Acquisition of New Knowledge In TutorJ

Giuseppe Russo, Arianna Pipitone and Roberto Pirrone
Università di Palermo - Dipartimento di Ingegneria Informatica - DINFO
Viale delle Scienze Ed. 6 p. 3 - Palermo (Pa) Italy 90128
{russo, pipitone, pirrone} @dinfo.unipa.it

Abstract
This paper presents a methodology to acquire new knowledge in TutorJ using external information sources. TutorJ is an ITS whose architecture is inspired to the HIPM cognitive model, while meta-cognition principles have been used to design the knowledge acquisition process. The system behavior is intended to increase its own knowledge as a consequence of the interaction with users. The implemented methodology uses external links and services to capture new knowledge from contents related to discussion topics and transforms these contents into structured knowledge that is stored inside an ontology. The purpose of the proposed methodology is to lower the effort of system scaffolding creation and to increase the level of interaction with users. The focus is on self-regulated learners while meta-cognitive strategies have to be defined to adapt and to increase the effectiveness of tutoring actions.

Introduction
This paper presents the latest developments of TutorJ (Pirrone, Rizzo, and Pilato 2008) towards the use of meta-cognitive strategies in supporting learners. In particular, a new methodology for knowledge acquisition from external information sources is presented that is triggered by user interaction. TutorJ is an Intelligent Tutoring System able to interact with users to assess their skills, and to improve their knowledge in a particular domain. The paper is focused primarily on the definition of procedures that are able to add new items to the system’s knowledge base. This module has two major components: a static one consisting of an ontology able to describe the concepts involved in the domain, and a set of procedures that are organized with the purpose to extend the ontology via integration of external informative sources.

As previously stated, the knowledge base is organized according to an ontology. This ontology has is purposed to define and to connect the main concepts of the domain. A knowledge base that doesn’t evolve during time has major limitations: the system is unable to reply to questions that are outside the ontology boundaries and is not capable to increase its own knowledge with respect to the interaction with users or to add new external information sources. To overcome these limitations we are developing a set of tools and strategies starting from cognitive and meta-cognitive perspectives. The interactions between users and the system have to be designed with continue feedbacks able to increase the system knowledge while it is required. Cognitive and meta-cognitive strategies are primarily focused on the definition of the so-called self-regulated learning process (Zimmermann and Schunk 1998). The main characteristics of self-regulated students are (Zimmermann 2001) (Montalvo and Gonzales-Torres 2004):

- The ability to perform cognitive strategies to deal with the information size growth.
- The ability to adopt time and effort strategies in order to optimize learning results.
- A high level of curiosity and motivation for novelties.
- A commitment to result’s achievement.
- The ability to perform meta-cognitive processes.

Pintrich (Pintrich 2000) defines Self Regulated Learning (SRL) as a process with three categories of strategies that students should have access:

- Cognitive learning strategies
- Meta-cognitive and regulation strategies
- Resource management strategies

A software agent inspired to SRL has to provide features in all the three mentioned categories. The presentation of materials and documents has to be simple and intuitive to support cognitive learning strategies with the possibility to personalize learning paths and to assess the level of users’ knowledge automatically. The meta-cognitive and regulation strategies have to be supported with the capability of the system to predict student performances and to modify its strategies according to a series of relevant indicators. Resource management strategies are a crucial point in the development of systems able to deal with SRL students, and this is the central issue of the presented work. Learning resources have to be always up-to-date, and they have to be tailored to the user queries. In general, this requires a huge system scaffolding activity that has to be done manually. A
SRL tutoring agent has to behave to acquire new knowledge autonomously in response to the user needs. Finally, the ability to detect and to adjust to students uncertainty is an important prerequisite for a system able to interact with self-regulated students. Many promising studies are shown that a system able to detect student uncertainty (Forbes-Riley, Litman, and Rotaru 2008) (D’Mello et al. 2008) has a higher level of assessment and evaluation of correctness of the students’ replies.

The rest of the paper is arranged as follows: next section briefly describes the architecture of TutorJ. The third section presents theoretical backgrounds to build more effective systems in the defined perspective. The forth section presents the procedure to add new knowledge in the system. Finally, some conclusions and future works are reported.

