Human-Aware Plan Recognition

Hankz Hankui Zhuo
Department of Computer Science,
Sun Yat-Sen University, Guangzhou, China. 510006
zhuohank@mail.sysu.edu.cn

Abstract

Plan recognition aims to recognize target plans given observed actions with history plan libraries or domain models in hand. Despite of the success of previous plan recognition approaches, they all neglect the impact of human preferences on plans. For example, a kid in a shopping mall might prefer to "executing" a plan of playing in water park, while an adult might prefer to "executing" a plan of having a cup of coffee. It could be helpful for improving the plan recognition accuracy to consider human preferences on plans. We assume there are historical rating scores on a subset of plans given by humans, and action sequences observed on humans. We estimate *unknown* rating scores based on rating scores in hand using an off-the-shelf collaborative filtering approach. We then discover plans to best explain the estimated rating scores and observed actions using a skip-gram based approach. In the experiment, we evaluate our approach in three planning domains to demonstrate its effectiveness.

Introduction

Plan recognition aims to look for target plans to best explain the observed actions based on plan libraries and/or domain models (Kautz and Allen 1986; Ramírez and Geffner 2009a; Zhuo, Yang, and Kambhampati 2012). Computer-aided cooperative work scenarios become increasingly popular, human-in-the-loop decision support has become a critical challenge (Cohen et al. 2015; Dong et al. 2004; Manikonda et al. 2014). An important aspect of such a support is recognizing what plans the human in the loop is making. There have been large amount of works on plan recognition. For example, Kautz and Allen proposed an approach to recognizing plans based on parsing observed actions as sequences of subactions and essentially model this knowledge as a context-free rule in an "action grammar" (Kautz and Allen 1986); Bui et al. (Bui 2003; Geib and Goldman 2009) presented approaches to probabilistic plan recognition problems; Kabanza and Filion (Kabanza et al. 2013) proposed an anytime plan recognition algorithm to reduce the number of generated plan execution models based on weighted model counting; just to name a few.

Despite the success of previous approaches, they all ignore the preferences humans have on the plans they aim to execute. In many real-world applications, humans often have different preferences on plans. For example, in travel planning systems, such as *triphobo*[1], which provides personalized trip plans to 14000 cities, travelers would like to have a trip from Hong Kong to Phoenix. Some travelers may prefer to stay at Los Angeles to visit Disney Land before going to Phoenix, while others may prefer to go directly from Los Angeles to Phoenix. Suppose we observe two actions that a traveler "flies from Hong Kong to Los Angeles" and "flies from Los Angeles to Phoenix". We do not have any clues telling us the traveler is executing the former plan or the latter. As another example, a kid in a shopping mall might prefer to "executing" a plan of playing in water park, while an adult might prefer to "executing" a plan of having a cup of coffee.

In this paper, we aim to consider human preferences when recognizing plans with observed actions. We assume that human preferences are described by a matrix of rating scores, suggesting that human provides a rating score on a plan after he executes the plan to express his interest in the plan. It is feasible to collect rating scores in real world applications. For example, in travel planning systems, they collect rating scores after travelers finish their trips by asking the travelers' feedbacks. We also assume we have a set of observations of action sequences. It is also possible in real world applications. For example, travel systems can "observe" some actions of travelers by communicating with travelers via mobile phone apps.

With rating scores and observed action sequences as input, we propose a novel plan recognition approach called `HARE`, which stands for **H**uman-**A**ware plan **RE**cognition. In `HARE`, we first estimate the possible preferences (in the form of rating scores) humans may have on plans that they did not execute before according to "relations" to others' rating scores. We then learn vector representations of actions to calculate the possibility of the target plans that cover the observed action sequences. After that we compute the final target plans that best explain the estimated rating scores and observed action sequences.

In the remainder of the paper we organize the paper as

[1] https://www.triphobo.com

follows. We first review previous work on plan recognition problems. After that we give a formal definition of our human-aware plan recognition problem and present the details of our HARE approach step by step. We then evaluate our HARE approach by comparing to previous approaches refined to allow human ratings as input to exhibit the effectiveness of our HARE approach. Finally we conclude the paper with future work.

