Meta-Level Patterns for Interactive Knowledge Capture

Jihie Kim
University of Southern California/
Information Sciences Institute
Marina del Rey, CA 90292 USA
jihie@isi.edu

ABSTRACT
Current knowledge acquisition tools have limited understanding of how users enter knowledge and how acquired knowledge is used, and provide limited assistance in organizing various knowledge authoring tasks. Users have to make up for these shortcomings by keeping track of past mistakes, current status, potential new problems, and possible courses of actions by themselves. In this paper, we present a novel extension to existing knowledge acquisition tools where the system organizes the episodes of past interactions through a set of declarative *meta-level patterns* and improves its suggestions based on relevant episodes. In particular, we focus on 1) assessing the level of confidence in suggesting an action, 2) suggesting how a knowledge authoring action can be done based on successful past actions, and 3) monitoring dynamic changes in the environment to suggest relevant modifications in the knowledge base. A preliminary study with varying synthetic user interactions shows that this meta-level assessment may reduce the number of incorrect suggestions, prevent some of the user mistakes and improve the overall problem solving results.

Categories and Subject Descriptors
I.2.0 [Computing Methodologies]: Artificial Intelligence

General Terms: Algorithms, Human Factors

Keywords: Knowledge acquisition.

INTRODUCTION
Knowledge acquisition (KA) remains a key challenge to knowledge-based AI applications. There have been increasing interests in supporting end users (i.e., ordinary users who do not have a computer science background) to directly enter complex problem solving knowledge on how to perform tasks. Although these techniques have been applied in building sizable knowledge bases in some cases, detailed analyses of the user interactions reveal that users remain largely responsible for the acquisition process[2]. Ideally, KA tools should be able to exploit past interaction history in assisting users. For example, systems could suggest a KA action according to the level of confidence assessed in performing the action. The system could suggest how a KA action can be done based on similar KA actions that have been successful. When there are dynamic changes in the environment, the system could recognize them and suggest making appropriate changes in the knowledge base. The challenges in supporting these meta-level capabilities and providing improved suggestions include:

- The system should access past situations that are relevant to a KA action and assess the level of confidence in suggesting the action.
- The system should relate the current KA action to similar KA actions in the past and assess whether the actions led to successful results or not.
- The system should recognize dynamic changes in the environment and decide how to guide users in entering or modifying relevant knowledge (called k-items).

In this paper, we present a novel extension to existing KA tools where the system adds an additional layer to existing tools and explicitly keeps track of past interactions through a set of declarative *meta-level patterns*. Meta-level patterns find and collect a set of interaction *episodes* (i.e., sequences of events) that are relevant to KA actions. In particular they keep track of 1) the episodes that either *support* or *oppose* an action, 2) the episodes where KA actions either improved or didn't improve the knowledge base, and 3) the episodes where certain situations led to unexpected modification of some k-items. These episodes that are captured in pattern instantiations are used in improving suggestions. In suggesting a KA action, the system presents its levels of confidence based on its supporting and opposing episodes. The system also provides additional suggestion on how to perform an action by retrieving similar past actions that were successful. Suggestions on k-item modifications are further improved base on whether there were potentially relevant changes in the environment.

We have built a system called Echo (mEta-Cognitive History analysis and Organization) that implements these capabilities. We have tested the system with a domain of interactive scheduling where the user incrementally builds scheduling constraints and the user entered constraints assist users during scheduling. The details of the system are described in [1].

IMPROVING KA SUGGESTIONS WITH META-LEVEL PATTERNS

The standard steps to assist users in entering k-items would be:

- If there is no matching k-item for a problem solving step and execution of the step fails (e.g. cancellation of

K-CAP'05, October 2–5, 2005, Banff, Alberta, Canada.
ACM 1-59593-163-5/05/0010.

a meeting that is scheduled without k-item assistance), ask the user to enter a new k-item for such step.

- If there is an inconsistency between a k-item and past and/or current results (e.g. Gary's 5:30pm meeting that is chosen based on a k-item failed), ask the user to modify the k-item to make it consistent with the results.

Existing KA systems either ignore some of these steps, or follow similar steps rather implicitly. That is, the above steps are not explicit in the design and they are reflected in the implementation of the tool. The example-based validation approaches refine k-items collectively, and cannot tell 'when' and 'how' k-items can be improved or how confident the system is in modifying or adding k-items.

The interaction history contains events from different phases: 1) KA events (e.g. entering a new k-item), 2) problem solving events (e.g. deciding Gary's meeting time using a k-item), 3) execution events (e.g. the scheduled meeting was successfully performed) and 4) earlier suggestions. In identifying episodes as *meaningful* sequence of events that are relevant to a KA action, we focus on dependencies among these events that are either consistent or inconsistent with the action.

