# MLeXAI: A Project-Based Application-Oriented Model

INGRID RUSSELL
University of Hartford
ZDRAVKO MARKOV
Central Connecticut State University
TODD NELLER
Gettysburg College
and
SUSAN COLEMAN
University of Hartford

Our approach to teaching introductory artificial intelligence (AI) unifies its diverse core topics through a theme of machine learning, and emphasizes how AI relates more broadly with computer science. Our work, funded by a grant from the National Science Foundation, involves the development, implementation, and testing of a suite of projects that can be closely integrated into a one-term AI course. Each project involves the development of a machine learning system in a specific application. These projects have been used in six different offerings over a three-year period at three different types of institutions. While we have presented a sample of the projects as well as limited preliminary experiences in other venues, this article presents the first assessment of our work over an extended period of three years. Results of assessment show that the projects were well received by the students. By using projects involving real-world applications we provided additional motivation for students. While illustrating core concepts, the projects introduced students to an important area in computer science, machine learning, thus motivating further study.

Categories and Subject Descriptors: K 3.2 [**Computers and Education**]: Computer Science Education

General Terms: Experimentation

Additional Key Words and Phrases: Artificial intelligence, projects

This work is supported in part by the National Science Foundation under grant number DUE-0409497.

Author's address: I. Russell, Department of Computer Science, Dana 220 University of Hartford, West Hartford, Connecticut; email: irussell@hartford.edu.

**ACM Reference Format:**
Russell, I., Markov, Z., Neller, T., and Coleman, S. 2010. MLeXAI: A project-based application-oriented model. ACM Trans. Comput. Educ. 10, 3, Article 11 (August 2010), 36 pages.
DOI = 10.1145/1821996.1821999. http://doi.acm.org/10.1145/1821996.1821999.

## 1. MOTIVATION FOR AN INTRODUCTORY AI COURSE

Although we have written in the past about how a resource-constrained institution may integrate recommended core Intelligent Systems units into other core courses in the computer science (CS) curriculum [Russell and Neller 2003], we begin here by suggesting that an institution with sufficient resources would be well advised to offer an introductory AI course. Following this, we describe our own relevant curricular offerings and experiences with these materials across several institutions.

AI is a strong incubator for CS innovation. The philosophical hypotheses of whether or not machines can act as if they were intelligent versus whether or not machines can actually be intelligent are referred to respectively as the "weak AI" and "strong AI" hypotheses [Russell and Norvig 2003]. Consider the broad scope of "weak AI" research: The creation of any problem-solving artifact said to exhibit intelligent behavior could be said to be weak AI research. Why then do we not refer to most of CS research as AI research?

It could be argued that we use "AI" to encompass all computational innovation that is not (yet) embraced by another research subfield or curricular component. For example, when we go beyond what knowledge can be represented or inferred by modern databases, we find ourselves in the AI subfield of knowledge representation and reasoning. When control problems and/or computational approaches to such problems step outside the boundaries of classical control engineering, we find ourselves in the AI subfield of robotics. Artificial intelligence is the really interesting miscellaneous category of computer science. However, one must not confuse miscellaneous with extraneous.

AI is ambitious. Work in AI is often motivated by grand challenges that are extremely difficult. Whether spurred by a philosopher claiming that a computer could never play chess, or encouraged by DARPA organizing a race for autonomous vehicles, AI practitioners often start with an intractable problem and work to engineer a solution. They observe that humans with significant cognitive limitations often manage approximately optimal solutions to intractable problems. Human intelligence inspires intelligent engineering.

AI is interdisciplinary. Within a broad scope of engineering intelligent solutions, one finds broad interaction with philosophers, linguists, musicians, neuroscientists, and almost all other engineering disciplines. Richard Wyatt [2000] and Susan Fox [2007] propose introductory AI courses inviting interdisciplinary enrollment to foster interaction across disciplines.

Stuart Russell once said, "An intelligent system is one whose expected utility is the highest that can be achieved by any system with the same computational limitations" [Russell and Norvig 2003]. One might say that an intelligent curriculum is one whose expected utility is the highest that can be achieved by any curriculum with the same resource limitations. As an important incubator for computer science innovation, pursuing ambitious, broad goals that foster interdisciplinary interaction, artificial intelligence offers high utility to a curriculum.

In this article, we describe a machine learning-themed model to the introductory artificial intelligence (AI) course. Our approach is project-based and application-oriented. We present our experiences with this approach over a three-year period that includes six different offerings across three diverse institutions.

## 2. MACHINE LEARNING THEME

Many faculty members believe that an introductory AI course is challenging to teach because of the nature of the diverse and seemingly disconnected topics that are typically covered in the course [Kumar et al. 2006]. A number of workshops addressing the teaching of AI in the undergraduate curriculum have been organized. Going back to 1994, a symposium on Improving the Instruction of the Introductory AI course, sponsored by the American Association for Artificial Intelligence, was held in November 1994 in New Orleans. More recent workshops have addressed issues and challenges related to the teaching of AI related courses and topics in the curriculum [Dodds et al. 2006; Hearst 1995]. Recently, work has been done to address the diversity of topics covered to create a theme-based approach. Russell and Norvig [2003] present an agent-based approach. A number of instructors have been working to integrate robotics into the AI course [Dodds et al. 2006; Greenwald 2004; Harlan et al. 2001; Hearst 1995; Klassner 2006; Kumar and Meeden 1998].