Related work

Kautz and Allen proposed to recognize plans based on parsing observed actions as sequences of subactions and essentially model this knowledge as a context-free rule in an "action grammar" (Kautz and Allen 1986). All actions, plans are uniformly referred to as goals, and a recognizer's knowledge is represented by a set of first-order statements called event hierarchy encoded in first-order logic, which defines abstraction, decomposition and functional relationships between types of events. Kabanza and Filion (Kabanza et al. 2013) proposed an anytime plan recognition algorithm to reduce the number of generated plan execution models based on weighted model counting. These approaches are, however, difficult to represent uncertainty. They offer no mechanism for preferring one consistent approach to another and incapable of deciding whether one particular plan is more likely than another, as long as both of them can be consistent enough to explain the actions observed.

Instead of using a library of plans, Ramirez and Geffner (Ramirez and Geffner 2009b) proposed an approach to solving the plan recognition problem using slightly modified planning algorithms, assuming the action models were given as input. Except previous work (Kautz and Allen 1986; Bui 2003; Geib and Goldman 2009; Ramirez and Geffner 2009b) on the plan recognition problem presented in the introduction section, Note that action models can be created by experts or learnt by previous systems, such as ARMS (Yang, Wu, and Jiang 2007) and LAMP (Zhuo et al. 2010), Amir and Gal addressed a plan recognition approach to recognizing student behaviors using virtual science laboratories (Amir and Gal 2011). Ramirez and Geffner exploited off-the-shelf classical planners to recognize probabilistic plans (Ramirez and Geffner 2010).

Early work on human-in-the-loop planning scenarios in automated planning went under the name of "mixed-initiative planning" (e.g. (Ferguson, Allen, and Miller 1996)). Different from our work, that work was that the humans in the loop were helping the automated planner (with a complete action model) navigate its search space of plans more efficiently. In contrast, we are interested in planning technology that helping humans develop plans, even in the absence of complete formal models of the planning domain. While some work in web-service composition (c.f. (Dong et al. 2004)) did focus on this type of planning support, they were hobbled by being limited to simple input/output type comparison. In contrast, we believe that HARE learns and uses a model that captures more of the structure of the planning domain (while still not insisting on complete action models). While HARE focuses on exploiting human preferences and learning models from plan corpora, some recent work looked at using crowdsourcing to acquire domain models. For example, Lasecki et al. (Lasecki et al. 2013) introduce Legion:AR, which combines the benefits of automatic and human activity labeling for robust and deployable activity recognition. By engaging a group of people, Legion:AR is able to label activities as they occur more reliably than a single person can, especially in complex domains with multiple actors performing activities quickly. Such crowdsourcing methods can complement the plan-corpus based approach proposed in HARE.

Problem Definition

A matrix of rating scores is denoted by $\mathcal{R} = [r_{ui}]$, where r_{ui} is a rating score given by user u on plan p_i. A rating score is either an integer from $\{1, 2, 3, 4, 5\}$, or a symbol "?" indicating the score is unknown (to be estimated). A set of users is denoted by U, a set of plans is denoted by $\mathcal{L}$ called a plan library, a set of actions is denoted by A. A plan p is composed of an action sequence $\langle a_1, a_2, \ldots, a_n \rangle$, where $a_i \in A$ $(1 \leq i \leq n)$. A set of observations $\mathcal{O} = \{O_u^s\}$ specifies the actions observed by monitors such as "sensors" or "log recorders", where O_u^s is the sth action sequence observed on u. Note that for the simplicity of specification we consider the observations as actions directly, i.e., each observed action in O_u^s is in A. In real world applications, O_u^s can be replaced by sensor signals which can further be projected to actions.

Our problem can be defined as follows. Given a matrix of rating scores $\mathcal{R} = [r_{ui}]$, and a set of observations $\mathcal{O} = \{O_u^s\}$, we aim to output a set of plans $\tilde{P} = \{p\}$ that best explains $\mathcal{O}$, where $p \in \mathcal{L}$. An example input of our recognition problem in the blocks[2] domain is shown in Figure 1, where Figure 1(a) is p_1, p_2, p_3, p_4, p_5 are five plans described in Figure1(c), Figure 1(b) is a set of observations with respect to different humans u_1, u_2, u_3, u_4, u_5. For example, "stack-B-A,stack-C-B" indicates an observed action sequence on human u_1; other sequences are omitted. An example output of our approach, given the input shown in Figure 1, is p_3, p_1, p_1, p_3, p_2 for u_1, u_2, u_3, u_4, u_5, respectively.

