identified on the usage of goals in various systems.

Big Data and Query Processing. As the amount of data outgrows the capabilities of query processing technology and the number of emerging applications, from social networks to scientific experiments, is growing fast, there is a clear need for efficient

Table V. Qualities Added to the Usage of Certain Applications by Considering Goals

Qualities	Intelligent Interfaces	Dialogue Systems	Interactive Systems for Education, Game Playing	Personalized Target Marketing	Web Search Engines	Assistance Technologies
Velocity	✓	✓	×	×	✓	✓
Easiness	✓	✓	×	×	✓	✓
Interestingness	✓	×	✓	✓	×	×
Personalization	✓	×	✓	✓	✓	✓
Serendipity	✓	×	✓	✓	✓	×
Extra knowledge	×	×	✓	✓	✓	×
Effectiveness	✓	✓	✓	✓	✓	✓

Table VI. Qualities That Can Be Added to the Usage of Certain Applications *If Goals Are Considered*

Qualities	Big Data and Query Processing	Interactive Data Exploration	Recommendations	Retrieval and Data Mining
Velocity	✓	✓	×	×
Easiness	✓	✓	×	×
Interestingness	×	×	✓	✓
Personalization	×	✓	✓	✓
Serendipity	×	×	✓	✓
Extra knowledge	×	✓	✓	✓
Effectiveness	✓	✓	✓	✓

big data query processing to enable the evolution of businesses and sciences to the new era of data deluge. In this context, introducing goal-aware data and query processing methods can provide a whole new perspective into building database systems that are tailored for big data and the goals of the users accessing these data by providing features such as adaptive indexing, adaptive loading, and sampling-based query processing and goal-aware query processing and optimization methods. These directions focus on reconsidering fundamental assumptions and on designing next-generation database architectures for the big data era. A system, by considering, for instance, sets or sequences of queries that operationalize certain goals, can focus only on the part of the data that will serve the user's purpose and not every possible piece of data that satisfies the query conditions. The query conditions, in this context, would constitute the conditions on the environment variables of the respective goals.

The number of goals that are typically to be processed are not proportional to the volume of the data. Therefore, even goal models that are meant for a smaller number of goals (e.g., those based on consistency checking) may be easily employed. What is challenging in this case is to build and maintain adequate structures and design algorithms that will enable the answering of queries in a timely and effective fashion.

Interactive Data Exploration. Interactive data exploration is an emerging form of data-intensive analytics in which users ask questions over a dataset to make sense of the data, identify interesting patterns and relationships, and bring aspects of interest into focus for further analysis. Interactive data exploration is fundamentally a multistep, nonlinear process. Data exploration requires users to possibly ask a large number of queries as they try to navigate through large datasets. Incorporating notions of goals seems like a natural step and requirement for reducing the human workload and serving better results faster. The operationalization of the goals considered by the system may contain whole queries, and/or interactions with tuples or columns, or clicking on single fields and values. However, since users often have underspecified and

shifting end-goals, goal modeling and recognition are very challenging. Hierarchical goal inference could be more appropriate in this case to capture the refinements of user goals while interacting with the system the same way they often do in dialogue systems. Systems intended for interactive data exploration can exploit knowledge about the operationalization of goals of the average user, as well as enhance goal models by information regarding the current user so as to give personalized results when needed. However, the main objective remains always to facilitate data exploration in terms of time and effort.

Recommender Systems. The aim of a recommender system is to estimate the utility of a set of objects belonging to a given domain, starting from the information available about users and objects. Algorithms such as collaborative filtering have traditionally assumed that the object utility is a function of user preferences (expressed as ratings, likes, etc.) and in some cases of the user context. Adding goals into the equation can modify the whole perception of what a recommender system is. By considering goal operationalizations that involve interactions with the objects, a goal model allows the system to identify the most appropriate objects for the user, that is, the objects that serve the goal(s) he or she is currently pursuing. Toward this direction, adaptive e-learning systems, for instance, can automatically generate personalized learning experiences starting from a learner's profile and a set of target learning goals. Moreover, a new need rises for algorithms that can model and recognize user goals by observing the user actions performed so far, that is, the interactions with the objects.

