iLOG: an Artificial Intelligence Framework for Automatic Metadata Generation for Online Learning Objects

L.D. Miller, Computer Science and Engineering, University of Nebraska-Lincoln, USA
Leen-Kiat Soh, Computer Science and Engineering, University of Nebraska-Lincoln, USA
Ashok Samal, Computer Science and Engineering, University of Nebraska-Lincoln, USA
Gwen Nugent, Center for Children, Youth, Families & Schools, University of Nebraska-Lincoln, USA

Abstract.

We present a framework for the automatic tagging of learning objects (LOs) with empirical usage metadata. The framework involves real-time tracking of each user sessions (an LO wrapper), offline data mining to identify key attributes or patterns on how the LOs have been used as well as characteristics of the users (MetaGen), and the selection of these findings as metadata. Mechanisms used in the data mining include data imputation, association rule mining, and feature selection via an ensemble of clustering algorithms. This paper describes the methodology of the automation in meta-tagging, presents the results on the evaluation and validation of the algorithms, and discusses the metadata found and the implications of such in improving student pedagogy. Our implementation of the Intelligent Learning Object Guide (iLOG) was used to collect interaction data of over 200 students’ interactions with eight LOs in introductory computer science topics. We show that iLOG successfully tracks student interactions that can be used to automate the creation of meaningful empirical usage metadata using real-world usage and student outcomes.

Keywords. Learning Objects, Empirical Usage Metadata, Association Rule Mining, Feature Selection, Data Imputation, SCORM

INTRODUCTION

The traditional classroom approach (i.e., textbook and lecture hall) where an instructor presents content to many students simultaneously has several significant problems. First, it requires the physical presence of students. This is impractical for students in the Third world where students have difficulty attending class for a variety of reasons (e.g., lack of public transport, safety concerns due to conflict, etc.). It is also impractical for many students in developed countries who have schedules that conflict with traditional classroom hours (e.g., students who are working full time jobs). Second, this approach severely limits the amount of student interactions with the instructor. For example, the instructor cannot pause to answer every question during a course in a lecture hall with 300 students. This one-way presentation of content is a less effective pedagogical technique than allowing for increased student interactions. Third, this approach requires that an instructor be both an expert on the content and effective in presenting it to the students. For many content areas (e.g., advanced mathematics, physics, etc.) this combination makes it difficult to effectively present content to the students. Regardless, online education programs (e.g., Phoenix University) have expanded rapidly to address the first problem with traditional classroom approach. In online programs, students are able to
attend lectures, take exams and submit assignments all electronically. Even traditional education programs have moved rapidly to provide similar services (e.g., University of Nebraska’s Extended Education). However, such online programs do not address the second and third problems with the traditional classroom approach. Attending lectures electronically does not automatically allow for increased student interactions. Further, online education program still requires instructors that are capable of effective content presentation.

Learning objects are one commonly used approach to address these two problems. LOs are small, self-contained lessons designed to provide instruction on specific content. One of the most commonly used standards is the Sharable Content Object Reference Model (SCORM) developed by the Advanced Distributed Learning (ADL) initiative (ADL, 2003). LOs commonly consist of three components: (1) tutorial, (2) practice, and (3) assessment. The practice component contains interactive examples based on the content. Working through such examples facilitates increased student interactions by providing immediate feedback. Thus, an obvious approach is to augment traditional or online education programs with self-contained LOs relevant to segments of content the instructor is presenting. This could be done by selecting LOs from online repositories (e.g., MERLOT (www.merlot.org), Maricopa (mcli.maricopa.edu), Careo (www.ucalgary.ca/commons/careo), etc.). Such an approach will result in increased student-instructor interactions. However, the use of LOs does automatically address the third problem of effective content presentation. Instructors must be capable of finding suitable LOs that effectively present the content without searching through the large proliferation of LOs, potentially relevant to the content, in an online repository.

In addition to automating metadata creation, there is also need to customize it for different users (Ravasio, 2003; Kosba, 2007). Ravasio (2003) discusses the need for teacher- and student-friendly metadata. Metadata should support the creators of learning objects and also support users trying to find and access them. According to Kosba (2007), different kinds of automatic feedback are required for instructors and users in an online course. Instructors are the facilitators and need to know about user interactions with the LOs. In particular, instructors should be aware of any problems that users are having with LOs. On the other hand, users often require personalized help on the LO content. As a result, metadata created for users should reduce their feeling of isolation, while metadata for instructors should reduce their communication overhead. The iLOG metadata created from user interactions can be used for both purposes. For users, the metadata emphasizes what interactions are needed to succeed in the course. For instructors, the metadata also identifies pitfalls encountered by previous students. In the Results section we discuss how iLOG metadata supports custom metadata for teachers and students.

The SCORM standard for LOs specifies a set of metadata designed to help instructors select suitable LOs in a repository. Unfortunately, at this time, such metadata is optional and the search and usage results are often highly subjective. This makes it very difficult for an instructor to search for suitable LOs without personally reviewing each LO. It also requires that instructor be capable of vetting whether the LO designers are effectively presenting the content. Currently, this combination is a roadblock to using LOs to augment traditional or online education courses. The Intelligent Learning Object Guide (iLOG) framework proposed and described in this paper removes this roadblock by using Artificial Intelligence (AI) to automatically generate metadata for the LOs in a repository. Such metadata is based on actual user interactions with the LOs rather than being subjectively provided by the LO designers. Thus, it provides an objective metric for searching for suitable LOs and reduces the need for personally vetting the LOs before they can be included in the course.
Additionally, such metadata would be useful for many other groups including (1) students, (2) LO designers and (3) researchers. First, metadata from previous semesters could be used as guidelines for success in the course for students. For example, the metadata could indicate that previous students who did not go through the exercises had a much higher tendency to fail the LO. Further, the metadata could be used to generate prerequisites with a much finer level of granularity than usually provided for courses. For example, MetaGen, the main component of the automatic tagging of iLOG, discovered that students with low motivation towards computer science study and no Calculus experience had a higher tendency to fail on content in introductory computer science (CS) courses (Riley et al., 2008). Second, metadata could be used by LO designers to determine whether the LO is effectively presenting content to the students. For example, MetaGen discovered that female students tended to not do well in the assessment component of the LOs in introductory CS courses (Riley et al., 2008). Obviously, this suggests a future improvement to individual learning objects. Third, for researchers the metadata provides high level summaries useful for initial exploration, while the iLOG framework stores all user interactions with the content for supporting in-depth analysis (e.g., data mining).

In the following, we first present related work to our research in terms of metadata and automation of generating metadata for learning objects. We also look more specifically at metadata derived from association rules mined from data and summarize the relationship between metadata and the LO repository. Second, we describe in detail our methodology for iLOG, focusing on the automation: from tracking to metadata generation. On tracking, we present the module called the LO Wrapper, that can be embedded with standard learning management systems (LMSs) that administer SCORM-compliant LOs. On metadata generation, we present MetaGen and its three key steps: data imputation, feature selection ensemble, and association rule mining. Third, in the Results section, we provide two discussions, one on the validation of the algorithms used in the automation, and the other on the validation of the metadata from a student pedagogical perspective. Finally, we conclude with future work.

Before we continue, note that previously, we created several LOs on introductory computer science (CS) concepts. Each of these LOs contained (1) a tutorial, (2) exercises and (3) assessment components consistent with the organizational guidelines given in Thompson (2005). Preliminary work for automatic metadata creation from user interactions collected from these LOs was encouraging (Riley et al., 2009). The iLOG system prototype was able to create metadata with both high confidence for each of the learning objects considered. From these results, we were able to gain useful insights for the usage properties for different types of students. Further, Nugent et al. (2009) used the iLOG system prototype to evaluate real learning goals, specifically, the impact of active learning and elaborative feedback on student learning. In this work, we have expanded the iLOG system with many refinements for automatic metadata creation. We also provide a rigorous validation for the metadata created using iLOG. The iLOG metadata is designed based on three metrics for learning object metadata (Ochoa, 2008). The metadata is complete because iLOG considers all relevant attributes when creating association rules from user interactions. Second, the metadata is accurate because iLOG provides the confidence values for all the metadata. Third, the metadata has provenance because iLOG system updates the metadata on existing LOs based on new user interactions.
RELATED WORK

In this section, we first provide existing work on learning object (LO) metadata standards. Second, we discuss two different approaches for automatic metadata creation based on (1) the LO content and (2) user interactions with the LO. The iLOG framework adopts the second approach. Finally, we describe existing work on automating LO selection from repositories.