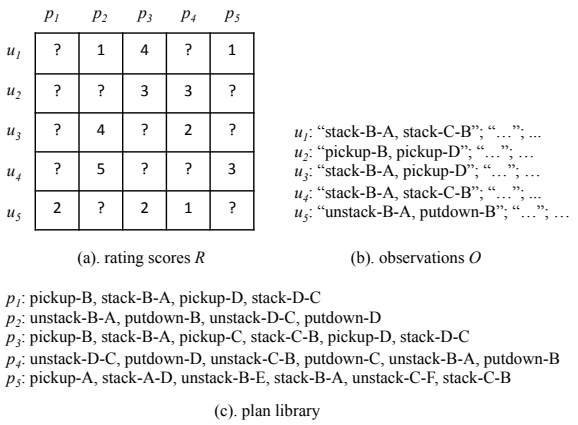
Our HARE Algorithm

In this section we present our HARE algorithm in detail. An overview of HARE is shown in Algorithm 1, where we first learn vector representations of actions and plans (i.e., Steps 1 and 2 of Algorithm 1), and estimate the rating scores of plans (i.e., Steps 3 and 4) and recognize plans based on the estimated rating scores (i.e., Step 5).

Learning Representations of Actions and Plans

To search a plan that can best explain the observed actions, we exploit a vector-representation approach to calculating the probability of a plan given the observed actions (i.e., Step 1 of Algorithm 1). This approach takes into account the "grammar" information behind all the plans, which has been demonstrated effective in completing plans by (Tian, Zhuo, and Kambhampati 2016). In the following we describe the

[2] http://www.cs.toronto.edu/aips2000/



	p_1	p_2	p_3	p_4	p_5
u_1	?	1	4	?	1
u_2	?	?	3	3	?
u_3	?	4	?	2	?
u_4	?	5	?	?	3
u_5	2	?	2	1	?

(a). rating scores R

u_1: "stack-B-A, stack-C-B"; "…"; ...
u_2: "pickup-B, pickup-D"; "…"; …
u_3: "stack-B-A, pickup-D"; "…"; …
u_4: "stack-B-A, stack-C-B"; "…"; ...
u_5: "unstack-B-A, putdown-B"; "…"; …

(b). observations O

p_1: pickup-B, stack-B-A, pickup-D, stack-D-C
p_2: unstack-B-A, putdown-B, unstack-D-C, putdown-D
p_3: pickup-B, stack-B-A, pickup-C, stack-C-B, pickup-D, stack-D-C
p_4: unstack-D-C, putdown-D, unstack-C-B, putdown-C, unstack-B-A, putdown-B
p_5: pickup-A, stack-A-D, unstack-B-E, stack-B-A, unstack-C-F, stack-C-B

(c). plan library

Figure 1: An example input of our human-aware plan recognition problem

Algorithm 1 An overview of our HARE algorithm

input: Rating scores $\mathcal{R}$, observations $\mathcal{O}$.
output: Plans $\tilde{P}$.

1: Learn vector representations w_i of actions a_i based on the plan set P (specified in $\mathcal{R}$).
2: Calculate vector representations of plans $\tilde{V}$.
3: Estimate rating scores based on matrix factorization:

$$\mathcal{R} \sim UV^T \text{ and } V \sim \tilde{V}$$

where U is a matrix of user features, and V is a matrix of plan features.
4: Calculate the estimated rating scores by $\mathcal{R}^* = UV^T$.
5: Calculate $\tilde{P}$ that best explains $\mathcal{O}$ based on $\mathcal{R}^*$ and vector representations of actions w_i.
6: **return** $\tilde{P}$

detail of the learning procedure, which has been presented in (Tian, Zhuo, and Kambhampati 2016) as well.

Since actions are denoted by a name strings, actions can be viewed as words, and a plan can be viewed as a sentence. Furthermore, the plan library $\mathcal{L}$ can be seen as a corpus, and the set of all possible actions $\mathcal{A}$ is the vocabulary. We thus can learn the vector representations for actions using the Skip-gram model with hierarchical softmax, which has been shown an efficient method for learning high-quality vector representations of words from unstructured corpora (Mikolov et al. 2013).

The objective of the Skip-gram model is to learn vector representations for predicting the surrounding words in a sentence or document. Given a corpus $\mathcal{C}$, composed of a sequence of training words $\langle w_1, w_2, \ldots, w_T \rangle$, where $T = |\mathcal{C}|$, the Skip-gram model maximizes the average log probability

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c \le j \le c, j \neq 0} \log p(w_{t+j}|w_t) \quad (1)$$

where c is the size of the training window or context.