Goal-aware recommenders can offer a more surprising and richer experience for the user since the connections of objects due to goals are not obvious nor known by all users, especially when goal models capture data from diverse plans.

Furthermore, there is a source of data valuable for marketing and recommendations (as was highlighted in Section 3.5) that makes the problem of goal modeling and recognition more challenging due to the volume and the variety of the sources; this source is social data, that is, data from social networking sites and applications.

Another interesting direction where social data can be used is for analyzing the content that a specific user shares online to extract user goals. Users with similar goals could get connected to discuss common problems and get motivated. Moreover, special services and features can be implemented for groups of users with common goals.

Retrieval and Data Mining Tasks. A direction touching upon issues related to recommendations but from a different perspective is that of identifying the proximity or relatedness among objects for retrieving or mining objects in order to respond to user actions and requests. By considering that the objects are implicated in the fulfillment of a number of goals and adjusting accordingly the modeling (representation) of the objects as well as the respective algorithms, such tasks can become more effective.

Toward this direction, web queries have been modeled and clustered with the help of goal models where goals represent the target web pages. The objective of this work was to retrieve related queries [Sadikov et al. 2010] (refer to Section 4.2.3). In web retrieval, on the other hand, where the objects of the analysis are the web pages, goal taxonomies in an environment consisting of intrinsic web page features managed to increase user satisfaction in web query answering [Broder 2002; Herrera et al. 2010] (refer to Section 3.2.1).

These works show us two different alternatives that can be explored for diverse object types and applications (besides web querying). Specifically, they can be explored in systems that respond to actions such as clicking on objects, publishing multimedia items or texts, and so forth either (1) to build a goal model that involves actions and

goals or (b) to discover intrinsic features (i.e., variables) that are adequate for defining and recognizing the implied goals.

Corpuses of click logs marked with goals, human annotations of features, and experimentally tested hypotheses regarding variables are sources of goal data that seem more promising.

ACKNOWLEDGMENTS

Work partially supported by the ERC grant Lucretius 267856 and the Keystone Cost Action IC1302. A special thanks to Avigdor Gal and Sarah Keren for their valuable comments and recommendations.

REFERENCES

Icek Ajzen. 1991. The theory of planned behavior. *Organizational Behavior and Human Decision Processes* 50, 2 (1991), 179–211.

Han The Anh and Luis Moniz Pereira. 2013. State-of-the-art of intention recognition and its use in decision making. *AI Communications* 26, 2 (2013), 237–246.

Marcelo Gabriel Armentano and Anala Amandi. 2007. Plan recognition for interface agents. *Artificial Intelligence Review* 28, 2 (2007), 131–162.

Marcelo Armentano and Analía Amandi. 2009. Goal recognition with variable-order Markov models. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*. 1635–1640.

David John Atkinson and Micah Henry Clark. 2013. Autonomous agents and human interpersonal trust: Can we engineer a human-machine social interface for trust? In *AAAI Spring Symposium: Trust and Autonomous Systems*.

James T. Austin and Jeffrey B. Vancouver. 1996. Goal constructs in psychology: Structure, process, and content. *Psychological Bulletin* 120, 3 (1996), 338.

Ricardo Baeza-Yates, Liliana Calderan-Benavides, and Cristina Gonzalez-Caro. 2006. The intention behind web queries. In *String Processing and Information Retrieval*. Lecture Notes in Computer Science, Vol. 4209. Springer, Berlin, 98–109.

Richard P. Bagozzi and Utpal M. Dholakia. 2002. Intentional social action in virtual communities. *Journal of Interactive Marketing* 16, 2 (2002), 2–21.