Learning Object Metadata

There has been a considerable interest involving metadata for LOs. Several standards for specifying what metadata to include with LOs have been created; the most commonly used is the IEEE Learning Object Metadata (LOM) Standard (IEEE, 2002). Friesen (2004) provides a brief description for the organization of metadata consistent with the LOM standard. The IEEE LOM Standard for metadata is the most widely accepted, but it is far from perfect. First, it lacks metadata on the quality of learning objects as judged by the users (Vargo, 2003). Second, according to Polsani (2003), one of the functional requirements for LOs is accessibility. This is done by tagging the LO with metadata so it can be stored and referenced in a repository. However, current metadata standards do not require content developer to provide all the metadata. This often leads to omitted metadata that minimizes accessibility for the LOs. Friesen (2004) conducted an international survey on the implementation of the LOM standard and found that much of the metadata was not provided by human users making it difficult to incorporate into the LO design and meta-tagging processes. Finally, Cardinaels (2006) discussed the need for confidence in the precision or accuracy of metadata for a specific LO and for context-aware metadata. Recently, a study was conducted on human-generated, metadata using the IEEE LOM standard (Cechinel, 2009) in which students used an annotation tool to enter the metadata. The results show a high percentage error on entering the correct metadata—as much as 25% on some metadata sections—despite years of refinement to metadata standards and annotation tools. Thus, even with the LOM standard, there is still a need for automating the creation of metadata. Ultimately, automatic metadata creation is more efficient, less costly and more-consistent than human processing (Roy, 2008).

An alternative to automatic metadata creation is forcing the developers to provide the metadata. Bailey (2006) discusses learning activity nuggets which contain specific elements such as subject area, level of difficulty, prerequisite skills, environment, etc. Nuggets have a set of metadata which is automatically populated when a nugget is created using the online editor. However, this approach does not follow any standard and is unlikely to be widely adopted.

Automating Metadata Creation

There has been considerable work in the last ten years involving automating metadata creation for LOs. The first common approach focuses on the LO content (Cardinaels, 2005; Luciano, 2005; Gasevic, 2005, Brooks, 2006; Javanovic, 2006; Saini, 2006; Zouaq 2007a; Zouaq 2007b; Roy 2008). This approach first populates an ontology using the content and then creates metadata from the ontology. The main advantage for ontology (Javanovic, 2006) is convenience in searching through LO repositories. Semantic web reasoning could search for LOs with content of a certain type using context ontology, dealing with a certain topic using a domain ontology, or with a certain level of granularity using a structural ontology. The ontologies used in this approach are either (1) provided
by the content developer (Gasevic, 2005; Javanovic, 2006; Saini, 2006) or (2) populated using natural language processing to extract keywords, word dependencies, and concepts from the LO content (Luciano, 2005; Brooks, 2006; Zouaq, 2007a; Zouaq, 2007b; Roy, 2008).

Cardinaels (2005) discusses an automatic indexing framework which generates metadata for SCORM-based LOs. This framework consists of two components. First, the context-based indexer creates metadata based on when the LO is used. Second, the object-based indexer creates metadata based on organization of the learning object (e.g., types of files included). In a case study this framework was able to automatically generate metadata fields included with SCORM. However, these fields were limited to metadata about the structure of the LO.

TANGRAM (Javanovic 2006) uses an ontology approach for the metadata. It employs structural and context ontologies for storing the content for the learning objects. It also maintains ontologies for learner paths and user models. The user models are created from initial questionnaires. TANGRAM allows a content developer to upload new LOs to a repository, automatically tagging them with high-quality metadata, search the LO repository, and compose a new LO using components from existing LOs. However, the majority of the metadata required for annotating the LO must first be supplied manually by the content author. After this is done, TANGRAM automatically annotates the LO components and integrates the LO into the repository. This annotation is consistent with IEEE LOM standard. The main difference between TANGRAM and iLOG is that, in TANGRAM, user interactions play no part in the automatic creation of the metadata. Initially, the metadata is created based on sample metadata supplied by the developer. Subsequent user interactions with the LO are stored on a separate ontology and never used to revise the metadata.

Roy (2008) uses an algorithm to identify concepts in the text for learning objects in three different subjects (physics, biology and geography). The algorithm distinguishes between outcome concepts necessary for the learning goal and prerequisite concepts which the user must understand before the outcome concepts. Concepts are extracted using a shallow parsing approach for identifying verbs used in definitions (e.g., defined, derived, called, etc.). The algorithm uses a three-layered hierarchical knowledge base. First, the term layer stores lexical terms (i.e., keywords). Second, the concept ontology contains the relationships between domain-specific concepts. Third, the topic layer organizes the concepts, discussed for each topic, based on the learning requirements for the institution. The algorithm uncovered many of the same concepts that were also manually observed by human experts. The automatic annotation tool adds these concepts in a machine comprehensible format compliant with the IEEE LOM standard. This algorithm computes the metadata using only the LO content whereas iLOG uses both the LO content and user interactions to compute the metadata.

The iHelp system (Brooks, 2006) provides a keyword- and concept-based metadata extractor. The keyword-based extractor uses natural language processing to select the most relevant keywords and sentences in the LO content. The concept-based extractor uses a conceptual ontological graph to organize sentences into a hierarchical representation of concepts.

Finally, Saini (2006) provides an algorithm for the automatic classification of the LO into an ontology based on the LO content. This method uses a semi-supervised algorithm based on expectation maximization (EM) where the keywords available in the ontology are used for bootstrapping the classification of the LOs.

In summary, the novelty of our iLOG framework is the type of metadata considered, one that is derived empirically, as opposed to being authored, from the usage data of the LOs—that is, in terms of how each LO has been used. This type of metadata can provide additional insights into the effectiveness of an LO in relation to student characteristics. Further, this type of metadata is to a large
extent domain- or subject-independent, as shown later in our Methodology section. This property has the potential of allowing LOs of different topics, or student users of different topics, be studied and evaluated more systematically.

**Metadata from Association Rules**

It should be noted that there are indeed reported research works that make use of the user interactions with the LOs (Bigdoli, 2004; Etchells, 2006; Castro, 2007; Wolpers, 2006; García, 2009a; García, 2009b; Segura, 2009; Liu; 2009). The general underlying paradigm is mining the stored user interactions into suitable metadata (Castro 2007). In particular, association rule miners are often used for automatic metadata creation (Bigdoli, 2004; García, 2009a; García, 2009b; Segura, 2009) because they provide human-comprehensible metadata and an evaluation metric. However, other algorithms have been tried including Bayesian belief networks (Liu, 2009) and artificial neural networks (Etchells, 2006). Here we review these algorithms briefly and distinguish them from iLOG.

The CAM framework (Wolpers, 2007) intercepts user interactions with many applications such as the web browser. These user interactions are converted into metadata which are stored on an external database. This approach is very similar to the wrapper described for the iLOG system. Both intercept user interactions and send them to an external database. In CAM, the transmission is one way because metadata never leaves the database. However, in iLOG the transmission is two-way. The metadata computed by iLOG from user interactions is sent back the LO repository making it available to both users and instructors.