The basic probability $p(w_{t+j}|w_t)$ is defined by the hierarchical softmax, which uses a binary tree representation of the output layer with the K words as its leaves and for each node, explicitly represents the relative probabilities of its child nodes (Mikolov et al. 2013). For each leaf node, there is an unique path from the root to the node, and this path is used to estimate the probability of the word represented by the leaf node. There are no explicit output vector representations for words. Instead, each inner node has an output vector $v'_{n(w,j)}$, and the probability of a word being the output word is defined by

$$p(w_{t+j}|w_t) = \prod_{i=1}^{L(w_{t+j})-1} \Big\{\sigma(\mathbb{I}(n(w_{t+j}, i+1) = child(n(w_{t+j}, i))) \cdot v_{n(w_{t+j},i)} \cdot v_{w_t})\Big\}, \quad (2)$$

where $\sigma(x) = 1/(1+\exp(-x))$. $L(w)$ is the length from the root to the word w in the binary tree, e.g., $L(w) = 4$ if there are four nodes from the root to w. $n(w, i)$ is the ith node from the root to w, e.g., $n(w, 1) = root$ and $n(w, L(w)) = w$. $child(n)$ is a fixed child (e.g., left child) of node n. v_n is the vector representation of the inner node n. v_{w_t} is the input vector representation of word w_t. The dimension of vector representations is denoted by $v_n = D$, which is a preset constant. The identity function $\mathbb{I}(x)$ is 1 if x is true; otherwise it is -1.

We can thus build vector representations of actions by maximizing Equation (1) with corpora or plan libraries $\mathcal{L}$ as input. We will exploit the vector representations to discover the unknown plan $\tilde{P}$ in the next subsection.

With the vector representations of actions, we take a straightforward way to calculate vector representations of plans by computing an average over all of the action representations. Specifically, vector representation $\tilde{V}_i$ of plan $p_i = \langle w_1, w_2, \ldots, w_L \rangle$ can be defined by $\tilde{V}_i = \frac{1}{L}\sum_{x=1}^{L} v_{w_x}$, where L is the length of plan p_i, and v_{w_x} is the vector representation of action w_x. As a result, the plan library $\mathcal{L}$ can be represented by $\tilde{V}$ with dimension $|\mathcal{L}| \times D$, where D is the dimension of vector representations of both actions and plans.

Estimating Rating Scores

In Step 3 of Algorithm 1, we aim to estimate the rating scores missing in the given rating score matrix $\mathcal{R}$ using matrix factorization, which has been applied to recommender systems. The objective function is defined by

$$\min_{U_i, V_j} \sum_{i,j} one(i,j)\{(r_{i,j} - U_i V_j^T)^2 + \lambda_1 ||V_j - \tilde{V}_j||_2 + \lambda_2(||U_i||^2 + ||V_j||^2)\}, \quad (3)$$

where $one(i, j)$ is 1 if $r_{i,j} \neq$ "?", and 0 otherwise. λ_1 and λ_2 are constants used to control the regularization. U_i is a feature vector characterizing ith user, and V_j is a feature vector characterizing jth plan, whose dimension is set to be the same as $\tilde{V}_j$, i.e., $|V_j| = D$. The first term of Equation (3) suggests $\mathcal{R} \sim UV^T$ and the second term indicates

$V \sim \tilde{V}$. Note that we assume that the resulting feature vector V should be close to $\tilde{V}$ learnt from the plan library.

We exploit a stochastic gradient algorithm to learn the parameters U and V, and calculate the estimated rating scores by $\mathcal{R}^* = UV^T$, i.e., Step 4 of Algorithm 1.

Calculating the Recognized Plans

In this subsection we aim to build a model to discover plans that can best explain both estimated rating scores and observed actions. To consider the two factors, rating scores and observed actions, we exploit a straightforward way by multiplying these two factors, indicating (1) the larger the rating score is, the higher the possibility of the plan to be the target plan is; (2) the larger the likeliness of the observed actions being covered by the plan, the higher the possibility of the plan to be the target plan is.