Our approach incorporates machine learning as a unifying theme across the different AI topics to address this problem and to enhance student learning experiences in the introductory AI course. Our work involves the development, implementation, and testing of a suite of projects that can be closely integrated into a one-term AI course.

Machine learning (ML) provides a bridge between AI technology and modern software engineering. As Mitchell [2006] points out, machine learning is now considered as a technology for both software development (especially suitable for difficult-to-program applications or for customizing software) and building intelligent software (i.e., a tool for AI programming): (1) ML is becoming an increasingly important area of traditional computer science taught at the undergraduate level with applications that connect software development and classical AI areas, (2) the coverage of search algorithms in an AI course provides an ideal setting to easily expand such coverage to ML algorithms, and (3) machine learning provides excellent examples of heuristic approximation algorithms.

Some courses seek to use robotics to unify diverse AI topics, and multiple platforms currently simplify their use.[1] However, in an introductory course one

[1]For example, Lego Mindstorms (http://mindstorms.lego.com/), Myro (http://www.roboteducation.org/), and Pyrobot (http://www.pyrorobotics.org/) platforms.

generally wishes to impart a wide variety of topics efficiently, providing a sort of index to the major areas of the field. Although robotics has the advantage of rooting learning in application and experience, the overhead of robotic prototyping directly impacts the time available to cover a breadth of topics in an introductory AI course. By contrast, a machine learning application can be more rapidly prototyped, allowing learning to be grounded in engaging experience without limiting the important breadth of an introductory course. Physical robotics is both time-consuming and rewarding. Simulated robotic applications with machine learning would provide most of the benefit without the overhead of engineering hurdles which would limit an introductory AI course. While the approach in Nilsson's book [1998] takes an evolutionary, agent-based approach to topic unification with a heavy emphasis on machine learning, the text places almost no emphasis on application of ideas through implementation. In contrast, our approach allows for varying levels of mathematical sophistication with implementation of concepts being central to the learning process.

Simply put, there is a spectrum of learning approaches to the introductory AI course that range from broad textbook readings with written exercises to focused, detailed, and physically-grounded, real-world applications. For the authors, machine learning applications provide an excellent balance between the necessary overhead of good experiential learning and the worthy goal of broad topic coverage.

## 3. PROJECT GOALS AND OBJECTIVES

The AI course at all three institutions is offered at the junior and senior levels with data structures as a prerequisite. Typically, this course provides students with basic knowledge of the theory and practice of AI as a discipline concerning intelligent agents capable of making and enacting decisions. Our offerings had been largely consistent with traditional offerings in providing students with basic knowledge of the theory and practice of AI as a discipline. The course covers core AI topics such as search algorithms, knowledge representation, and reasoning. Traditionally, in addition to the core topics mentioned above, brief introductions to several subfields of AI have been provided. However, we believe that presenting an overview of many subfields provides students with superficial understanding of these areas. Instead, our approach would cover a smaller yet broad selection of subfields, connecting them via machine learning projects. Approximately two of fourteen weeks of class time is allotted to project coverage.

The project goal was to develop a framework for teaching core AI topics with a unifying theme of machine learning. The objectives were to:

—enhance the student learning experience in the AI course by implementing a unifying theme of machine learning to tie together the diverse and seemingly disconnected topics in the AI course;

—increase student interest and motivation to learn AI by providing a framework for the presentation of the major AI topics emphasizing the strong connection between AI and computer science;

—introduce students to an increasingly important research area, thus motivating them to pursue more advanced courses in machine learning and to pursue undergraduate research projects in this area; and

—increase student interest and motivation to build AI applications by allowing them to develop learning systems where they can implement the various concepts covered in the AI course.

To achieve these objectives, a suite of adaptable, hands-on laboratory projects were developed that can be closely integrated into a one-term AI course. Using machine learning as a unifying theme is an effective way to tie together the various AI concepts while at the same time emphasizing AI's strong tie to computer science. We focus on fundamental algorithms and knowledge representation.

Each project involves the design and implementation of a learning system which enhances a particular commonly-deployed AI application. In addition, the projects provide students with an opportunity to address not only core AI topics, but also many of the issues central to computer science, including algorithmic complexity and scalability problems. Instructors may select one of the projects to use in class or may wish to give students the option of selecting one from all or a subset of the projects based on the comfort level of the instructor. The class project is not intended to replace regularly scheduled assignments, but is intended to complement them. Adjustments in the number of these assignments may be needed so as to avoid overloading students.

The rich set of applications that students can choose from spans several areas including recommender systems, Web document classification, pattern recognition, data mining, and games. Studies have shown that the choice of context or problem domain of assignments and examples used in class can have a dramatic impact on student motivation and in turn on the quality of their learning [Wilensky 1991]. A problem domain that a student relates to and finds relevant leads to deeper understanding and hence smoother transfer to other domains, something that assessment of our work supported.