The Learning Online Network with Computer-Assisted Personalized Approach (LON-CAPA) (Bigdoli, 2004) employs association rule mining to describe user interactions with online course work. The mining contrast rules (MCR) algorithm in LON-CAPA computes a set of conjunctive, contrast rules based on (1) student attributes, such as GPA and gender, (2) problem attributes, such as the assignment difficulty, and (3) student/problem interactions, such as the number of attempts and time spent on the assessment. The MCR computes association rules based on whether students pass/fail and each rule has both a rule support and confidence value associated with it. This is very similar to the association rule miner component in the iLOG system. However, there is no provision in MCR for replacing missing values. Further, MCR assumes that all attributes are potentially relevant to the association rules. Thus, MCR requires hand-tuning to avoid being swamped with less interesting rules based on irrelevant attributes. The imputation and feature selection components in the iLOG system handle both eventualities.

Garcia (2009a) uses the Apriori algorithm for association rule mining on user interactions with e-learning courses. This algorithm provides recommendations to the instructors based on the association rules. It employs reinforcement learning based on instructor responses and expert evaluation. Unlike iLOG, this system requires the active assistance of the instructor during metadata generation.

Sugura (2009) combined the clustering technique with association rule mining on LOs from several repositories. First, this method clusters all the LOs based on the LOM metadata included in each LO (i.e., metadata used as attributes for clustering). Second, it used the Apriori association rule miner, separately, on the metadata in each cluster. This method is similar to the imputation component in iLOG. Both use clustering algorithm to create partitions where missing attribute-values can be filled in from similar values. However, the iLOG imputation uses a more complex combined, hierarchical approach than the K-Means clustering algorithm in this study to address the large amount of missing values and noise found in the data.
The Eliminating and Optimized (EOS) Selection algorithm (Liu, 2009) is designed to select a suitable set of LOs for users from a repository. EOS uses a Bayesian Belief network to compare the user attributes collected from the survey data with attributes for the learning objects. The user attributes include gender, year of student, major, reading level, etc. The learning object attributes include pedagogical objective, environment, expected reading level, etc. The network is trained on the collected survey data and LO attributes subjectively specified for each LO. The network computes a different weight for each combination of user and LO attributes. These combinations are metadata used to select LOs for each user based on the specific user attributes from the survey. Both EOS and iLOG select which attributes are relevant for the metadata. EOS considers combinations in the network with a significantly high weight, while iLOG employs feature selection to choose the subset of attributes most relevant to the assessment. The LO attributes for EOS incorporate some aggregate information from previous users (e.g., duration the LO access, the number of help requests, assessment result for the user, etc.). However, the emphasis is on the survey results from the user and the aggregate information is not automatically collected. The iLOG system also employs surveys, but there is a greater emphasis on evaluating individual, user interactions which are automatically collected from the database. Additionally, the Bayesian Belief network used in EOS can only compare user attributes with LO attributes. The association rule miner in the iLOG system compares all the attributes (i.e., both user and LO) together.

Finally, Etchells (2006) discusses finding usage features in LOs to predict student final grades. A fuzzy inductive reasoning (FIR) is used for feature selection and a neural network for orthogonal search-based rule extraction (OSRE). This approach uses one feature selection algorithm, rather than the ensemble approach used by iLOG, which could result in fewer relevant features identified compared to the ensemble. Further, the neural network hand-tweaking is used to prevent overfitting and only selects the rule most relevant to the label (i.e., pass/fail the LO).

**LO Repositories**

Vargo (2003) suggested organizing repositories using levels of learning objects. Level 1 refers to single page, Level 2 to a lesson, Level 3 refers to a collection of Level 2 objects, (e.g., a course), and Level 4 refers to a set of courses that leads to a certificate. Unfortunately, such an organization has yet to be adopted. At present, existing LOs are stored in repositories such as Campus Alberta Repository of Educational Objects, Federal Government Resources for Educational Excellence, FreeFoto, Maricopa Learning Exchange, Merlot, Wisconsin Online Resource Center (Nash, 2005). These repositories are searchable based on LO metadata. However, there are three problems with searching for learning objects (Nash, 2005). First, the LOs are not interchangeable due to size, or inconsistent languages. Many also have a cultural bias. Second, there is an inconsistent classification scheme. Specifically, the learning levels for LOs (K-12 through graduate) are not specified. Third, the quality for LOs is highly variable in terms of production, classification, etc. Tompsett (2005) discusses how it can be very difficult for developers to create new courses from LOs stored in repositories. This is due to the difficulty of finding a set of LOs which integrate together well while still covering all the topics in the course. There is some existing work in helping developers to select LOs from repositories (Pythagoras, 2004; Broisin; 2005).

Pythagoras (2004) gives an algorithm to automatically select LOs from a repository by emulating the human-based LO selection process. This is done by training a classifier on the metadata for LOs selected by the developer over a small-scale test. The downside to this approach is that it requires all
the LOs to use the same set of metadata. Additionally, this approach can only be used to find LOs similar to those originally selected by the developer. Thus, this approach will not replace the need for developers to search LO repositories.

Broisin (2005) gives a service oriented architecture consisting of three layers: (1) learning management system (LMS) to deliver courseware, (2) learning object repository (LOR) to manage LOs and (3) mediation layer which bridges the LMS and LOR. This architecture automatically extracts a variety of metadata from the LMS and updates the LO in the repository. This includes general metadata (such as the title), semantics metadata (such as the science type and main discipline), pedagogical metadata (such as the role of the user), and technical metadata (such as the required operating system). On the surface, this approach is similar to that used by iLOG. However, the metadata supplied by this architecture is based entirely on the LO content. There is no consideration for creating metadata from user interactions with the LO as in iLOG. Thus, the performance of the LO is not considered in the metadata making it more difficult for developers to choose suitable LOs from the repository.

METHODOLOGY

In this section, we describe the two halves of the iLOG framework (see Figure 1). First, the LO Wrapper surrounds existing LOs and intercepts user interactions between the user and the Learning Management System (LMS). These user interactions are logged to the external iLOG database. The wrapper also adds metadata created by the iLOG framework to the existing LOs. Second, the MetaGen system is used by iLOG for automatic creation of metadata. MetaGen first extracts user interaction and static user/LO data into a self-contained dataset. MetaGen then analyzes the dataset using feature selection and rule mining components to create rules and statistics which are used as LO metadata. The iLOG framework adheres to both the SCORM (Dodds, 2001) and LOM (IEEE, 2002) standards. For existing SCORM-compliant LOs, using the iLOG framework only requires adding the LO Wrapper to a zipped file format. The LOs can then be uploaded to any SCORM-compliant LMS. The LO wrapper automatically stores the user interactions in real-time. MetaGen runs offline, but is fully automatic and can be run whenever new metadata is required.

![Figure 1. The Intelligent Learning Object Guide (iLOG) Framework (from Riley et al. 2009).](image-url)
Automation

Generally, tracking user interactions with the LOs requires modifications to the LMS where the LOs are displayed to users. The downside to this approach is that it requires non-standard modifications to the LMS which severely restrict the interoperability for these LOs and the potential user base both of which are inconsistent with the SCORM standard. It is a better idea to provide LOs with their own capability for tracking user interactions. First, this allows the LO to be deployed seamlessly using any existing SCORM-compliant LMS. Second, interested parties could then access the LOs directly to obtain stored user interactions. Miller et al. (2008) proposes adding this capability to the SCORM 2.0 project. However, this capability does not currently exist. Instead, we use the LO Wrapper which can be easily integrated into any SCORM-compliant LO.

The LO Wrapper uses the Easy Shareable Content Object (SCO) Adapter for SCORM 1.2 (Ostyn, 2006). The SCO adapter provides a direct interface with the SCORM API the LMS uses for displaying the LO. This connection to the SCORM API updates the LO Wrapper when pages are displayed to the user and also provides information about the assessment component. The LO Wrapper also uses existing web technologies including JavaScript and PHP to create a bridge between the LO and an external database. Using this bridge, the wrapper can transmit user interactions to the database and metadata back to the LO. This bridge requires a connection to the Internet, but this is generally not an issue because such a connection is also required for most LMSs.