Alok Baikadi, Jonathan Rowe, Bradford Mott, and James Lester. 2012. Toward narrative schema-based goal recognition models for interactive narrative environments. In *Intelligent Narrative Technologies: AIIDE Workshop AAAI Technical Report WS-12-14*.

Sonia Bergamaschi, Elton Domnori, Francesco Guerra, Raquel Trillo Lado, and Yannis Velegrakis. 2011. Keyword search over relational databases: A metadata approach. In *SIGMOD 2011*. 565–576.

Bin Bi, Hao Ma, Bo-June Paul Hsu, Wei Chu, Kuansan Wang, and Junghoo Cho. 2015. Learning to recommend related entities to search users. In *International Conference on Web Search and Data Mining (WSDM'15)*. 139–148.

Nate Blaylock and James Allen. 2003. Corpus-based, statistical goal recognition. In *IJCAI 2003*. Morgan Kaufmann Publishers, 1303–1308.

Nate Blaylock and James Allen. 2005. Recognizing instantiated goals using statistical methods. In *Workshop on Modeling Others from Observations (MOO'05)*, G. Kaminka (Ed.). 79–86.

Andrei Broder. 2002. A taxonomy of web search. *SIGIR Forum* 36, 2 (2002), 3–10.

Sandra Carberry. 1983. Tracking user goals in an information-seeking environment. In *AAAI*. 59–63.

Sandra Carberry. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11, 1–2 (2001), 31–48.

Claudio Carpineto, Stefano Mizzaro, Giovanni Romano, and Matteo Snidero. 2009. Mobile information retrieval with search results clustering: Prototypes and evaluations. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 877–895.

Malú Castellanos, Meichun Hsu, Umeshwar Dayal, Riddhiman Ghosh, Mohamed Dekhil, Carlos Ceja Limon, Marcial Puchi, and Perla Ruiz. 2012. Intention insider: Discovering people's intentions in the social channel. In *Proceedings of the Extended Database Technologies Conference (EDBT)*. 614–617.

Yao-Sheng Chang, Kuan-Yu He, Scott Yu, and Wen-Hsiang Lu. 2006. Identifying user goals from web search results. In *Web Intelligence.* 1038–1041.

Eugene Charniak and Robert P. Goldman. 2013. Plan recognition in stories and in life. In *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (CoRR)*.

Muhammad Aamir Cheema, Zhitao Shen, Xuemin Lin, and Wenjie Zhang. 2014. A unified framework for efficiently processing ranking related queries. In *Proceedings of the 17th International Conference on Extending Database Technology (EDBT'14)*. 427–438.

Christos Chelmis and Viktor Prasanna. 2012. Predicting communication intention in social networks. In *International Conference on Social Computing (SocialCom'12)*. 184–194.

Christy M. K. Cheung and Matthew K. O. Lee. 2010. A theoretical model of intentional social action in online social networks. *Decision Support Systems* 49, 1 (2010), 24–30.

Wing S. Chow and Lai Sheung Chan. 2008. Social network, social trust and shared goals in organizational knowledge sharing. *Information & Management* 45, 7 (2008), 458–465.

Paul A. Crook and Oliver Lemon. 2010. Representing uncertainty about complex user goals in statistical dialogue systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. 209–212.

Fabiano Dalpiaz, Victor Silva Souza, and John Mylopoulos. 2014. The many faces of operationalization in goal-oriented requirements engineering. In *Proceedings of the Tenth Asia-Pacific Conference on Conceptual Modeling (APCCM)*. 3–7.

Munmun De Choudhury, Hari Sundaram, Ajita John, and Doree D. Seligmann. 2007. Contextual prediction of communication flow in social networks. In *Web Intelligence 2007*. 57–65.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 19, 4 (1997), 380–393.

Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. 2000. Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Uncertainty in Artificial Intelligence (UAI'00)*. Morgan Kaufmann Publishers, 176–183.

Anton N. Dragunov, Thomas G. Dietterich, Kevin Johnsrude, Matthew Mclaughlin, Lida Li, and Jonathan L. Herlocker. 2005. Tasktracer: A desktop environment to support multi-tasking knowledge workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*. ACM Press, 75–82.