## 4. MACHINE LEARNING PROJECTS

We have developed and tested six hands-on laboratory projects that can be closely integrated into a one-semester AI course. Preliminary results on the various projects were published in Markov et al. [2006b] and Russell et al. [2005a, 2005b, 2003]. The following sections provide details of each of the projects. Sample solutions along with support code are also available, upon request, to all instructors using the material.[2]

### 4.1 Web Document Classification

4.1.1 *Introduction.* Along with search engines, topic directories (a.k.a., Web directories) are the most popular sites on the Web as they are usually

[2]Additional information and complete project descriptions are available at the project Web page http://uhaweb.hartford.edu/compsci/ccliphase1/.

provided to narrow searches. Topic directories organize Web pages in a hierarchical structure (i.e., taxonomy, ontology) according to their content.

The purpose of this structuring is twofold. First, it helps Web searches focus on the relevant collection of Web documents. The ultimate goal here is to organize the entire Web into a directory, where each Web page has its place in the hierarchy and thus can be easily identified and accessed. The Open Directory Project and About are some of the best-known projects in this area.[3]

Second, the topic directories can be used to classify Web pages or associate them with known topics. This "tagging" process can be used to extend the directories themselves. In fact, such well-known search engines as Yahoo and Google may return with their responses the topic path of the response, if the response URL has been associated with some topic found in a topic directory. As these topic directories are usually created manually they cannot capture all URLs, therefore just a fraction of all responses are tagged.

4.1.2 *Project Objectives.* While working on this project, students will learn the basics of information retrieval, data mining, and machine learning; gain experience in using recent software applications; and most importantly have a better understanding of the role that fundamental AI concepts as knowledge representation and search play in these areas.

While reinforcing traditional AI core topics within a single, unified task of Web document classification, the project allows the discussion of various issues related to machine learning, including:

—basic concepts and techniques of machine learning,
—learning system implementation issues,
—the role of learning in improved performance and in allowing a system to adapt based on previous experiences,
—the important role data preparation and feature extraction play in machine learning,
—the vector space model for representing Web documents,
—feature extraction techniques with associated pros and cons for identifying and classifying documents, and
—the importance of model evaluation in machine learning and, in particular, the training and testing framework used to choose the best model for Web page classification.

4.1.3 *Project Description.* The aim of the project is to investigate the process of tagging Web pages using the topic directory structures and apply ML techniques for automatic tagging or classifying Web pages into topic categories. This would help filter search engine responses or rank them according to their relevance to a user-specified topic. For example, a Yahoo

[3]See dmoz.org and about.com.

keyword search for "Machine Learning" may return topic directory paths along with the pages found.

Category: Artificial Intelligence > Machine Learning
Category: Artificial Intelligence > Web Directories
Category: Maryland > Baltimore > Johns Hopkins University > Courses

However, most of the pages returned are not tagged with directory topics. Assuming that we know the general topic of such an untagged Web page, for example, artificial intelligence, and this is a topic in a directory, we can try to find the closest subtopic to the Web page found. This is where machine learning comes into play. Using some text document classification techniques we can classify the new Web page to one of the existing topics. By using the collection of pages available under each topic as examples we can create category descriptions (e.g., classification rules or conditional probabilities). Using these descriptions, we can then classify new Web pages.

Another approach would be the similarity search approach, where we find the closest text document according to some metric, and assign its category to the new Web page. The project is split into three major parts. These parts are also phases in the overall process of knowledge extraction from the Web and classification of Web documents (tagging). Since this process is interactive and iterative in nature, the phases may be included in a loop structure allowing each phase to be revisited so that feedback from earlier phases can be used. The parts are well-defined, modular, and can be combined manually or semi-automatically. Hereafter, we describe the project phases along with the deliverables that the students need to document in the final report for each phase.

### Phase 1: Collecting sets of Web documents grouped by topic

The purpose of this phase is to collect sets of Web documents belonging to different topics (subject areas). The student begins by examining a topic directory structure. Such structures are available from various Web directories.[4] Students first find several topics (e.g., 5), each of which is well-represented by a set of at least 20 documents. Alternative approaches would be to extract Web documents manually from a list of search engine hits from a general keyword searches, or collect Web pages by using a Web Crawler from the Web page structure of a large organization (e.g., university).

The outcome of this phase is a collection of several sets of Web documents representing different topics or subjects, where the following restrictions apply.

(a) As these topics will be used for learning and classification experiments at later phases they have to form a specific structure (part of the topic hierarchy). It's good to have topics at different levels of the topic hierarchy

[4]For example, dmoz.org (the Open Directory project), the Yahoo! directory (dir.yahoo.com), About.com and http://en.wikipedia.org/wiki/List_of_web_directories.

and with different distances between them (i.e., different depths in the hierarchy tree). An example of such structure is the following.

topic1 > topic2 > topic3
topic1 > topic2 > topic4
topic1 > topic5 > topic6
topic1 > topic7 > topic8
topic1 > topic9

The set of topics here is formed by the leaves of the tree which are topic3, topic4, topic6, topic8, and topic9. Also, it would be interesting to consider a more general graph structure where topics may have more than one parent, as this would make the classification task more difficult.

(b) There must be at least five different topics with at least 20 documents in each.

(c) Each document should contain a certain minimum amount of text, for example, 200 words excluding stopwords and punctuation marks.