Figure 2 summarizes the user interactions automatically captured by the LO Wrapper with corresponding examples of LO content for each component in iLOG LOs (i.e., tutorial, exercise, and assessment). In the tutorial, the wrapper captures user interactions with each page by hooking into the mouse events in the hypertext markup language (HTML) for LO pages in SCORM-compliant LOs. From these mouse events, the wrapper can deduce the type of user interactions. For example, the wrapper can distinguish between users scrolling down a page in the LO or clicking on an external hyperlink. The wrapper stores such user interaction events in collections in the JavaScript. Additionally, the LO wrapper is notified by the interface with the SCORM API when new LO pages are loaded in the tutorial. The wrapper updates collections in the JavaScript such as user interaction navigations along with the time spent on each page. In the exercise, the LO wrapper uses a direct interface with the exercise to collect user interactions from inside the embedded exercise. The wrapper provides an interface for exercises written in Flash and for those written as Java Applets. This interface allows the wrapper to collect user interactions about specific steps in the exercise. For example, the wrapper can obtain the time spent on the first sorting step in Figure 2 and whether or not the user got the correct answer. The wrapper updates collections in the JavaScript with information about each exercise step. The wrapper also stores the order steps are taken to reach the end of the exercise and any steps that prompt user requests for help. In the assessment, the LO wrapper stores in JavaScript collections all the information from the SCORM API for each problem in the assessment. This includes the time spent on each problem, the user answer, the correct answer, etc. The wrapper also stores the order problems are answered and overall answer statistics (e.g., average user score on problem in Figure 2). After the users finishes the assessment, the LO Wrapper automatically uses the JavaScript/PHP bridge to transmit the user interactions in the JavaScript collections to an external database.

MetaGen runs on a system external to the LMS. When it runs, MetaGen first connects to the iLOG database and downloads the user interactions and static LO/user data. Next, it uses a preprocessing script to extract a dataset from the database. Missing attribute-values are filled in using
a data imputation component (discussed below). Finally, MetaGen automatically computes suitable metadata using (1) statistics supplied by the developer, (2) ensemble feature selection component, and (3) association rule mining component. The latter two are discussed in more detail below. Currently, MetaGen is run offline to create new metadata after LOs are deployed to the LMS.

MetaGen Components

The MetaGen framework uses three separate modules for automatic metadata creation: (1) data logging, (2) data extraction, and (3) data analysis. First, the data logging module of MetaGen integrates data from three sources: (1) static LO data, (2) static student data, and (3) user interactions from the LO wrapper. Next, the data extraction module creates the iLOG dataset from the database. Each record in the dataset corresponds to a particular student-LO session. This module uses the Data Imputation component to fill in the missing attribute-values for the records. Finally, the data analysis module uses a multi-step process to generate the metadata. First, this module uses the Feature Selection Ensemble component to select only the most relevant features from the database. This feature subset is then passed Association Rule Miner component which creates useful metadata for the LOs. This module also contains usage statistics specified by the content developer. These statistics are also included as metadata for the LOs. For more information, consult Riley et al. (2009).
We next discuss all three important components in the MetaGen framework: (1) Data Imputation from the data extraction module, (2) Feature Selection Ensemble from the data analysis module and (3) Association Rule Miner also from the data analysis module.

**Data Imputation**

In our previous work (Riley et al., 2009) we discovered there were many records in the iLOG datasets which contained missing attribute-values. This was often the direct result of a lack of user interactions. For example, if the user skipped one of the interactive exercises or the evaluation survey then the attributes corresponding to this exercise/survey would have missing values for the record in the dataset. Many such records contain both missing and present attribute-values. The missing attribute-values make it difficult to use this record for feature selection and rule mining. However, simply discarding any record with missing attribute-values wastes a considerable amount of potentially interesting metadata. We would like to utilize such records in data analysis rather than preprocessing to remove all records with missing attribute-values. To facilitate this, we have added a Data Imputation component to the Metagen framework which fills in the missing attribute-values in dataset records.

The Data Imputation component uses the novel *Cluster-Impute algorithm* which employs (1) hierarchical clustering, (2) dynamic tree cuts, and (3) linear regression classifiers to fill in the missing attribute-values. The pseudocode for Cluster-Impute is given in Figure 3. The values for the parameters minInstClusterSize (= 5) (denoting the minimum size that a cluster of data instances must be) and minAttrClusterSize (3) (denoting the minimum size that a cluster of “attribute” instances must be) are chosen so that the algorithm can distinguish between attribute-values which can be filled in from data records and which need to be imputed from similar attributes. For minInstClusterSize, we want to make sure that each cluster has at least a handful (and of an odd number) of data points (or instances). The odd number preference is to allow for a majority in the decision making process. At the same time, we do not want to impose a large cluster size minimum as that would likely skew the clustering results. Thus, we chose 5 for our current design. As for minAttrClusterSize, we postulate that a larger cluster size for similar attributes would introduce too much noise into the algorithm. If two attributes are truly different then forcing them into one cluster would not be helpful. Further, since minAttrClusterSize is used only when the first step of Cluster-Impute fails, we believe that 3 was a good enough value for this parameter. Finally, missingThreshold is the threshold we use to decide whether the data points in the cluster are “strong” enough to perform step 1 of imputation, as will be elaborated further in the next paragraph. We chose 50% as the minimum.

First, Cluster-Impute employs a hierarchical clustering (Johnson, 1967) separately on both the data records and the attributes. This clustering consists of an agglomerative approach which starts with clusters containing a single object. These clusters are merged together progressively until there is only a single cluster containing all the objects. This results in a tree dendrogram containing different sets of clusters. Cluster-Impute uses a dynamic tree cut algorithm (Langfelder et al. 2008) to choose a set of clusters with sufficient size. Then, to impute the missing values, we use a 2-step mechanism. First, missing attribute-values are imputed using the values that are present for other data points in the same cluster, essentially exploiting the cluster-membership to estimate the missing values. However, if the cluster does not contain sufficient members with the attribute-values (i.e., less than 50%), then Cluster-Impute activates a second step that runs a linear regression classifier on the attribute clustering to determine which attributes are similar enough to be used instead to impute the missing attribute-
values. This second step is basically utilizing attribute-similarity to estimate the missing values—in a way, allowing attributes of high “correlation” to “help each other out”. The Cluster-Impute algorithm is later validated in the Results section.

Algorithm CLUSTER-IMPUTE

Input: (Data, minInstClusterSize, minAttrClusterSize, missingThreshold)

Initialize: minInstClusterSize := 5; minAttrClusterSize := 3; missingThreshold := .5;

InstanceClusters := GetInstanceClusters(Data, minInstClusterSize)
AttributeClusters := GetAttributeClusters(Data, minAttrClusterSize)
AttClassifiers := GetLinearRegressionClassifiers(Data, AttributeClusters)

For (InstClust in InstanceClusters) DO
    clusterSize := InstClust.numInstances;
    For (AttValuesInCluster in InstClust) DO
        If (# of AttValuesInCluster missing exceeds missingThreshold) THEN
            // Fill the missing values for that attribute in the cluster with the mean of the
            // non-missing values for the attribute in that cluster. (Step 2 Imputation)
        Else
            // Call the classifier that is built for this attribute on each instance in the cluster to
            // fill in the values for that attribute. (Step 1 Imputation)
        End If
    End For
End For

// CLUSTER-IMPUTE

GetLinearRegressionClassifiers(Data, AttClusters);
// returns a list of classifiers, one for each attribute
Classifiers := empty list of type classifier

Foreach Att in Data:
    ClusterData := subset of Data consisting only of attributes in AttCluster with Att
    Label := Att
    TrainingData := ClusterData minus Att
    Classifiers[Att] := GetClassifier(TrainingData, Label)
End Foreach

Return Classifiers

GetClassifiers(TrainingData, Label); returns an object of type classifier
// Call any classifier (we use Linear Regression) algorithm from the RWeka set of tools
// (Witten & Frank, 2005).