MP1: Assess need of a new k-item (PS-step1)
Suggestion: (CREATE_KITEM_FOR_STEP PS-step1)
When to instantiate pattern: (NO_APPLICABLE_KITEM PS-step1)
When to produce suggestion:
(NO_APPLICABLE_KITEM_AND_EXE_FAIL PS-step1)
Supporting episodes:
(NO_APPLICABLE_KITEM_AND_EXE_FAIL PS-step1) medium
(SYSTEM_CONFIDENTLY_SUGGEST_CREATING_KITEM PS-step1) high
...
Opposing episodes:
(NO_APPLICABLE_KITEM_AND_EXE_SUCCEED PS-step1) low
...
When to remove instantiation:
(APPLICABLE_KITEM_CREATED PS-step k_item1)

MP1-I1: Assess need of new k-item for
PS-step: deciding Mary's meeting time
Supporting episodes:
NO_APPLICABLE_KITEM_AND_EXE_FAIL *step-12;;* deciding Mary's time for Ken's visit
NO_APPLICABLE_KITEM_AND_EXE_FAIL *step-23;;* deciding Mary's time for Tom's visit
...

(a) Meta-level pattern for assessing need of new k-item

MP2: Assess how to create/modify k-item (PS-step1)
Suggestion: (SUGGEST_TO_CREATE_SIMILAR_DEFINITION PS-step1 k-item1)
When to instantiate pattern: (SUGGEST_TO_CREATE_KITEM PS-step)
When to produce suggestion: (SUGGEST_TO_CREATE_KITEM PS-step1) (SUGG..)
Supporting episodes:
(KITEM_APPLIED_AND_EXE_SUCCEED k-item1 PS-step1) medium
(USER_FOLLOW_SUGG_TO_CREATE_SIMIALR_KITEM k-item1 PS-step1) high
...
Opposing episodes:
(KITEM_APPLIED_AND_EXE_FAILED k-item1 PS-step1) medium
...

MP2-I3: Assess how to create k-item for Mary's meeting time
Supporting episodes: none

MP2-I1: Assess how to create k-item for Gary's meeting time
KITEM_CREATED *k-item-3* in deciding: (<a> when ?) : (<a> what meeting) (<a> who Gary)
if (<a> what meeting) (<a> who Gary) choose: {5pm, 5:30pm} "to avoid cancellation"
Supporting episodes:
KITEM_APPLIED_AND_EXE_SUCCEED *k-item-3 step-12*
USER_FOLLOW_SUGG_TO_CREATE_SIMIALR_DEFINITION *k-item-3* …

(b) Meta-level pattern for assessing how to create or modify a k-item

Figure 1: Meta-level patterns for improving suggestions on k-item creation

Figure 1-(a) and (b) show a pattern for assessing the confidence level a k-item creation and a pattern for generating additional suggestion on how to create the k-item respectively. The set of patterns used in Echo is shown in [1]. Given a set of meta-level patterns Echo follows the steps shown in Figure 2.

1. For each new event *e*,
   - identify episodes with *e* and prior events
2. For each new episode *es*,
   - For each meta-level pattern where *es* matches with '*when to instantiate pattern*', create a new instantiation
   - For each pattern instantiation where *es* matches with its '*when to remove instantiation*', remove matching instantiations
   - For each pattern instantiation where *es* matches with its '*supporting episodes*', add it to the corresponding list of supporting episodes
   - For each pattern instantiation where *es* matches with its '*opposing episodes*', add it to the corresponding list of opposing episodes
   - For each pattern instantiation whose '*when to produce suggestion*' matches with es,
     - compute the confidence level
       - Compute weighted difference of supporting and opposing episodes based on their significance level and degree of repetition
       - Combine confidence level of similar suggestions
     - If the confidence level is above θ (a defined value), present
       - Suggestion
       - Confidence level
       - Explanation: supporting and opposing episodes
         Suggestions from similar instantiations

Figure 2: The Echo procedure.

PRELIMINARY RESULTS

We performed a preliminary evaluation of Echo with a set of synthetic scenarios with varying user interactions and mistakes. We compared two KA systems where both of them use the same episodes of problem solving and problem changes but one of them was enhanced with Echo's reflection patterns. The results show that with Echo, the KA system can reduce the number of incorrect suggestions and the number of problem solving failures.

Average per scenario	With meta-level patterns	Without meta-level patterns
Avg # decisions assisted with k-items(# total decisions)	24.0(84.0)	22.03(82.03)
Avg # wrong user decisions	8	8
Avg # additional user mistakes	2.73	5.97
Avg # k-items creation or modification	4.87	4.34
Avg # sugg (Avg # confident sugg)	13.83 (10.27)	13.0 (0)
Avg # wrong sugg	0.90	3.07
Avg # failures	4.10	6.73

Table 2: Results from initial analysis

REFERENCES

[1] Kim, J. Reflection Patterns for Interactive Knowledge Capture,http://www.isi.edu/~jihie/papers/echo-RP.pdf.

[2] Kim, J. and Gil, Y., Acquiring Problem-Solving Knowledge from End Users: Putting Interdependency Models to the Test. *Proceedings of AAAI-2000, 2000.*