In our experiences with this project, students easily understand this phase and prepare the initial data well without much guidance. This phase also provides an opportunity to connect to other CS and AI topics such as trees, graphs, semantic networks, and ontologies. For example, an interesting question that students may be asked to investigate is the possibility of cycles in the topic hierarchy. This question is related to the generality/specificity of topics and may be also considered in the context of set-subset relationships.

### Phase 2: Feature extraction and data preparation

In this phase, Web documents are represented by feature vectors, which in turn are used to form a training data set for the machine learning phase. This conversion actually illustrates the basic concepts of the vector space document model that plays an important role in information retrieval and Web search. Various approaches and tools may be used to achieve the goals of this phase in the project. However, we recommend using the Weka machine learning system in this and later phases. Although it is a large software suite, it is easy to install and use with an intuitive graphical interface. Learning to use the whole functionality of the system might overwhelm students in a one-semester course. However, for the purposes of this project only a small subset of functions is needed. Students may easily learn these functions on their own using online material, but in this phase we would especially recommend faculty assistance. Our experience shows that this may be done in a one or two hour lab that may also be used as a demo session for the ML part of the AI class.

The basic steps that have to be taken at this phase are the following.

(1) Download and install Weka.[5] It comes with documentation and helpful examples to familiarize students with its use. A suggested exercise for this

[5]See http://www.cs.waikato.ac.nz/ml/weka/

step is experimenting with the weather data set, which is also a classical example for the ML part in the AI course.

(2) Create an initial representation of each document in string data format: This is a plain text file containing the concatenation of all of a Web document's text corpus in a Weka ARFF data file.

(3) Create training data sets by using the string data file from the previous step: These data sets use the three basic ways to represent document vector: Boolean (using only the presence or absence of a term in the document), term counts (taking into account the term frequency in a document), and TFIDF (term frequencies combined with the number of term occurrences across different documents in the corpus).

As a result of this phase, students have to provide ARFF data files containing the feature vectors for all Web documents collected at Phase 1. We recommend that students prepare several files using different approaches to feature extraction, for example, one with Boolean attributes and one with numeric ones created by applying the TFIDF approach. Versions of the data sets with different numbers of attributes can be also prepared.

A suggested reading for this phase is Chapter 1 of the book [Markov and Larose 2007]. This chapter discusses the basics of information retrieval and provides a set of exercises explaining the details of all steps needed to complete this phase of the project. Example datasets as well as Weka files for the string, Boolean, term count, and TFIDF representation formatted as ARFF files are available from the companion Web site of the book and also from the authors' Web page.[6]

To complete this phase, in addition to Weka, other software tools for text processing are also needed, especially at Step 2 in creating the string file with the text corpus. Students are usually provided with suggested approaches and tools for this purpose (for example, using MS Word to concatenate documents and convert them into plan text format). They are also given a choice to use any other software they may find suitable for the task. Our experience with this project shows that students at this level generally have the required background and skills to find and use the software they need. (Some even find this step interesting and challenging and write their own programs for text processing.) However, concerning Weka, we believe more faculty guidance is needed. Students find the Weka documentation insufficient and often ask for help. As we mentioned above, an hour or two hands-on lab usually suffices to fill this gap and to allow students to use Weka successfully in other phases of the project and for other Weka-based projects beyond.

### Phase 3: Machine learning

At this phase, machine learning algorithms are used to create models of the data sets. These models are then used for two purposes: (1) the accuracy of the initial topic structure is evaluated, and (2) new Web documents are classified

[6]http://www.dataminingconsultant.com/DMW.htm and
http://www.cs.ccsu.edu/~markov/dmwdata.zip.

into existing topics. The Weka system is used for both purposes. The ML phase consists of the following steps.

(1) *Preprocess Web document data*. ARFF files created at project Phase 2 have to be verified for consistency and analyzed by using the preprocess panel.
(2) *Create decision tree models with Weka's decision tree algorithm (J48)*. Good questions to ask students at this step are: Which are the most important terms for each data set (the terms appearing on the top of the tree)? How do they change with changing the data set? Also, have students check the classification accuracy and the confusion matrix obtained with 10-fold cross-validation in order to find out which topic is best represented by the decision tree.
(3) *Apply the Naïve Bayes and Nearest-Neighbor (IBk) algorithms*. Students compare the classification accuracy and confusion matrices obtained with 10-fold cross-validation from these algorithms with the ones produced by the decision tree in order to evaluate which is superior and provide an explanation.
(4) *Experiment with clustering*. Three basic clustering algorithms, *k*-means, EM, and Cobweb are applied to all data sets where the class attribute (document topic) is ignored. The resulting clusters are then analyzed and compared with the original set of topics, or with the topic hierarchy when using Cobweb. Weka's classes to clusters evaluation method may be also used for this purpose. A data mining lab that provides useful information and tips about this step is available online.[7]
(5) *Classify new Web documents*. Students find Web documents from the same subject areas (topics), but not belonging to the original set of documents prepared in project Phase 1. Documents from different topics are also needed. Then, feature extraction is applied and ARFF files are created for each document. Using the Weka test set option the new documents are classified and their original topic is compared with the one predicted by Weka. Additional online material provides guidelines for this step.[8]

In this phase of the project, students write a report on the experiments performed, including detailed descriptions of the experiments (input data, Weka outputs), answers to the questions, and interpretation and analysis of the results with respect to the original problem stated in the project: Web document classification.