Return Classifier

GetInstanceClusters(Data, minInstClusterSize); // returns array of inst cluster assignments
InstDistMatrix := GetDistMatrix(Data)
Return GetClusters(InstDistMatrix)

GetAttributeClusters(Data, minAttrClusterSize); // returns array of cluster assignments
AttDistMatrix := GetDistMatrix(transpose(Data))
Return GetClusters(InstDistMatrix)

12
GetClusters(DistMatrix);
  Dendrogram := hierarchicalClustering(DistMatrix)
Return dynamicTreeCut(Dendrogram)

Figure 3: CLUSTER-IMPUTE Algorithm

Feature Selection Ensemble

Attributes in the iLOG dataset are collected from many different kinds of user interactions with the
learning objects (LOs). The learning outcome (i.e., label) for the iLOG dataset is whether or not
students pass the assessment component for the LO. For the iLOG dataset, not all the attributes
collected are equally important (i.e., relevant) to the learning outcome. The inclusion of unimportant
attributes (i.e., irrelevant to the label) often degrades the classification model (Hand, et al. 2001).
Unfortunately, we do not know which attributes are relevant in the iLOG dataset when running
MetaGen. As a result, MetaGen uses feature selection algorithms to choose the relevant subset of
attributes used for the entire iLOG dataset. Note, for purposes of terminology, features and attributes
are equivalent in this section.

The MetaGen feature selection component uses feature selection algorithms from the Weka
software library (Witten & Frank, 2005). There are two different types of feature selection algorithms
in Weka: (1) subset evaluation and (2) attribute evaluation. The key difference between the two types
is the way they evaluate the attributes. The subset evaluation algorithms evaluate the attributes
together, while the attribute evaluation algorithms evaluate them separately. Subset evaluation
algorithms use a search algorithm to find subsets of attributes and each uses a distinct fitness function
to evaluate the subsets. Attributes are added to subsets only if they improve fitness. After searching,
the algorithm returns the subset of attributes with the highest fitness. On the other hand, attribute
evaluation algorithms evaluate each attribute separately. Each attribute evaluation algorithm uses a
distinct fitness function to individually evaluate the attributes. After evaluation, the algorithm returns
all the attributes each with a score based on the fitness values.

Neither of the feature selection algorithms is intrinsically superior to the other. Rather, each
algorithm specializes on finding different kinds of relevant attributes based on the attribute-values in
the dataset. The iLOG dataset contains different kinds of attributes collected from user interactions.
Thus, to improve feature selection in MetaGen we employ an ensemble of feature selection
algorithms. In an ensemble approach, multiple algorithms are run on the same dataset and each
contributes to the final decision of which attributes are relevant. The MetaGen ensemble currently
employs 10 different feature selection algorithms summarized in Table 1. A description of the
individual algorithms is outside the scope of this paper. Interested readers should consult (Guyon &
Elisseef, 2003) for more details.

The ensemble combines the relevant attributes chosen by all the algorithms using a voting
scheme. The subset evaluation algorithms vote for all the attributes in their subset. On the other hand,
the attribute evaluation algorithms vote for attributes based on the individual score for the attribute
computed by that algorithm. After all the votes are tallied, the ensemble chooses the relevant
attributes which have votes from the majority of the algorithms. This approach allows the ensemble to
leverage the strengths of multiple feature selection algorithms and insures highest percentage of
relevant attributes is found. The ensemble is validated in the Results section.
Table 1: Feature selection algorithms in MetaGen ensemble

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CfsSubsetEval</td>
<td>SUBSET</td>
</tr>
<tr>
<td>ClassifierSubsetEval</td>
<td>SUBSET</td>
</tr>
<tr>
<td>ConsistencySubsetEval</td>
<td>SUBSET</td>
</tr>
<tr>
<td>CostSensitiveSubsetEval</td>
<td>SUBSET</td>
</tr>
<tr>
<td>FilteredSubsetEval</td>
<td>SUBSET</td>
</tr>
<tr>
<td>WrapperSubsetEval</td>
<td>SUBSET</td>
</tr>
<tr>
<td>ChiSquaredAttributeEval</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td>ReliefFAttributeEval</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td>SymmetricalUncertAttributeEval</td>
<td>ATTRIBUTE</td>
</tr>
<tr>
<td>CostSensitiveAttributeEval</td>
<td>ATTRIBUTE</td>
</tr>
</tbody>
</table>

**Association Rule Mining**

Association rules show attribute values that co-occur frequently in a given dataset. Identifying or mining these rules allows one to gain insights to how attributes behave in relevance to each other. The Association Rule Mining component in the iLOG Framework uses the Tertius algorithm that is a top-down rule discovery system based on first-order logic representation (Flach & Lachiche, 2001). The main advantage of Tertius over other rule discovery systems, such as Apriori, is in the confirmation function it uses to evaluate the rules. The confidence function uses concepts from categorical data analysis. First, the confidence function evaluates potential rules using a modified chi-squared test on the attribute-values. Second, it employs an A* search to find only the unique rules--other rules are automatically pruned. Finally, the confirmation function allows the use of background knowledge when computing the rules. In the iLOG framework, this background knowledge consists of the assessment score for the student (i.e., pass/fail). The use of background knowledge allows Tertius to be applied to supervised learning tasks such as concept learning. In iLOG, concept learning consists of finding attributes-values which, considered together, are most relevant to the assessment score. This differs from the feature selection ensemble. Tertius considers individual, attribute-values for separate attributes, while the ensemble considers all the attribute-values for a single attribute to decide whether the entire attribute is relevant. Tertius uses a top-down search algorithm when creating the association rules. The search first starts with an empty rule. Next, Tertius iteratively refines the rule by adding new-attribute values. Tertius continues to refine the rule as long as such refinements increase the confidence value. Finally, Tertius adds the rule and restarts the search to create new rules. Tertius ends when no additional rules can be created with sufficient confidence values. Afterwards, it returns the set of rules along with their confidence values which are used as metadata for the iLOG framework.

The Tertius algorithm used in Association Rule Mining component only operates on nominal-valued attributes (Deltour, 2001). This is because of the implementation of the non-redundant refinement operator. As a result, the numeric attributes from the Data Imputation component must be converted from numeric- to nominal-valued attributes. The iLOG framework uses the multi-interval discretization method proposed in Fayyad (1993) that uses an information entropy minimization heuristic to convert the attributes. We validate the Tertius algorithm in the Results section.
RESULTS

In this section, we first provide a validation for all three MetaGen components used for automatic metadata creation. We then discuss the suitability of the iLOG metadata separately for both instructors and users.

Validation for MetaGen Components

To demonstrate the effectiveness of Metagen, we provide a rigorous validation for all three MetaGen components: (1) Data Imputation, (2) Feature Selection Ensemble, and (3) Association Rule Mining. This validation includes analysis of results for all three components run separately on a mix of the iLOG dataset, synthetic datasets, and datasets from the UCI machine learning repository (Asuncion & Newman, 2007).

Data Imputation

The Data Imputation component is used to fill in missing attribute-values for records in the iLOG dataset. The goal of validation for this component was to determine if the missing values are imputed correctly. We use the iLOG dataset to validate this component because it contains a wide variety of different attributes for imputation.

First, the iLOG dataset is pre-processed to remove all the records containing missing attribute-values so we can determine which attributes are imputed correctly. We refer to this version as the iLOG complete dataset. Next, a certain percentage of the total, remaining attribute-value are selected uniformly at random and marked as missing. Finally, we run the Cluster-Impute algorithm to fill in all the missing attribute-values and compare them to the original, correct attribute-values. The Data Imputation accuracy is measured as the ratio of the number of missing attribute-values correctly imputed over the total, missing attribute-values. However, the chances that the exact, numeric attribute value will be computed are very small. Thus, we measure whether the two attribute-values are approximately equal using a heuristic based on statistical methods for computing equivalency (Wellek, 2002). The imputed attribute-value is considered to be correct if it is within the acceptance interval of one standard deviation measured of the correct attribute value. The standard deviation is measured using all the values for a single attribute in the dataset.