We define the objective function below:

$$\mathcal{G}(O_u^s, p_i, W, r_{ui}^*) = r_{ui}^* F(p_i|O_u^s, W), \tag{4}$$

where O_u^s is the sth observed action sequence with respect to user u, p_i is a plan from the plan library, $W = \{w_i\}$ is a set of vector representations of actions in A, and r_{ui}^* is the estimated rating scores on plan p_i given by user u. $F(p_i|O_u^s, W)$ is defined by

$$F(p_i|O_u^s, W) = \mathbb{I}(p_i, O_u^s)\{\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} \log p(w_{t+j}|w_t)\}, \tag{5}$$

where $\mathbb{I}(p_i, O_u^s)$ is $\frac{L}{|p_i|}$ if plan p_i *covers* observed action sequence O_u^s, i.e., there exists a subsequence of p_i, $\langle a_{k_1} a_{k_2} \dots a_{k_L}\rangle$, such that $O_u^s = \langle a_{k_1} a_{k_2} \dots a_{k_L}\rangle$, where L is the length of actions in O_u^s and $1 \le k_1 < k_2 < \dots < k_L \le |p_i|$; $\mathbb{I}(p_i, O_u^s)$ is zero, otherwise. The intuition of $\mathbb{I}(p_i, O_u^s)$ is that the larger the ratio of actions in a plan is observed, the larger likely is the plan to be the target one.

Algorithm 2 Calculating the final recognized plans

input: Rating scores $\mathcal{R}^* = [r_{ui}^*]$, observations $\mathcal{O} = \{O_u^s\}$
output: Plans $\tilde{P}$

1: $\tilde{P} = \emptyset$
2: **for** Each user u and each sth sequence of user u **do**
3: Calculate a plan $\tilde{p}_u$ that best explain $O_u^s \in \mathcal{O}$:

$$\tilde{p}_u = \arg\max_{p_i \in P} \mathcal{G}(O_u^s, p_i, W, r_{ui}^*) \tag{6}$$

4: $\tilde{P} = \tilde{P} \cup \{\tilde{p}_u\}$
5: **end for**
6: **return** $\tilde{P}$

The framework of calculating the final recognized plans is shown in Algorithm 2, where O_u^s is the sth observed action sequence of user u.

Handling Cold Start Issue

In our human-aware plan recognition problem we assume the plan library is finite and should be provided in advance before doing collaborative filtering procedure (i.e., Step 3 of Algorithm 1). It is, however, possible that new plans are added to the plan library and do not have any rating scores, which is known as cold start problem, a difficult problem in collaborative filtering. Our approach can be easily extended to handling the cold start problem since we can calculate similarity between plans based on vector representations of plans and transfer ratings of plans that are already in the plan library by assuming that humans have similar interests in similar plans. We calculate ratings $r_{u,new}$ of new coming plans p_{new} with respect to user u below:

$$r_{u,new} = \sum_{i=1}^{|\mathcal{L}|} r_{ui} * similarity(\tilde{V}_i, V'_{new}), \tag{7}$$

where $r_{ui} \in \mathcal{R}^*$ is the rating of plan $p_i \in \mathcal{L}$ given by Step 4 of Algorithm 1, V'_{new} is the vector of new plan p_{new}, and $similarity(\tilde{V}_i, V'_{new})$ is the similarity between plan p_i and new plan p_{new}, which is defined by the cosine similarity between their corresponding vectors (which is rescaled to (0,1]). In this way $\mathcal{R}^*$ can be extended to a new matrix to incorporate p_{new}. The remaining procedure is the same as Step 5 of Algorithm 1.

Experiments

To evaluate the effectiveness of our algorithm, we built a system to simulate real-world applications and synthesized training and testing data. We generated data in three planning domains blocks[2], depots[3], and driverlog[3].. The reason why we used planning domains is it is simple to generate plans by running an off-the-shelf planner. It is similar to apply our approach to other domains with historical plans in hand instead of generation with planners. To generate training data, we randomly created 5000 planning problems for each domain, and solved these planning problems with a planning solver, such as FF[4], to produce 5000 plans, whose length is various (generally from 40 to 200). Features of datasets are shown in Table 1, where the second column is the number of plans generated, the third column is the total number of words (or actions) of all plans, and the last column is the size of vocabulary used in all plans.