## 4.2 Web User Profiling

4.2.1 *Introduction.* The Web is the largest collection of electronically accessible documents, making it the richest source of information in the world. However, one of the greatest problems of the Web is that this information is not well-structured and organized so that it can be easily retrieved. Search

[7]See http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-Ex3.html.
[8]See http://www.cs.ccsu.edu/~markov/ccsu_courses/DataMining-Ex2.doc.

engines help in accessing Web documents by keywords, but this is still far from what we need in order to effectively use the knowledge available on the Web. Machine learning and data mining approaches go further, extracting knowledge from the raw data available on the Web by organizing Web pages in well defined structures or by observing patterns of Web user activity. This project focuses on this challenge and explores the applicable ML techniques.

Web searches provide large amounts of information about Web users. Data mining techniques can be used to analyze this information and create Web user profiles. A key application of this approach is in marketing and offering personalized services, an area referred to as the "data gold rush".

4.2.2 *Project Objectives.* The aim of this project is to develop a system that helps us develop an intelligent Web browser. The project will focus on the use of decision tree learning to create models of Web users. Students will be provided with decision tree learning tools and will collect data from Web searches. They will then experiment with creating Web user models and use these models in order to improve the efficiency of Web searches performed by the same or new users. The learning objectives of the project are to

—learn the basics of information retrieval and machine learning,
—gain experience with recent software applications in these areas, and
—gain better understanding of fundamental AI concepts such as knowledge representation and search.

4.2.3 *Project Description.* As in the Web document classification project (see Section 4.1) this project is split into three major parts/phases: data collection, feature extraction and machine learning (mining). At the data collection and feature extraction phases, Web pages (documents) are collected and represented as feature vectors. Unlike the document classification project, documents are now mapped to users rather than topic categories. At the machine learning phase, various learning algorithms are applied to feature vectors in order to create user models for vector (document) mapping. Then, such models can be used to filter Web documents returned by searches, providing users more focused information. Users can also be identified by their preferences, and new users can be classified accordingly. We next briefly describe the project phases.

### Phase 1: Collecting sets of Web documents grouped by user preference

In this phase, students collect a set of Web documents labeled with user preferences. First, a user performs Web searches with simple keyword search, just browses the Web or examines a set of pages collected by a Web crawler. To each Web document the user assigns a label representing whether or not the document interests the user. As in the Web document classification project, some restrictions apply: (1) the number of Web pages should be greater than the number of selected features; (2) the Web pages should have sufficient text content so that they could be well described by feature vectors.

### Phase 2: Feature extraction and data preparation

This phase is very similar to the one described in the Web Document Classification project. By using the Weka filters, Boolean or numeric values are calculated for each Web document and the corresponding feature vector is created. Finally, vectors are included in the ARFF file for input to the Weka system. Note that at this last step the vectors are extended with class labels (e.g., interesting/non-interesting or +/-) according to the user preferences. As in the Web document classification project, the outcome of this phase is an ARFF data file containing the feature vectors for all Web documents collected at Phase 1. As before, we recommend that students prepare several files using different approaches to feature extraction - Boolean attributes, numeric attributes (using the TFIDF approach) and with a different number of terms. We wish students to do more experiments with different data sets and different ML algorithms in order to find the best user model.

### Phase 3: Machine learning

At this phase, the approaches and experiments are similar to those described in the Web Document Classification project with an important difference in the last step where the machine learning models are used. This step can be called "Web document filtering" (i.e., focusing the search) and can be described as follows: A number of Web documents are collected using one of the approaches suggested in Phase 1. Feature extraction is applied, and an ARFF test file is created with one data row for each document. Using the training set prepared in Phase 2 and the Weka's test set option, new documents are classified so that each one maps to a corresponding label (interesting/non-interesting or +/-). Finally, non-interesting documents are discarded and interesting ones are presented to the user. This project may be extended by incorporating this step into a Web browser, so that it automatically labels all Web pages as interesting/non-interesting according to the user's preferences.

## 4.3 Neural Network Character Recognition

4.3.1 *Introduction.* Neural networks have been applied to a wide variety of areas including speech synthesis, diagnostic problems, medicine, business and finance, robotics control, and signal processing. Neural networks have been shown to be particularly useful in solving problems where traditional artificial intelligence techniques involving symbolic methods have failed or proved inefficient. Such networks have shown promise in problems involving low-level tasks that are computationally intensive, including vision, character recognition, speech recognition, and many other problems that fall under the category of pattern recognition. For some application areas including pattern recognition, neural models show promise in achieving human-like performance over more traditional artificial intelligence techniques.

This project introduces students to basic neural network concepts and to neural models and learning. Students implement basic types of neural networks as well as some important approaches to learning in order to solve a number of typical pattern recognition problems. Except for the character recognition system, students work out the weight values by hand. For the character recognition system, students are given an option of either developing their own implementation or using some of the available tools such as those of MATLAB or free open source libraries such as those of Generation 5 or Fast Artificial Neural Network.