Table 2 gives the accuracy for Cluster-Impute on the iLOG complete dataset with varying amounts of total, missing-attribute values. It also gives the results for the two one-sided tests for equivalence (TOST) (Wellek, 2002). Note that many methods in statistics (e.g., t-test, ANOVA, Kolmogorov-Smirnov, etc) are designed to show that two samples are sufficiently different by rejecting the null hypothesis that they are the same. TOST works the opposite way; it shows two samples are sufficiently equivalent by rejecting the null hypothesis that they are different. The results show that Cluster-Impute achieves high imputation accuracy even when 20% of the total attribute-values are missing. Further, there is statistical significance (at p-value <0.0001, epsilon 0.36) that the imputed attribute-values are equivalent to the correct attribute-values. These results indicate that the Cluster-Impute algorithm is able to correctly impute missing attribute-values even in datasets with a wide variety of different attributes.
Table 2: Cluster-Impute results on iLOG dataset. The missing percentage (Miss) of attribute-value pairs is given along with the imputation accuracy and the results of the TOST equivalence test.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Miss</th>
<th>Accuracy</th>
<th>Hypothesis</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>iLOG Complete</td>
<td>5%</td>
<td>0.7921</td>
<td>reject</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>iLOG Complete</td>
<td>10%</td>
<td>0.8180</td>
<td>reject</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>iLOG Complete</td>
<td>15%</td>
<td>0.8260</td>
<td>reject</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>iLOG Complete</td>
<td>20%</td>
<td>0.7905</td>
<td>reject</td>
<td>&lt;0.0001</td>
</tr>
</tbody>
</table>

**Feature Selection Ensemble**

Results in Riley et al. (2009) show that the iLOG dataset contains both relevant and irrelevant attributes. However, we cannot be certain which attributes in the iLOG dataset are relevant—i.e., we do not have the “ground-truth”—because they are based on the real-world user interactions with the learning objects. Thus, our validation strategy is based on simulated datasets which resemble the iLOG dataset specifically exemplifying this property of having a mixture of relevant and irrelevant attributes. These synthetic datasets were created using RDG1 generator in Weka (Witten & Frank, 2005). The RDG1 generator uses a decision list to create instances with attribute-values consistent with rules based on the labels. Interested readers should consult (Witten & Frank, 2005) for more information. For these datasets, we can specify the exact number of relevant and irrelevant attributes. Thus, we have the full information on which attributes are relevant/irrelevant allowing us to evaluate the ensemble.

Furthermore, this evaluation consists of two parts. First, we justify the need for a feature selection ensemble. This is done by showing that feature selection algorithms identify different subsets of relevant attributes on the same datasets. None of the algorithms are intrinsically superior because none identify all the relevant attributes by themselves. Thus, we could find more relevant attributes by combining the results using an ensemble. Second, we demonstrate that an ensemble effectively combines the results to identify the relevant attributes despite the presence of irrelevant attributes. This is done by comparing the number of relevant attributes found for individual algorithms and the ensemble.

The same 30 synthetic datasets are used for both parts of the feature selection validation. All these datasets contain 100 data points (or instances) with 20 total attributes each. However, they contain varying numbers of relevant and irrelevant attributes. Datasets D1-D10 contain 5 relevant and 15 irrelevant attributes, D11-D20 contain 10 of each, and D21-D30 contain 15 relevant and 5 irrelevant attributes. Thus, we also consider the effects of datasets with a greater percentage of relevant/irrelevant attributes on the algorithms. For part one of the validation we use the chi-square test on a contingency table to evaluate whether the relevant attributes selected are dependent on individual feature selection algorithms. For part two of the validation we use ANOVA contrasts to compare the performance of the individual algorithms with the ensemble.

**Validate Need for Ensemble**

Table 3 gives the relevant attribute counts for all 10 feature selection algorithms used in the Ensemble run, independently, on the 30 synthetic datasets. It also shows the number of times each attribute was relevant considering all the datasets. Overall, there is considerable variation in the counts for the
number of attributes found. In fact, a chi-square test on the resulting contingency table provides evidence (with \( p < 0.0001 \)) that the attributes selected are not independent of the feature selection algorithms. Recall that feature selection algorithms use different fitness functions and specialize on finding different kinds of attributes. Some of the algorithms are more conservative (e.g., \( \text{CfsSubsetEval} \)) and some are more aggressive (e.g., \( \text{SymmetricalUncertAttributeEval} \)). However, none of the algorithms were able to find all the times each attribute was relevant. Further, the feature selection algorithms each choose different subsets of relevant features. Taken together, this provides motivation for using an ensemble approach for feature selection which can combine the different subsets to find more relevant attributes.

**Table 3:** The number of times an attribute (A1-A20) was identified as relevant by the different feature selection algorithms and the total number of times each attribute was relevant for 30 synthetic datasets.

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{CfsSubsetEval} )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \text{ClassifierSubsetEval} )</td>
<td>6</td>
<td>9</td>
<td>6</td>
<td>4</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>( \text{ConsistencySubsetEval} )</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>( \text{CostSensitiveSubsetEval} )</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{FilteredSubsetEval} )</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>( \text{WrapperSubsetEval} )</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>16</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>( \text{ChiSquaredAttributeEval} )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \text{ReliefFAttributeEval} )</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>13</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>( \text{SymmetricalUncertAttributeEval} )</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>15</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>( \text{CostSensitiveAttributeEval} )</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td><strong>TIMES RELEVANT</strong></td>
<td>16</td>
<td>14</td>
<td>14</td>
<td>10</td>
<td>23</td>
<td>13</td>
<td>15</td>
<td>12</td>
<td>15</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A11</th>
<th>A12</th>
<th>A13</th>
<th>A14</th>
<th>A15</th>
<th>A16</th>
<th>A17</th>
<th>A18</th>
<th>A19</th>
<th>A20</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{CfsSubsetEval} )</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{ClassifierSubsetEval} )</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>11</td>
<td>9</td>
<td>6</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>( \text{ConsistencySubsetEval} )</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>( \text{CostSensitiveSubsetEval} )</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( \text{FilteredSubsetEval} )</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( \text{WrapperSubsetEval} )</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>( \text{ChiSquaredAttributeEval} )</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \text{ReliefFAttributeEval} )</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>12</td>
<td>4</td>
<td>13</td>
<td>10</td>
<td>2</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>( \text{SymmetricalUncertAttributeEval} )</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>12</td>
<td>3</td>
<td>12</td>
<td>10</td>
<td>6</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>( \text{CostSensitiveAttributeEval} )</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td><strong>TIMES RELEVANT</strong></td>
<td>14</td>
<td>16</td>
<td>12</td>
<td>19</td>
<td>12</td>
<td>19</td>
<td>17</td>
<td>12</td>
<td>18</td>
<td>17</td>
</tr>
</tbody>
</table>

Validate Ensemble Results
Table 4 gives the number of relevant attributes found by the individual algorithms and the ensemble on each of the 30 synthetic datasets. From the results, we observe that the ensemble finds many more relevant attributes than the individual algorithms. In fact, ANOVA contrasts comparing (1) the ensemble with the subset evaluation algorithms, (2) with the attribute evaluation algorithms, (3) and with all the algorithms together provides evidence (with p<0.0001) that the ensemble achieves superior results in terms of identifying relevant attributes. This demonstrates that the ensemble is capable of merging the results from the individual algorithms to identify a larger subset of relevant attributes. Further, the number of relevant attributes found on most synthetic datasets is very close: 5 on datasets D1-D10, 10 on D11-D20, and 15 on D21-D30. The varying number of relevant and irrelevant attributes has little impact on the ensemble because it utilizes both conservative and aggressive feature selection algorithms. However, we observe that the ensemble also misidentify a greater number of irrelevant attributes as relevant. In the current design of the iLOG framework, these irrelevant attributes would be eventually filtered out by the rule mining process (as rules with irrelevant attributes will likely lead to low coverage and confidence). Nevertheless, we realize the need to balance the effectiveness and efficiency of our ensemble algorithm and will address this issue in our future work.

Table 4: Relevant attributes found on 30 synthetic datasets. Datasets D1-D10 have 5 relevant attributes, D11-D20 have 10 relevant attributes, and D21-D30 have 15 relevant attributes. The results show that some algorithms are more aggressive than others.