Richard E. Fikes and Nils J. Nilsson. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. In *IJCAI*. Morgan Kaufmann Publishers, 608–620.

Martin Fishbein and Icek Ajzen. 1975. *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research.* Addison-Wesley.

Atsushi Fujii. 2008. Modeling anchor text and classifying queries to enhance web document retrieval. In *WWW*. ACM, 337–346.

C. W. Geib and R. P. Goldman. 2001. Plan recognition in intrusion detection systems. In *DARPA Information Survivability Conference and Exposition (DISCEX'01)*.

Kevin Gold. 2010. Training goal recognition online from low-level inputs in an action-adventure game. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Eun Y. Ha, Jonathan P. Rowe, Bradford W. Mott, and James C. Lester. 2012. Goal recognition with Markov logic networks for player-adaptive games. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

The Anh Han and Luís Moniz Pereira. 2010. Collective intention recognition and elder care. In *Proactive Assistant Agents, Papers from the 2010 AAAI Fall Symposium*.

Ahmed Hassan, Rosie Jones, and Kristina Lisa Klinkner. 2010. Beyond DCG: User behavior as a predictor of a successful search. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*. ACM, 221–230.

Yulan He. 2010. Goal detection from natural language queries. In *Applications of Natural Language to Information Systems (NLDB'10)*. Springer-Verlag, 157–168.

David Heckerman. 1996. *A Tutorial on Learning with Bayesian Networks*. Technical Report. Learning in Graphical Models.

Mauro Rojas Herrera, Edleno Silva de Moura, Marco Cristo, Thomaz Philippe C. Silva, and Altigran Soares da Silva. 2010. Exploring features for the automatic identification of user goals in web search. *Information Processing Management* 46, 2 (2010), 131–142.

Gerold Hoelzl, Marc Kurz, and Alois Ferscha. 2012. Goal oriented opportunistic recognition of high-level composed activities using dynamically configured hidden Markov models. *Procedia Computer Science* 10 (2012), 308–315.

Jun Hong. 2000. Plan recognition through goal graph analysis. In *ECAI*. 496–500.

Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. 1998. The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 256–265.

Chin-Lung Hsu and Judy Chuan-Chuan Lin. 2008. Acceptance of blog usage: The roles of technology acceptance, social influence and knowledge sharing motivation. *Information & Management* 45, 1 (2008), 65–74.

Marcus J. Huber and Richard Simpson. 2003. Plan recognition to aid the visually impaired. In *Proceedings of the 9th International Conference on User Modeling (UM'03)*. Springer-Verlag, 138–142.

Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. 2008. Determining the informational, navigational, and transactional intent of web queries. *Information Processing & Management* 44, 3 (2008), 1251–1266.

Peter A. Jarvis, Teresa F. Lunt, and Karen L. Myers. 2004. Identifying terrorist activity with AI plan recognition technology. In *Innovative Applications of Artificial Intelligence (IAAI'04)*. AAAI Press, 858–863.

Froduald Kabanza, Philipe Bellefeuille, Francis Bisson, Abder Rezak Benaskeur, and Hengameh Irandoust. 2010. Opponent behaviour recognition for real-time strategy games. In *Plan, Activity, and Intent Recognition (AAAI Workshops)*, Vol. WS-10-05. AAAI.

In-Ho Kang and GilChang Kim. 2003. Query type classification for web document retrieval. In *Proceedings of the 39th International ACM SIGIR Conference*. ACM, 64–71.

Henry A. Kautz. 1991. A formal theory of plan recognition and its implementation. In *.Reasoning About Plans*. Morgan Kaufmann Publishers, 69–124.

Sarah Keren, Avigdor Gal, and Erez Karpas. 2016a. Goal recognition design with non-observable actions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 3152–3158.