4.3.2 *Project Objectives.* The aim of this project is to provide students with knowledge and skills to understand implement and use basic neural network models. This will allow students to understand the role that neural networks play in the more general context of AI techniques and tools. In particular, the learning objectives of the project are to

—learn the basics of single-layer neural networks and perceptron type models and their use as pattern associators,
—understand the limitations of single-layer networks and introduce the basic types of multilayer networks as well as the backpropagation training algorithm,
—learn the mathematical foundations of the neural network computation,
—better understand the nature of problems that neural networks can solve,
—gain experience in implementing neural network algorithms for solving basic pattern recognition problems, and
—better understand the differences between the neural network approach to solving AI problems and those based on classical symbolic knowledge representation, search, and learning.

4.3.3 *Project Description.* Artificial neural networks are computer systems that attempt to simulate the structure of the neural connections in order to achieve parallelism. A neural network is made up of highly-interconnected processing elements that respond to a set of input signals.

The goal of this project is to develop a character recognition system based on a neural network model. To achieve this goal, students go through several steps of creating different neural network models:

(1) Create single-layer linear networks (pattern associators) and model a single-layer network using Hebbian learning.
(2) Create threshold networks (perceptrons) and model a single-layer network using the delta rule.
(3) Implement a basic character recognition system that involves two steps:
    (a) Using the delta learning rule, a zero initial weight vector, and a threshold value of 0.5, students implement a single layer network that can be trained to recognize the letters A and B. Letters should be represented by a binary matrix and hence the input layer should consist of 25 input units. Students must figure out the necessary number of output units (iterations) for the network to recognize the two letters. They must also show the values of the weight matrix, as well as the output values after each iteration.

   (b) Students write and test a program that implements the character recognition system described above. The program should read in the threshold value and the number of iterations to be used. Students then explain what test data they used and write a report on the results of testing and classification of the input patterns.

(4) Compute the XOR problem using a multilayer network: In this step, students design a multilayer network with one hidden layer that can compute the XOR problem, explaining the significance of the values of the corresponding weight vector and which threshold value they used.

(5) Write a final report commenting on the relation of the approaches used in this project to the areas of search and knowledge representation and reasoning.

## 4.4 Solving the *N*-Puzzle Problem

4.4.1 *Introduction.* The *N*-puzzle game provides a good framework for illustrating conceptual AI search in an interesting and motivating way. Various uninformed and informed search algorithms are usually applied in this setting and their performance is evaluated. In the eight-puzzle version of the game, a 3 × 3 board consists of eight tiles numbered 1 through 8 and an empty square. One may move any tile into an orthogonally adjacent empty square, but may not move outside the board or diagonally. The problem is to find a sequence of moves that transforms an initial board configuration into a specified goal configuration. The following are examples of initial and goal configurations:

Initial configuration

| | | |
|---|---|---|
| 1 | 6 | 2 |
| 5 | 7 | 3 |
| | 4 | 8 |

Goal configuration

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

The domain theory of the *N*-puzzle problem can be expressed by a set of facts describing state transitions, and a search engine that can be used to find paths between initial and goal states. Given a pair of initial and goal states (a training example), the search algorithm finds the shortest path between them (explanation or proof). Then applying the Explanation-Based Learning (EBL) techniques, the path is generalized so that it can be used later to match other initial states and bring the search algorithm directly to the goal state, without the resource-consuming exploration of the huge state space of the game. With carefully chosen training examples, useful rules for typical moves can be learned and then integrated into the search algorithm to achieve better performance.

4.4.2 *Project Objectives.* In this project, students are asked to incorporate explanation-based learning (EBL) into the *N*-puzzle problem. This allows them to better understand the concepts of analytical learning, and to see how learning improves the performance of search algorithms. The goal is to introduce the student to analytical (explanation-based) learning using the classical AI framework of search. Hands-on experiments with search algorithms combined with an EBL component give the student a deep, experiential understanding of the basics of EBL and the role it plays in improving the performance of search algorithms in particular and problem solving approaches in general.

4.4.3 *Project Description.* Many learning algorithms (usually considered as inductive learning) generalize on the basis of regularities in training data. These algorithms are often referred to as similarity-based. Generalization is primarily determined by the syntactic structure of the training examples, and the use of domain knowledge is limited to specifying the hypothesis language and exploring the hierarchy of the attribute values. A learning system typically uses domain knowledge and is expected to have some ability to solve problems. Thus, the objective of learning is to improve the system's knowledge or performance by use of domain knowledge. This task can be seen as knowledge reformulation or theory revision.

EBL uses a domain theory to construct an explanation of the training example, usually a proof that the example logically follows from the theory. Using this proof, the system filters the noise, selects only the aspects of the domain theory relevant to the proof and organizes the training data into a systematic structure. This makes the system more efficient in later attempts to deal with the same or similar examples. The basic components in EBL are the following.

—*Target concept*. The task of the learning system is to find an effective definition of this concept. Depending on the specific application the target concept could be a classification rule, theorem to be proven, a plan for achieving goal, or heuristic to make a problem solver more efficient (e.g., a state space search heuristic).

—*Training example*. This is an instance of the target concept. For example, this may be a good (efficient) solution in a state space search.

—*Domain theory*. Usually, this is a set of rules and facts representing domain knowledge. They are used to explain how the training example is an instance of the target concept.

—*Operationality criteria*. This is a means to specify the form of the concept definition. That is, this is the language of expressing the target concept definition, which is usually a part of the language used in the domain theory.