<table>
<thead>
<tr>
<th></th>
<th>CfsSubsetEval</th>
<th>ClassifierSubsetEval</th>
<th>ConsistencySubsetEval</th>
<th>CostSensitiveSubsetEval</th>
<th>FilteredSubsetEval</th>
<th>WrapperSubsetEval</th>
<th>ChiSquaredAttributeEval</th>
<th>ReliefFAttributeEval</th>
<th>CostSensitiveAttributeEval</th>
<th>OneRAttributeEval</th>
<th>AVERAGE</th>
<th>ENSEMBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
<td>D4</td>
<td>D5</td>
<td>D6</td>
<td>D7</td>
<td>D8</td>
<td>D9</td>
<td>D10</td>
<td>D11</td>
<td>D12</td>
</tr>
<tr>
<td>CfsSubsetEval</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ClassifierSubsetEval</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ConsistencySubsetEval</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CostSensitiveSubsetEval</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>FilteredSubsetEval</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>WrapperSubsetEval</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ChiSquaredAttributeEval</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ReliefFAttributeEval</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CostSensitiveAttributeEval</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OneRAttributeEval</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>AVERAGE</strong></td>
<td><strong>3.1</strong></td>
<td><strong>2.4</strong></td>
<td><strong>2.4</strong></td>
<td><strong>3.1</strong></td>
<td><strong>2.5</strong></td>
<td><strong>2.4</strong></td>
<td><strong>2.6</strong></td>
<td><strong>3.5</strong></td>
<td><strong>1.6</strong></td>
<td><strong>2.1</strong></td>
<td><strong>0</strong></td>
<td><strong>0</strong></td>
</tr>
<tr>
<td><strong>ENSEMBLE</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>4</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
<td><strong>5</strong></td>
</tr>
</tbody>
</table>


The Tertius component in MetaGen uses the relevant attributes from the iLOG dataset chosen by feature selection ensemble. Tertius then returns a set of association rules in the form of horn clauses with literals based on attribute-values associated with a specific assessment values. This following example rule contains three literals (including the assessment):

\[
\text{takenCalculus?} = \text{yes AND assessmentMaxSecOnAPageAboveAvg?} = \text{yes} \rightarrow \text{pass.}
\]

These association rules are a significant part of the metadata supplied by MetaGen. Obviously, we cannot know for certain whether the association rules should be associated with specific assessment values in the iLOG dataset. To address this problem, we validate the Tertius rule miner, in this section, on synthetic, benchmark datasets from the UCI machine learning repository. On these datasets we know \textit{in advance} which attribute-values are associated with labels (i.e., assessment values). We expect Tertius to be able to discover association rules with these attribute-values and specific labels. Further, recall that feature selection in MetaGen removes attributes completely that are deemed irrelevant to the assessment value. As part of the validation, we would like to determine whether such irrelevant features have any influence on Tertius. Thus, we do not perform any feature selection on these datasets.

Specifically, we used four datasets: Monks-1, Monks-2, Monks-3, and Tic-Tac-Toe from the UCI machine learning repository (Asuncion & Newman, 2007). All three Monks datasets contain the same set of six attributes with nominal values and one binary label (i.e., the classification of an instance).
They differ in which attribute-values are associated with the binary label. The Tic-Tac-Toe dataset gives the end-game positions on a Tic-Tac-Toe board. The label is whether player X is the winner or not. The datasets all share the following properties: (1) the label always involve multiple relevant attributes, (2) many different combinations of attribute-values give the same label, and (3) we know all the attribute-value combinations associated with each label. We chose these datasets because of their similarity to the iLOG dataset. Specifically, there are many different attribute-values combinations in the iLOG dataset which result in passing the assessment (Riley et al., 2009). We cannot evaluate Tertius on iLOG because we do not know all the attribute-value combinations associated with each label. However, we can evaluate Tertius on these four datasets because all the combinations are known and generalize from the results the validity or quality of our metadata when applying Tertius to iLOG.

Table 5 gives the association rules with the highest confidence created by Tertius for the Monks datasets. Based on the dataset description, we use three literals for Tertius.

First, for the Monks-1 dataset, Tertius correctly identifies the attribute-values associated with both class values (i.e., labels). The rules for the class=1 are identical to those given in the Monks-1 dataset description. The class=1 rules have higher confidence because there are more attribute-value combinations for the class=0 label. Specifically, any other combination results in the class=0 label.

Second, for the Monks-2 dataset, the association rules are not consistent with the dataset description. According to the dataset description, the Monks-2 dataset has class=1 when exactly two attributes=1 rather than class=0. For example, two attributes=1 have class=1, three (or more) attributes=1 have class=0. However, there are 72 data points in Monks-2 with the attribute-values for the Monks-2 rule in Table 5, but only 12 of them have class=1. Thus, the Tertius association rules are consistent with the data, but inconsistent with the dataset description. To exactly match the dataset description for Monks-2 (i.e., exactly 2 attributes equal to 1) would require increasing the number of literals used to one for each attribute. Regardless, the rules given using only three literals are still a good approximation for the data points in Monks-2.

Finally, for the Monks-3 dataset, Tertius correctly identifies the attribute-values with class=0. These are the specific attribute-values given in the dataset description. The opposite of Monks-1, other attribute-values in Monks-3 have class=1. Again, the class=0 rules have higher confidence than class=1 rules because there are many more attribute-value combinations. This is consistent with the dataset description.

Further, we observe that Tertius generates rules involving irrelevant attributes for Monks-1 (not shown) and Monks-3 (see Table 5). However, these rules are otherwise identical (i.e., attribute-values and labels) to higher confidence rules with only relevant attributes. Thus, Tertius seems fairly resistant to the inclusion of irrelevant attributes. Taken together, the results validate the efficacy of Tertius on the Monks datasets. Specifically, (1) rules with fewer literals are still a good approximation of the data and (2) the presence of irrelevant attributes only results in the creation of additional, weaker rules.

Table 5: Tertius Rules for the Monks Datasets, where Conf denotes rule confidence.
Table 6 gives the highest confidence rules created by Tertius for the Tic-Tac-Toe dataset. This dataset represents the end-game states for the simple Tic-Tac-Toe game. The label (i.e., class) is positive when "x" wins and negative when "o" wins or there is a draw. Based on the game, we use four literals for Tertius corresponding to three squares needed to win and the label. The first two association rules in Table 6 are certainly consistent with the dataset description. In Tic-Tac-Toe, the player who takes the center square is much more likely to win. The rest of the association rules in Table 6 correspond to winning configuration for Tic-Tac-Toe (i.e., three in a row, column or diagonal). This includes all eight win conditions for "x". Many of the other rules created by Tertius (not shown) involve the middle square and one other square. All attribute-values and labels for such rules are consistent with the probable outcome based on the dataset description. Again, the results for this dataset validate the efficacy of Tertius. However, all but the first two association rules in Table 6 require four literals to consider the three attributes and label. Thus, using too few literals can result in the omission of interesting association rules. On the other hand, increasing the number of literals (1) drastically increases the running time for Tertius and (2) seems to generate a larger number of weaker rules.