Table 1: Features of datasets

domain	#plan	#word	#vocabulary
blocks	5000	292250	1250
depots	5000	209711	2273
driverlog	5000	179621	1441

In each domain, we generated the dataset with the size of 10000 virtual humans. In order to generate rating scores on plans, we need to consider the homogeneous preferences on plans among humans. We divided humans into 100 groups, with 100 members in each group, and characterized each

[3]http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume20/long03a-html/JAIRIPC.html

[4]https://fai.cs.uni-saarland.de/hoffmann/ff.html

plan p with a vector defined by:

$$vector(p) = \sum_{i=1}^{K} (\frac{1}{K})^{i-1} w_i, \quad (8)$$

where K is the length of plan p, w_i is the vector representation of the ith action in plan p. We can see that the definition of $vector(p)$ can be used to characterize different plans. For example, two different plans p_1 and p_2 with swapped actions have different vectors $vector(p_1)$ and $vector(p_2)$ (even though the length of p_1 and p_2 is identical). Note that it is possible to explore other definitions of vector representations of plans. We assume that the rating score given by human group g ($1 \leq g \leq 100$) depends on the Gaussian distribution $\mathcal{N}(\mu_g, 1)$, i.e.,

$$Prop(||vector(p)||) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(||vector(p)|| - \mu_g)^2}, \quad (9)$$

where μ_g is the expected value of $|vector(p)|$ ($||v||$ indicates the norm of vector v), and 1 is the variance. We denote the mean of all vector representations of plans by

$$\bar{\mu} = \frac{1}{5000} \sum_{i=1}^{5000} ||vector(p_i)||.$$

To capture different preferences among groups, we define μ_g by: $\mu_g = \frac{g}{100}\bar{\mu}$, for each group g from 1 to 100. Note that we assume humans in the same group have identical preferences (or identical Gaussian distribution) on rating plans. We generated a matrix of rating scores $\mathcal{R}$ by

$$r_{ui} = \text{round-up}(5 \cdot Prop(||vector(p_i)||)),$$

where round-up(x) indicates x is rounded up into an integer.

In order to generate ground-truth plans to be recognized with respect to observed action sequences for each human, we calculated a set of plans that *covers* the observed actions, and selected the plan based on Equation (6) by replacing the r^*_{ui} with the generated rating score r_{ui}. We generated 20 action sequences (viewed as the observed action sequences) for each human. We denoted the set of ground-truth plans corresponding to 20 observed action sequences of each human (5000 humans in total) by P^{truth}. We further calculated the plans corresponding to the 20 observed action sequences using HARE (with a ratio of rating scores missing in the generated matrix $\mathcal{R}$), which were denoted by $P^{\texttt{HARE}}$. The accuracy of our HARE algorithm is defined by

$$accuracy = \frac{|P^{truth} \cap P^{\texttt{HARE}}|}{|P^{truth}|}.$$

In the following subsections, we evaluate our HARE algorithm by varying the ratio of rating scores and the size of observed actions. Note that we randomly selected rating scores as known rating scores and excluded others from the matrix of rating scores.

To the best of our knowledge, there are no state-of-the-art plan recognition approaches that can be directly applied to our problem. We thus refine plan recognition approaches off the shelf, such as MARS (Zhuo and Li 2011) and DUP (Tian, Zhuo, and Kambhampati 2016), to incorporate human ratings. The refining procedure is shown as follows.

- HA−MARS: MARS aims to recognize team plans from the plan library given an observed team trace. The high-level idea of MARS is to build a set of weighted constraints based on plan libraries and team traces, and solve all of the weighted constraints by a MAXSAT solver, such as MaxHS (Davies and Bacchus 2013), discover a subset of team plans in the plan library to best explain the observed team traces. To exploit MARS to solve our human-aware recognition problem, we set the number of team members to be one in both plan libraries and team traces, and viewed human ratings of team plans in the plan library as a multiplier of the weights of constraints built based on the corresponding team plans. Different from our plan recognition problem, MARS assumes positions of missing actions in observe team traces are known in advance. We thus provided additional input information about positions of missing actions in the observed action sequences when using MARS to solve our recognition problem. Note that we directly assigned *unknown* ratings with an average over all *known* ratings. We denote the refined MARS by HA−MARS, short for **H**uman-**A**ware MARS.
- HA−DUP: DUP aims to learn vector representations of actions from plan libraries and exploit the representations to discover underlying complete plans of partially observed plans. Similar to MARS, DUP assumes positions of missing actions in observed action sequences are known in advance. Likewise, we provided additional information of positions of missing actions in observed action sequences when using DUP to solve our problem. We also directly assigned *unknown* ratings with an average over all *known* ratings. Since DUP randomly samples plans to cover observed actions, which are probably not in plan libraries, we refined the sampling procedure by iteratively resampling plans until the sampled plans belong to plan libraries. We viewed the ratings as multipliers of probability of sampled plans, i.e., $r \times P(p_i|W)$, where r is the rating score of p_i and $P(p_i|W)$ is the probability of p_i given vector representations W of actions. We denote the refined DUP by HA−DUP, indicating **H**uman-**A**ware DUP.