Sarah Keren, Avigdor Gal, and Erez Karpas. 2016b. Privacy preserving plans in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Uichin Lee, Zhenyu Liu, and Junghoo Cho. 2005. Automatic identification of user goals in web search. In *Proceedings of the World Wide Web Conference (WWW)*. ACM, 391–400.

Neal Lesh. 1998. *Scalable and Adaptive Goal Recognition*. Ph.D. Dissertation. University of Washington.

Neal Lesh, Charles Rich, and Candace L. Sidner. 1999. Using plan recognition in human-computer collaboration. In *Proceedings of the International Conference on User modeling*. Springer-Verlag New York, Secaucus, NJ, 23–32.

Yunyao Li, Rajasekar Krishnamurthy, Shivakumar Vaithyanathan, and H. V. Jagadish. 2006. Getting work done on the web: Supporting transactional queries. In *Proceedings of the 39th International ACM SIGIR Conference*.

Henry Lieberman. 2009. User interface goals, AI opportunities. *AI Magazine* 30, 4 (2009), 16–22.

S. Louvigne, N. Rubens, F. Anma, and T. Okamoto. 2012. Utilizing social media for goal setting based on observational learning. In *Advanced Learning Technologies*. 736–737.

Daniel Lowd and Jesse Davis. 2010. Learning Markov network structure with decision trees. In *Proceedings of the International Conference on Data Mining (ICDM'10)*. 334–343.

Yumao Lu, Fuchun Peng, Xin Li, and Nawaaz Ahmed. 2006. Coupling feature selection and machine learning methods for navigational query identification. In *International Conference on Information and Knowledge Management (CIKM'06)*. ACM, 682–689.

M. Maragoudakis, A. Thanopoulos, and N. Fakotakis. 2007. MeteoBayes: Effective plan recognition in a weather dialogue system. *IEEE Intelligent Systems* 22, 1 (2007), 67–77.

James R. Meehan. 1981. Tale-spin. In *Inside Computer Understanding*, R. Schank (Ed.). Lawrence Erlbaum, Hillsdale, NJ, 197–225.

Bradford Mott, Sunyoung Lee, and James Lester. 2006. Probabilistic goal recognition in interactive narrative environments. In *Artificial Intelligence*. AAAI Press, 187–192.

Davide Mottin, Matteo Lissandrini, Yannis Velegrakis, and Themis Palpanas. 2014. Exemplar queries: Give me an example of what you need. *PVLDB* 7, 5 (2014), 365–376.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence (UAI'99)*. Morgan Kaufmann Publishers, 467–475.

John Mylopoulos, Lawrence Chung, and Eric Yu. 1999. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM* 42, 1 (1999), 31–37.

Allen Newell. 1982. The knowledge level. *Artificial Intelligence* 18, 1 (1982), 87–127.

D. Papadimitriou. 2016. Goal-aware data management for retrieval and recommendations. In *32nd ICDE Workshops 2016*. IEEE Computer Society.

D. Papadimitriou, Y. Velegrakis, G. Koutrika, and J. Mylopoulos. 2015. Goals in social media, information retrieval and intelligent agents. In *2015 IEEE 31st International Conference on Data Engineering*. 1538–1540.

Eduardo Parra-Lpez, Jacques Bulchand-Gidumal, Desiderio Gutirrez-Tao, and Ricardo Daz-Armas. 2011. Intentions to use social media in organizing and taking vacation trips. *Computers in Human Behavior* 27, 2 (2011), 640–654.

Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz. 2003. Inferring high-level behavior from low-level sensors. In *Proceedings of International Conference UbiComp (Ubiquitous Computing)*. 73–89.

Marco Perugini and Richard P. Bagozzi. 2001. The role of desires and anticipated emotions in goal-directed behaviours: Broadening and deepening the theory of planned behaviour. *British Journal of Social Psychology* 40, 1 (2001), 79–98.

Miquel Ramırez and Hector Geffner. 2009. Plan recognition as planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence.* Morgan Kaufmann Publishers, 1778–1783.