Table 6: Tertius Rules for the Tic-Tac-Toe Dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Conf</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monks-1</td>
<td>0.55</td>
<td>attr5 = 1 =&gt; class = 1</td>
</tr>
<tr>
<td>Monks-1</td>
<td>0.31</td>
<td>attr1 = 1 and attr2 = 1 =&gt; class = 1</td>
</tr>
<tr>
<td>Monks-1</td>
<td>0.31</td>
<td>attr1 = 2 and attr2 = 2 =&gt; class = 1</td>
</tr>
<tr>
<td>Monks-1</td>
<td>0.31</td>
<td>attr1 = 3 and attr2 = 3 =&gt; class = 1</td>
</tr>
<tr>
<td>Monks-1</td>
<td>0.18</td>
<td>attr5 = 2 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-1</td>
<td>0.18</td>
<td>attr3 = 3 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-1</td>
<td>0.18</td>
<td>attr5 = 4 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-2</td>
<td>0.15</td>
<td>attr3 = 1 and attr4 = 1 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-2</td>
<td>0.15</td>
<td>attr1 = 1 and attr4 = 1 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-2</td>
<td>0.15</td>
<td>attr1 = 1 and attr6 = 1 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-2</td>
<td>0.15</td>
<td>attr2 = 1 and attr3 = 1 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-3</td>
<td>0.51</td>
<td>attr2 = 3 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-3</td>
<td>0.57</td>
<td>attr5 = 4 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-3</td>
<td>0.36</td>
<td>attr2 = 3 and attr3_irrel = 1 =&gt; class = 0</td>
</tr>
<tr>
<td>Monks-3</td>
<td>0.36</td>
<td>attr2 = 3 and attr3_irrel = 2 =&gt; class = 0</td>
</tr>
</tbody>
</table>

Table: Tertius Rules for the Tic-Tac-Toe Dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Conf</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.31</td>
<td>middle-middle-square = o =&gt; Class = negative</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.31</td>
<td>middle-middle-square = x =&gt; Class = positive</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.23</td>
<td>top-left-square = o and middle-middle-square = o and bottom-right-square = o =&gt; Class = negative</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.23</td>
<td>top-right-square = x and middle-middle-square = x and bottom-right-square = o =&gt; Class = negative</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.22</td>
<td>top-left-square = x and middle-middle-square = x and bottom-right-square = x =&gt; Class = positive</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.20</td>
<td>top-middle-square = x and middle-middle-square = x and bottom-middle-square = x =&gt; Class = positive</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.20</td>
<td>top-left-square = x and top-middle-square = x and top-right-square = x =&gt; Class = positive</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.20</td>
<td>top-right-square = x and middle-right-square = x and bottom-right-square = x =&gt; Class = positive</td>
</tr>
<tr>
<td>Tic-Tac-Toe</td>
<td>0.20</td>
<td>bottom-left-square = x and bottom-middle-square = x and bottom-right-square = x =&gt; Class = positive</td>
</tr>
</tbody>
</table>
Validation for MetaData Results

Here we demonstrate the suitability of the metadata created using the MetaGen framework, particularly from an instructional support perspective and a student pedagogical perspective. This metadata is the result of all three MetaGen components evaluated on the iLOG dataset from the initial deployment (Riley, et al. 2009). This dataset contains records with user interactions and static user/LO data from four introductory computer sciences courses using five separate learning objects. We use two different configurations for validating the metadata. The first configuration uses all the records for a single LO across multiple courses. The second uses all the records for a single course across multiple LOs. This results in nine separate metadata sets for the validation. Recall that iLOG metadata consists of association rules created from specific attribute-values with a corresponding confidence value. We need to determine whether these association rules are suitable for both users and instructors. These separate groups have very different needs for metadata (Ravasio, 2003; Kosba, 2007) which are inherently subjective making empirical validation of the metadata impractical. Instead, we collaborate with educational experts to validate the usefulness of metadata for both groups based on expert knowledge. In the future, we will consider machine learning techniques for the empirical validation of the metadata.

In order to provide a validation of the appropriateness of the metadata for use by teachers and students, results were reviewed by a faculty member from the College of Education. This education expert specifically focused on association rules with the highest rule strengths (i.e. above .15). The rules strengths are calculated based on the confirmation and counter example scores from Tertius (Riley, et al. 2009). In general, it was found that variables traditionally associated with higher learning were represented in the association rules. In particular, time spent on various sections of the LO, including the assessment, was predictive of pass/fail on the LO assessment. The level of interactivity, as represented by the number of clicks on sections of the LO, was also predictive of learning. This result clearly supports the value of active learning, which is a well researched instructional strategy (Nugent, et al., 2009; Astrachen, et al., 2002). Students evaluative rating of the LO, as determined by a 1 to 5 scale, was also a key variable, and supports research showing the relationship between student attitudes and achievement (Alsop & Watts, 2003; Koballa & Glynn, 2007). Students’ self-efficacy, as represented by perceptions of their confidence in their computer science knowledge and attitudes, sense of academic preparation for the particular computer science course, and grade expectation, was reflected in the association rules. Another attitudinal variable represented was student motivation, which tapped their motivation to learn more about computer science, their interest in the content area of the course, and their expectation to take more computer science courses. In summary, the association rules predicting students’ score on the LO assessment encapsulated key variables which research has shown to be predictive of student learning.

These results also support earlier research using an educational statistics regression approach to identify variables which predicted student learning on the LO assessment. Combining data across course and LOs, it was found that there were differences in the LOs in terms of student learning and that more time spent on the LO and the use of active learning strategies contributed to greater learning (Nugent, et al., 2009).

Using two different metadata configurations—one for the individual LOs and one for the specific courses—was considered a good strategy. The individual LO data provided information about the suitability of LOs for different types of students and different types of student behavior (i.e. high self
efficacy and high interactivity), while the course data provided insight into the types of students and student behavior in a particular course associated with greater learning from LOs.

The educational expert recommended that format of the metadata output be simplified and codified for greatest usability by students and teachers. While such detailed metadata will be of interest to researchers, teachers and students want clear and simplified information about what types of students will best benefit from the LOs and how the LO can most profitably be used.

CONCLUSIONS

The traditional classroom approach involving the textbook and lecture hall has several significant problems motivating the use of online education programs. Learning objects (LO) are small, self-contained lessons which are often used in such programs. LOs are commonly stored in searchable repositories to facilitate reuse. Course developers search a repository for suitable LOs based on the LO metadata. Unfortunately, based on the current standards, such metadata is often missing or incorrectly entered making searching difficult or impossible. In this paper, we investigate automating metadata creation for LOs based on user interactions with the LO and static information about LOs and users. We present the Intelligent Learning Object Guide (iLOG) framework which consists of two components. First, the LO wrapper logs user interactions with the LO to an external database and updates the LOs with new metadata. Second, the MetaGen system generates metadata automatically based on user interactions and static information. To accomplish this, MetaGen extracts and analyzes a dataset using three separate components. First, the data imputation component is used to fill in any missing attribute-values in the dataset. This component uses Cluster-Impute, a novel algorithm presented here which employs hierarchical clustering, dynamic tree cuts and linear regression to fill in missing attribute-values based similar, known attribute-values. MetaGen next employs a feature selection ensemble component to select the subset of attributes most relevant to the learning goal (e.g., helping students pass the assessment). Finally, MetaGen uses the association rule miner component to create rules based on only the relevant attributes for the database records. These rules are automatically combined with any usage statistics specified by the content developer into LO metadata. Such metadata could be appended to the LO using the LO wrapper. Lastly, we provide a rigorous validation for all three components and for metadata created from real-world datasets using MetaGen. The MetaGen components are validated using a mix of real-world and synthetic datasets. The metadata is validated separately based on its suitability for both users and instructors.

In the future, we intend to evaluate the iLOG framework on additional LOs deployed to larger group of students. This expanded deployment should allow iLOG to generate even more interesting metadata and should provide information on how instructors and users utilize existing metadata. We would also like to branch out into different LO content areas. The current sets of LOs are designed based on introductory computer science topics. We would like to compare metadata from user interactions on these topics to metadata created for LOs on different topics. Regarding the MetaGen system, the feature selection ensemble currently emphasizes only the relevant attributes. It finds all relevant attributes, but also some irrelevant attributes. We need to investigate how to reduce the number of irrelevant attributes found without adversely affecting the ensemble. Finally, the metadata validation currently requires the assistance of education experts. We would prefer to empirically evaluate the metadata based on its suitability for both users and instructors. One possibility is training a machine learning algorithm to identify which specific, attribute-values are suitable for users and which are suitable for instructors.
Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0632642 and an NSF GAANN fellowship. The authors would like to thank other members of the iLOG team for their programming and data processing work: Sarah Riley, Erica Lam, WayLoon Tan, Beth Neilsen, and Nate Stender.

References


and rule extraction in a virtual course. In *European Symposium on Artificial Neural Networks* (pp. 401-406). Bruges, Belgium.