Accuracy w.r.t. Ratio of Rating Scores

We first evaluated our HARE algorithm by varying ratios of ratings in $\mathcal{R}$ to see the effectiveness of ratings. We compared our HARE approach to both HA−MARS and HA−DUP. We set the window of training context c in Equation (1) to be three, constants λ_1 and λ_2 in Equation 3 to be 0.5, and number of observed actions to be 20 for each observed action sequence. The results are shown in Figure 2.

From Figure 2, we can see that in all three domains, the accuracy of HARE is generally higher than both HA−MARS and HA−DUP, which verifies that our HARE algorithm can indeed better utilize human preferences (in the form of rating scores) for recognizing plans from the plan library when incorporating vector representations of plans with collaborative filtering, i.e., much better than directly calculating an average over all of the known ratings, as done by HA−MARS and HA−DUP. Note that both HA−MARS and HA−DUP take additional information about positions of missing actions in

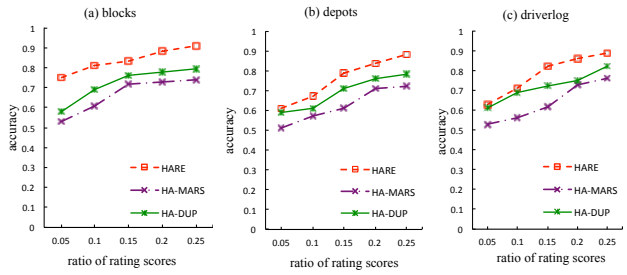


Figure 2: Accuracy w.r.t. different ratio of rating scores

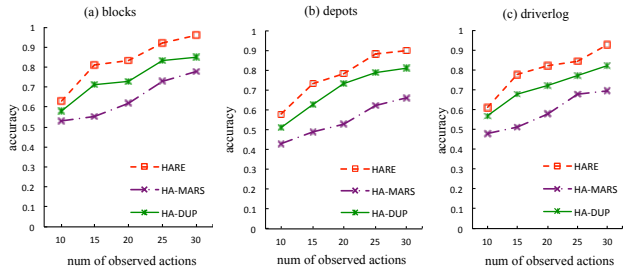


Figure 3: Accuracy w.r.t. different number of actions

the observed action sequences, while HARE does not utilize any position information of missing actions.

We can also see that both HARE and HA−DUP outperform HA−MARS in all three domains. This is because both HARE and HA−DUP leverage the semantic information of plans (represented by vector representations) to help discover underlying plans behind observed actions, and from the result we can see that the semantic information of plans can indeed improve the recognition accuracy, compared to HA−MARS which does not utilize this information. Looking at the changes of accuracies with respect to the ratio of rating scores, we can see that all the three algorithms generally become better when the ratio of available rating scores becomes larger in all three domains. This is consistent with our intuition since the larger the ratio of rating scores is, the more information is available for better describing the human preferences.

Accuracy w.r.t. Number of Observed Actions

We next evaluate the performance of our HARE algorithm with respect to the number of observed actions. Likewise, we set the context window c used in Equation (1) to be three, the constants λ_1 and λ_2 in Equation 3 to be 0.5 and the ratio of rating scores $\mathcal{R}$ to be 0.15. We varied the number of observed actions from 10 to 30. The results are shown in Figure 3.

From Figure 3, we find that accuracies of the three approaches generally become larger when the size of the observed actions increases in all three domains. This is consistent with our intuition, since the more the observed actions are, the more information is available for them to better discover the target plans. We can also see that the accuracy of our HARE algorithm is generally larger than HA−DUP and HA−MARS in all of the three domains, which verifies that our HARE algorithm can indeed better leverage rating scores by incorporating vector representations of plans with collaborative filtering, than both HA−DUP and HA−MARS, which directly calculate an average of ratings over all of the *known* ratings. Note that the average length of plans in the plan library is around 100 which is much larger than the observed actions.