**TutorJ Architecture**

TutorJ is an Intelligent Tutoring System (ITS) that was primarily intended for supporting learning of the Java programming language. The latest release of TutorJ deals with the Arts domain. A traditional ITS architecture is defined in terms of some cooperating modules (Wolff et al. 1998), (Rickel 1989). The main components are the expert model, the student model, the instructional model, and the learning environment. Traditional approaches to e-learning are based on the definition of the most suitable environment for the students (Piramuthu 2005). Moreover, the design of the student model, the expert model and the instructional model are necessary. The expert module is able to deal with system’s knowledge base to provide corrections, hints and material for students. The student model performs a comparison student’s abilities and skills to the expert model. The instructional model is used to present new knowledge as multiple choices questions, fill-in-the-blank exercises or problems to be solved. Our system has been realized as a cognitive system according to the Human Information Processor Model (HIPM) (Todorovski et al. 2005) that has been developed to describe human cognitive processes. A cognitive architecture is composed by a series of modules. Each module is devoted to a specific function involved in the cognitive processes. There are three main kinds of modules: perceptual module, cognitive module and motor/sensor module. The Fig. 1 illustrates a simplified version of the overall architecture.

Each module in TutorJ deals with a particular task. Sometimes a complex task is accomplished by several modules together. Common tasks are natural language interaction, learning contents management, planning of a learning path on the concept map interface, and acquisition of new knowledge.

The knowledge base manages a symbolic memory regarding the learned domain as an OWL ontology. The LSA module (Landauer, Foltz, and Laham 1998) manages a sub-symbolic memory that merges documents, concepts and terms in a common space thus providing an integrated representation of all the domain. It codes the elements with similar meaning as points that are close in the space. The WordNet module is used as an external resource and stores a similar knowledge represented in a symbolic form used for disambiguation purposes. The system owns also a working memory inside the Chatbot module. This memory stores stimuli from the environment. In turn, stimuli trigger the cognitive module, which is implemented as goal-oriented extension of the chatbot engine, and evaluates the actions to be executed. The working memory stores also the status of the single conversation and of the whole system (conversation mode, supplied materials, gathering of new knowledge). The chatbot interface and the map interact with the student using natural language and graphic elements.

**Theoretical Backgrounds**

Two main problems have to be faced to increase effectively a structured knowledge base using external information sources. At first, semi-structured knowledge sources (i.e. a wiki) have to be transformed into a structured form (an ontology-like structure). Then the new structured knowledge has to be aligned to the existing knowledge base to merge them together.

In our implementation we decided to use wikis as external information sources due to their wide use over the Internet for publishing structured information. Nowadays Wikipedia is a huge knowledge base with more than 3 millions of articles and a growth rate of 24% in 2009 (sta 2009). Our attention turned to the so called *semantic wikis* and to the well known problem of ontology alignment based on the use of semantic similarity between nodes of the knowledge bases to be merged. In what follows a brief review of these two topics is reported.

**Semantic Wikis**

Ontology instances and concepts directly associated to a wiki page are the so-called Semantic Wikis (Völkel et al. 2006) (Kawamoto, Kitamura, and Tijerino 2006) (Quan, Yu-ying, and Jing 2008). In this way wiki links and annotations are related the concepts while pages are related with each other. The navigation of the ontology and the definition of rules allows to obtain more accurate matching and better documents and pages retrieval. Many systems have been
presented in literature that allow to automatic relate a web page to an ontology. This task is performed with an annotation process. The main steps of this process are the selection of the appropriate part of the wiki page, the correlation with the concepts and the disambiguation of candidate from a thesaurus. The work presented in (Ahn, Jung, and Choi 2008) automatically annotates language resources online by referring to the semantic correspondences obtained from ontology mapping methods. Another interesting approach is presented in (Jung et al. 2006) that proposes a system that helps people in a community collaboratively annotate any web pages. Users are allowed to manually select a portion of text in a web page and annotate it using some metadata vocabulary. The collaboration of different users makes the time and computational efforts lower.

**Semantic Similarity**

The definition of a semantic similarity measure to correlate different portions of ontologies is crucial to obtain systems able to expand their knowledge base, and to retrieve different documents related to concepts via annotation (Rissland 2006). As stated in (Goodman 1972) there are not application independent laws to obtain an objective similarity measure. The selection of a particular semantic similar measure and the comparison with pre-existing ones are difficult, too. In the work of (Janowicz and Wilkes 2009) seven steps to define a similarity measure have been defined:

1. Definition of utilization area and intended audience
2. Selection of search (query) and target concepts
3. Concepts transformation to a canonical form
4. Definition of an alignment matrix for concept descriptors
5. Application of constructor specific similarity functions
6. Determination of standardized overall similarity
7. Interpretation of the resulting similarity values

Many approaches have been proposed in literature to define a measure for semantic similarity between concepts. There are four fundamental categories of measures: Lexical, Topological, Extensional and Model-based (Falconer, Noy, and Storey 2007). The first group of measures is based on the lexical properties of the involved concepts, while the second group is based on the topological (often called structural) distance between concepts in the ontology graph. Extensional measures make use of the information content, while model-based ones use ad-hoc models that are obtained often as a combination of the other approaches.