In the form outlined above, EBL can be seen as partial evaluation. In terms of theorem-proving, this technique is also called unfolding, that is, replacing body goals with the bodies of the rules they match, following the order in which goals are reduced (depth-first). Hence, in its pure form, EBL doesn't learn anything new, that is, all the rules inferred belong to the deductive closure of the domain theory. This means that these rules can be inferred from the theory without using the training example at all. The role of the training example is only to focus the theorem prover on relevant aspects of the problem domain. Therefore EBL is often viewed as a form of speed-up learning or knowledge reformulation. Consequently EBL can be viewed not as a form of generalization, but rather as specialization, because the rule produced is more specific than the theory itself (the EBL rule is applicable to only one example).

There are small and well defined theories, however practically inapplicable. For example, consider the game of chess. The rules of chess combined with an ability to perform unlimited look-ahead on the board states will allow a system to play well. Unfortunately this approach is impractical. An EBL system, given well-chosen training examples, will not add anything new to the rules of chess playing, but will actually learn heuristics to apply these rules, which might be practically useful. The *N*-puzzle domain is another typical example of this approach. As the search space is huge, any practical solution requires heuristics. And the role of EBL is to learn such heuristics from examples of successful searches.

To simplify the discussion hereafter we discuss the five-puzzle problem. In this representation tiles are numbered from 1 to 5. The empty square (no tile) is represented by 0. The state of the game is represented by a list of tiles (including 0), where their position in the list corresponds to their board position. For example, (1 2 3 4 5 0) correspond to the following board.

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 0 |

## State transitions

The state transitions are represented by reordering tiles in the list. For this purpose we use variables representing board positions, which hold the actual tile numbers.

For example, moving the empty tile from position 1 to position 2 is represented as transforming state

```
  (first second third fourth fifth sixth)
```

into state

```
  (second first third fourth fifth sixth),
```

where all list elements except for first (which holds the 0) can take actual tile numbers from 1 to 5 (all different). Thus, this generalized transition represents actual transitions between game states.

Depending on the position of the empty tile (0), we may have two or three possible transitions of the type discussed above. In LISP, this is implemented as a function, which adds the new states in the beginning of the list of current states.

Here is an example of extending state (0 1 2 3 4 5) by using such a function called move-1:

```
> (move-1 (list 0 1 2 3 4 5))
((3 1 2 0 4 5) (1 0 2 3 4 5) (0 1 2 3 4 5))
```

## Search algorithm

Any uninformed search algorithm that is able to find the shortest path in a graph may be used here. As the goal of EBL is to improve the efficiency, the algorithm can be a simple one. Iterative deepening and breadth-first search are good choices because they have high computational complexity. Thus, after the EBL step, the speed-up would be easily measured by the reduction of the size of the path between initial and goal states and run time and memory usage. Here is an example of solving the five-puzzle with breadth-first search (the printout shows the path between start and goal found by the algorithm):

```
> (setf start '(4 5 3 0 1 2))
> (setf  goal '(1 2 3 4 5 0))
> (breadth-first start goal)
((4 5 3 0 1 2) (0 5 3 4 1 2) (5 0 3 4 1 2)
 (5 1 3 4 0 2) (5 1 3 4 2 0) (5 1 0 4 2 3)
 (5 0 1 4 2 3) (0 5 1 4 2 3) (4 5 1 0 2 3)
 (4 5 1 2 0 3) (4 0 1 2 5 3) (4 1 0 2 5 3)
 (4 1 3 2 5 0) (4 1 3 2 0 5) (4 1 3 0 2 5)
 (0 1 3 4 2 5) (1 0 3 4 2 5) (1 2 3 4 0 5)
 (1 2 3 4 5 0))
```

### Training example and target concept

First we specify a training example, as a pair of start and goal states. Let us consider the transition from state (4 5 3 0 1 2) to (5 1 3 4 2 0). According to the EBL principles, the training example is an instance of the target concept. So, we have to run the algorithm in order to verify that this is a correct training example, that is, it is an instance of a correct target concept:

```
> (setf start '(4 5 3 0 1 2))
> (setf goal '(5 1 3 4 2 0 ))
> (breadth-first start goal)
((4 5 3 0 1 2) (0 5 3 4 1 2) (5 0 3 4 1 2)
 (5 1 3 4 0 2) (5 1 3 4 2 0))
```

### EBL generalization

In our setting, EBL generalization simply substitutes constants for variables. Following the representation adopted here, we create a new generalized transition from state

```
  (first second third fourth fifth sixth)
```

to state

```
  (second fifth third first sixth fourth),
```

where the following substitutions apply:

```
  first = 4, second = 5, third = 3
  fourth = 0, fifth = 1, sixth = 2
```

### Improving the search (improving the domain theory)

The last step in EBL is to add the new target concept definition to the domain theory. In the particular example, this means defining a new function that will allow the search algorithm to use the new state transition. In our LISP implementation, we have to modify the move-1 function in order to add the new state to the resulting list.

It is important to note that the new state transition generated by EBL should be used first by the search algorithm. We achieve this by adding the new state as a first element of the path extension.