Handling Cold Start Issue

To see the accuracy of our HARE algorithm when there are new plans (without any ratings) added into the plan library, we randomly added 100 new plans to the plan library (their underlying ratings were also generated according to Equation (9), but unknown to HARE). Likewise, we set the context window c used in Equation (1) to be three, the constants λ_1 and λ_2 in Equation 3 to be 0.5, the ratio of rating scores $\mathcal{R}$ to be 0.15, and the number of observed actions to be 20. We ran all of the three approaches in the *blocks* domain. The accuracies of HARE, HA−MARS and HA−DUP are 0.80, 0.58, and 0.62, respectively. This indicates that our HARE approach can indeed handle the cold start issue better than other approaches.

Final Remarks

In this paper we propose to recognize plans based on human preferences in the form of rating scores. We borrow the ideas of matrix factorization to estimate unknown rating scores and vector representations of actions and plans to estimate the probability of a plan given observations. We provided an algorithm framework to incorporate vector representations of plans with collaborative filtering and exhibit that it is effective on recognizing plans from the plan library, even though new plans without any ratings are added to the plan library.

In the future, it would be interesting to consider variable rating scores. Our approach can be seen as considering a snap-shot of human rating scores, which can be extended to accommodating variable rating scores by considering relationship between rating scores and their corresponding variable rating scores, based on Markov assumptions for example. The resulting algorithm can be seen as an evolutionary model of our approach based on rating scores varied over time. In addition, in this paper we evaluated our approach in a synthesized dataset generated from a simulation system. In the future we hope to collect data from real world applications, e.g., in travel planning systems, and evaluate the our approach in the real world dataset.

Acknowledgements:

Hankz Hankui Zhuo thanks the National Key Research and Development Program of China (2016YFB0201900), Natural Science Foundation (No. 61309011), Pearl River Science and Technology New Star of Guangzhou, and Guangdong Province Key Laboratory of Big Data Analysis and Processing for the support of this research.

References

Amir, O., and Gal, Y. K. 2011. Plan recognition in virtual laboratories. In *Proceedings of IJCAI*, 2392–2397.

Bui, H. H. 2003. A general model for online probabilistic plan recognition. In *Proceedings of IJCAI*, 1309–1318.

Cohen, P. R.; Kaiser, E. C.; Buchanan, M. C.; Lind, S.; Corrigan, M. J.; and Wesson, R. M. 2015. Sketch-thru-plan: a multimodal interface for command and control. *Commun. ACM* 58(4):56–65.

Davies, J., and Bacchus, F. 2013. Exploiting the power of mip solvers in maxsat. In *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, 166–181.

Dong, X.; Halevy, A. Y.; Madhavan, J.; Nemes, E.; and Zhang, J. 2004. Simlarity search for web services. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, 372–383.

Ferguson, G.; Allen, J.; and Miller, B. 1996. Trains-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, 70–77. Edinburgh, Scotland.

Geib, C. W., and Goldman, R. P. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence* 173(11):1101–1132.

Kabanza, F.; Filion, J.; Benaskeur, A. R.; and Irandoust, H. 2013. Controlling the hypothesis space in probabilistic plan recognition. In *IJCAI*.

Kautz, H. A., and Allen, J. F. 1986. Generalized plan recognition. In *Proceedings of AAAI*, 32–37.

Lasecki, W. S.; Song, Y. C.; Kautz, H. A.; and Bigham, J. P. 2013. Real-time crowd labeling for deployable activity recognition. In *CSCW*, 1203–1212.

Manikonda, L.; Chakraborti, T.; De, S.; Talamadupula, K.; and Kambhampati, S. 2014. AI-MIX: using automated planning to steer human workers towards better crowdsourced plans. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 3004–3009.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Ramírez, M., and Geffner, H. 2009a. Plan recognition as planning. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 1778–1783.

Ramirez, M., and Geffner, H. 2009b. Plan recognition as planning. In *Proceedings of IJCAI*, 1778–1783.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of AAAI*, 1121–1126.

Tian, X.; Zhuo, H. H.; and Kambhampati, S. 2016. Discovering underlying plans based on distributed representations of actions. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, 1135–1143.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence Journal* 171:107–143.

Zhuo, H. H., and Li, L. 2011. Multi-agent plan recognition with partial team traces and plan libraries. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 484–489.

Zhuo, H. H.; Yang, Q.; Hu, D. H.; and Li, L. 2010. Learning complex action models with quantifiers and implications. *Artificial Intelligence* 174(18):1540–1569.

Zhuo, H. H.; Yang, Q.; and Kambhampati, S. 2012. Action-model based multi-agent plan recognition. In *Neural Information Processing Systems*, 377–385.