Miquel Ramírez and Hector Geffner. 2011. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2009–2014.

Antoine Raux and Yi Ma. 2011. Efficient probabilistic tracking of user goal and dialog history for spoken dialog systems. In *INTERSPEECH*. 801–804.

Sebastian Riedel. 2012. Improving the accuracy and efficiency of map inference for Markov logic. *Empirical Methods in Natural Language Processing* abs/1206.3282 (2012).

Mark Owen Riedl. 2004. *Narrative Generation: Balancing Plot and Character*. Ph.D. Dissertation. North Carolina State University.

Dana Ron, Yoram Singer, and Naftali Tishby. 1994. Learning probabilistic automata with variable memory length. In *Proceedings of the 7th Annual ACM Conference on Computational Learning Theory*. ACM Press, 35–46.

Daniel E. Rose and Danny Levinson. 2004. Understanding user goals in web search. In *WWW*. ACM, 13–19.

Patrice Roy, Bruno Bouchard, Abdenour Bouzouane, and Sylvain Giroux. 2007. A hybrid plan recognition model for Alzheimer's patients: Interleaved-erroneous dilemma. In *International Conference on Intelligent Agent Technology 2007*. IEEE Computer Society, 131–137.

Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education.

Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the World Wide Web Conference (WWW)*. ACM, New York, NY, 841–850.

Fariba Sadri. 2012. Intention recognition in agents for ambient intelligence: Logic-based approaches. In *Agents and Ambient Intelligence*. 197–236.

Fariba Sadri. 2014. Logic-based approaches to intention recognition. In *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives*, N. Chong, F. Mastrogiovanni (Eds.). 346–375.

Roger C. Schank. 1995. *Tell Me a Story: Narrative and Intelligence*. Northwestern University Press.

Charles F. Schmidt, N. S. Sridharan, and John L. Goodson. 1978. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence* 11, 1–2 (1978), 45–83.

Amartya K. Sen, A. G. M. Last, and Randolph Quirk. 1986. Prediction and economic theory [and discussion]. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 407, 1832 (1986), 3–23.

Yoram Singer and Manfred K. Warmuth. 1996. Training algorithms for hidden Markov models using entropy based distance functions. In *Advances in Neural Information Processing Systems 9*. MIT Press, 641–647.

Dustin A. Smith and Henry Lieberman. 2010. The why UI: Using goal networks to improve user interfaces. In *Intelligent User Interfaces (IUI'10)*. ACM, 377–380.

Markus Strohmaier, Mark Kröll, and Christian Körner. 2009. Automatically annotating textual resources with human intentions. In *Proceedings of the Hypertext 2009*. 355–356.

Jigui Sun and Minghao Yin. 2007. Recognizing the agent's goals incrementally: Planning graph as a basis. *Frontiers of Computer Science in China* 1, 1 (2007), 26–36.

James D. Thompson and William J. McEwen. 1958. Organizational goals and environment: Goal-setting as an interaction process. *American Sociological Review* 23, 1 (1958), 23–31.

Martin J. Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundational Trends in Machine Learning* 1, 1–2 (Jan. 2008), 1–305.

Lynn Wilcox and M. A. Bush. 1992. Training and search algorithms for an interactive wordspotting system. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, 97–100.

Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan. 2007. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 20, 8 (2007), 17–31.

Jie Yin, Qiang Yang, Dou Shen, and Ze-Nian Li. 2008. Activity recognition via user-trace segmentation. *ACM Transactions on Sensor Networks* 4, 4, Article 19 (Sept. 2008), 34 pages.

Qing Zhang and Houfeng Wang. 2015. Improving collaborative filtering via hidden structured constraint. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. 1935–1938.

Shandian Zhe, Tian Xia, and Xueqi Cheng. 2010. Modeling users' information goal transitions and satisfaction judgment: Understanding the full search process. In *Web Intelligence*. 431–434.

Received January 2015; revised January 2016; accepted April 2016