Ontology alignment or ontology matching is the set of operations that are applied to two separate knowledge bases to define a multiple correspondence between some elements in the first ontology to other elements in the second one. The reader is referred to (Euzenat and Schvaiko 2007) for a complete survey.

More formally an alignment between an ontology \( O \) and a second ontology \( O' \) is the quadruple \( < e, e', s, n > \) where \( e \in O \) is an element of the first ontology, \( e' \in O' \) is an element of the second one, \( s \) is the relation between elements that defines the similarity and \( n \in [0..1] \) is the level of confidence (or strength) of the relation.

**Acquisition of New Knowledge**

Our knowledge acquisition process relies on meta-cognitive strategies. In particular, it has been designed according to Pintrich’s definition of “Feeling of Knowing” (FOK). When receiving a negative feedback about the presentation of a concept, the system enables a procedure to find new knowledge related to the topic under investigation. This is a typical self-regulation mechanism. As an example, a user could have interest for a specific knowledge item that is partially explained in the knowledge base. Our system may have or not a partial knowledge of the query domain coded in the inner ontology. In both cases TutorJ can enrich the ontology using a mapping process. As an example, new knowledge can be inferred on the fly using a semantic wiki on the Web. As we said before, our approach makes use of wikis as a primary source of knowledge. Moreover, wikis have a standard page structure so it is easy to extract relevant information through a spider. Our approach incorporates wiki pages into the internal document repository, while increasing symbolic knowledge by adding new nodes to the ontology.

The macro-steps of the procedure are (see Fig. 2):

1. User requests are inferred from the dialogue performed through the chatbot interface.
2. On-line services are used to retrieve knowledge about a low FOK topic.
3. New knowledge is mapped into the domain ontology.
4. Results are shown to the user.

![Figure 2: Interaction loop of the knowledge acquisition process](image)

The procedure is executed iteratively until the student feedback becomes positive so the focus of conversation is moved to another concept.

Here follow the main parameters involved in the process:
• \(D\): the domain ontology (represented as an OWL file (McGuinness and van Harmelen 2004)).
• \(N\): the number of concepts in \(D\) (number of classes).
• \(f\): the first level concept of \(D\) (a direct subclass of \(\text{owl:Thing}\) that is the ontology’s root).
• \(x\): a generic concept.
• \(O\): a generic object property.
• \(G\): a sub-graph of \(D\).
• \(S\): the on-line service that exports a document page of a semantic wiki in OWL-DL format.
• \(D_{xy}\): the distance between concept \(x\) and \(y\) measured according to a similarity measure.
• \(T\): the threshold for similarity measure. \(x\) and \(y\) are similar if \(D_{xy} < T\).
• \(V_c\): the selected concepts vector, its components are interesting concepts selected by the user indirectly.
• \(M\): the dimension of \(V_c\).
• \(V_u\): the URL vector, its components are the ontology URLs of the wiki documents exported by \(S\).
• \(M_d\): the distance matrix, a \(N \times M\) matrix used when concepts belonging to \(V_c\) have to be retrieved in \(D\).

The Chatbot module in TutorJ manages the dialogue with the student, and copes also with uncertainty in the conversation. It is able to assess the user, and to put in \(V_c\) interesting concepts that are extracted from her most relevant sentences. Each component of this vector corresponds to a candidate document. Then the procedure calls \(S\) and for each component \(c\) in \(V_c\) evaluates the existence of the related document in an external wiki. If the document exists, an ad-hoc service function of \(S\) is used to retrieve new documents in OWL format to extend \(D\). In our experiments we used the on line service ExportRDF (http://semanticweb.org/wiki/Special:ExportRDF) that is provided by Semantic Media Wiki (http://semanticweb.org/wiki/Semantic_MediaWiki). The obtained URL is inserted in \(V_u\). If the documents don’t exist in the wiki, we can’t extend \(D\) but we must retrieve information in \(D\) about the topic. This is obtained through the computation of the similarity distance \(D_{xy}\) between whatever concept \(x \in D\) and \(c\). If \(D_{xc} \leq T\) it is recorded at cell \((x,c)\) in \(M_d\). Otherwise, a standard marker value is stored in the cell to prevent that it can be used in subsequent browsing. Once all concepts have been browsed the concept \(x'\) whose cell stores the lowest value in \(M_d\) is selected. The corresponding sub-graph \(G\) is selected as the one centered in the node \(x'\). If \(V_u\) owns at least one component we achieve an extension of \(D\). Finally, \(c\) is mapped to \(x\) to extend \(D\). Three types of extension are possible:

• \(c\) already exists in \(D\), and it is the same as \(x'\); the procedure extends \(D\) directly adding \(c\) datatype and object properties to \(x'\).
• \(c\) doesn’t exist in \(D\), and has a similarity distance \(D_{xc} \leq T\); the procedure links \(c\) to \(x'\) via a suitable relation. If the relation is exactly \(<\text{RDF:InstanceOf}>\) then all the slots in \(c\) are filled using properties in \(x'\).
• \(c\) doesn’t exist in \(D\) and a new class is created as an \(f\) concept.

Application Scenario

The rules to add new knowledge to our system are created using JessTab (Jes ) (Friedman-Hill 2003), a plugin for Protégé that permits to use Jess and Protégé together. In this way, we can browse an ontology using either rules or Sparql (Prud’hommeaux and Seaborne 2006) queries and we can expand the domain ontology using JessTab functionalities directly. Sparql is useful to execute queries on OWL/RDF files, so we are able to browse remote ontologies that are the feeds of on line services of the semantic wikis. To this aim we have created an interface using Jena APIs (Carroll et al. 2004), which allows us to execute Sparql queries from JessTab directly.

In the following, a simple application scenario for our Arts ontology is presented. A snapshot of the ontology is reported in figure 3.

Let’s suppose the user wants to know something about a contemporary artist that could not be in our database because he is not very famous worldwide yet. As an example, we selected the Italian painter Roberto Fantini. We start from the artist’s presence personal page on Semantic Media Wiki. Once TutorJ knows the the user wants to know the only topic related to Roberto Fantini, \(V_c\) is filled, and has only one component that is the string containing the name of this artist.

Then system calls repeatedly the on-line services of the known semantic wikis, and puts different versions of \(V_c\) components (tiny, capital and so on) as input from time to time. The resulting ontology extracted by the ExportRDF service in Semantic Media Wiki is depicted in figure 4.

![Figure 4: The portion of the ontology exported from the service](image-url)
Figure 3: A snapshot of the domain ontology

| Contains the name of the artist |
| All-format-of is our function |
| Input parameter a string |
| Output all format of input |
| String |
| (bind ?topic (all-format-of Roberto_Fantini)) |
| (foreach ?string ?topic (find-all-instances ((?ist owl:Thing)) (eq (slot-get ?ist rdf:ID) ?string))) |

So, the system matches the RDF fragment from the semantic wiki to its domain ontology. Sparql is used to browse the semantic wiki, while JessTab functions are used to connect the concept to the domain ontology, and to fill the slots. The following query selects all of the Object Properties of the artist.

| Load function to execute Sparql query |
| From JessTab |
| SparqlInterfaceForJess |
| Execute query |
| (bind ?objprops (sparqlInterface ?query)) |
| Make-instance ?prop of owl:ObjectProperty |
| (rdfs:domain Frattini) |
| (rdfs:range ?range) |

The following JessTab functions map it in the domain ontology. In this case, the new concept is an instance of the general concept <Artist> in the domain ontology.

| Create instance of new artist |
| Make-instance Fantini of Artist |

Conclusion and Future Works

In this paper a methodology to acquire new knowledge inside the TutorJ's domain ontology has been presented. Our approach is inspired by meta-cognition principles, and in particular a self-regulation mechanism has been used that is the control of the student’s FOK. The methodology allows to extract information from semi-structured sources such as a wiki page and to transform it into structured knowledge to be re-used also for other learners. The implemented procedure is fully embedded inside TutorJ as a part of the knowledge base. Otherwise, we are not constrained by a particular domain. The current release of TutorJ has discarded Java, and it’s devoted to Arts. The purpose is to design a framework for systems able to deal with self regulated students allowing them to use metacognitive strategies in the learning process. We are currently tuning a general service able to
cope with Wikipedia directly. Finally, we’re defining a more suited similarity measure than a simple hybrid of lexical and topological approaches.

References