To preserve the completeness of the algorithm (in EBL terms: completeness of the theory), the new transitions should not replace the original basic ones (one-tile move). Rather, it should be just added, thus expanding the search space with new transitions. This has to be implemented in our new move-1 function too, so that it returns also the state transitions based on one-tile moves.

The newly learned EBL state transition may represent useful search heuristics. To achieve this, however, the training examples have to be carefully chosen. They should represent expert strategies to solve the game or at least pieces of such strategies. In fact, our training example was chosen with this idea in mind. Thus, the newly-learned concept (incorporated in the new move-1) improves the efficiency of the algorithm. This can be shown with the same pair of start and goal states that produced a path of 19 states with the standard breadth-first search. Now the path has only 13 states, which means that the new transition is used twice during the search.

```
> (setf start '(4 5 3 0 1 2))
> (setf goal '(1 2 3 4 5 0))
> (breadth-first start goal)
((4 5 3 0 1 2) (5 1 3 4 2 0) (5 1 0 4 2 3)
 (5 0 1 4 2 3) (0 5 1 4 2 3) (4 5 1 0 2 3)
 (4 5 1 2 0 3) (4 0 1 2 5 3) (4 1 0 2 5 3)
 (4 1 3 2 5 0) (4 1 3 2 0 5) (4 1 3 0 2 5)
 (1 2 3 4 5 0))
```

## Student Projects

Based on the framework described, the following student projects are suggested.

—Implement uninformed (e.g., breadth-first and iterative deepening) search algorithms in LISP or Prolog.

—Identify useful search heuristics, generating and verifying the corresponding EBL training examples.

—Perform experiments with training examples, measuring the improvement in terms of run time and memory requirements after the EBL step.

—Implement LISP or Prolog modules to automatically update the theory given a training example. This includes the EBL generalization step and incorporation of new transition rules into the search algorithm.

—Evaluate the effect of learning if too many or bad examples are supplied.

—Solving the eight-puzzle: (2 3 5 0 1 4 6 7 8) $\rightarrow$ (0 1 2 3 4 5 6 7 8)
  —Compare breadth-first, iterative deepening, and depth-first search.

—Explain why depth-first search fails.
—Find a board state that can be solved.

These projects are based in most part on classical AI search. Our experience shows that the EBL component does not require major effort by the student to understand the concepts and to implement the algorithms. This allows the projects to be offered as extensions of traditionally used in the AI curriculum projects to cover the topic of search. In most cases students easily add the EBL component to existing implementations of classical search algorithms and, by running the suggested experiments, quickly understand the basics of analytical learning. Faculty who have used the projects were able to integrate them in the AI curriculum without much overhead, because in most cases there was no need to add new material to the AI curriculum, provide specialized software or teach extra labs.

## 4.5 Solving the Dice Game Pig

4.5.1 *Introduction.* The jeopardy dice game Pig is very simple to describe, yet the optimal policy for play is far from trivial. Using the computation of Pig's optimal roll/hold policy as a central challenge problem, this project introduces students to core concepts of reinforcement learning, such as Markov decision processes (MDPs) and simple algorithms for computing optimal policies. Advanced undergraduate research projects are suggested as well.

4.5.2 *Project Objectives.* The object of this project is to give the student a deep, experiential understanding of dynamic programming and value iteration through explanation, implementation examples, and implementation exercises.

*Dynamic Programming:*

—Demonstrate the need for dynamic programming through Fibonacci number computation.
—Guide the student through the details of a Java solution to a simple Pig variant with an acyclic state space.
—Provide problem solving exercises that will ground the student's understanding of dynamic programming in experience.

*Value Iteration:*

—Introduce the method of value iteration.
—Demonstrate its application to a simple coin variant of Pig called Piglet.
—Guide the student through the details of a Java solution to Piglet.
—Provide problem solving exercises that will ground the student's understanding of value iteration in experience.

4.5.3 *Project Description.* The object of Pig is to be the first player to reach 100 points. Each turn, a player repeatedly rolls a die until either

—the player rolls a 1 (“pig”), the turn ends, and the player scores nothing, or

—the player holds, the turn ends, and the player scores the sum of the rolls (i.e., the turn total).

At any time during a player's turn, the player is faced with two choices: *roll* or *hold*. The decision whether to roll or hold depends on the risk/reward of potentially losing/improving the turn total with an additional roll. Note that a "pig" roll of 1 results in the loss of the turn total and does not affect the score from previous turns.

Pig has long been used by mathematicians to teach concepts of probability. In recent years, it has also found use across the computer science curriculum [Neller et al. 2006a]. In the following sections, we focus on its use in teaching fundamental reinforcement learning concepts from [Russell and Norvig 2003; Sutton and Barto 1998].

## Optimal Play of Pig

Knizia [1999] describes the odds-based analysis for a "hold at 20" policy, where each roll is viewed as a bet:

> ... we know that the true odds of such a bet are 1 to 5. If you ask yourself how much you should risk, you need to know how much there is to gain. A successful throw produces one of the numbers 2, 3, 4, 5, and 6. On average, you will gain four points. If you put 20 points at stake this brings the odds to 4 to 20, that is 1 to 5, and makes a fair game. ... Whenever your accumulated points are less than 20, you should continue throwing, because the odds are in your favor.

Although the "hold at 20" policy maximizes expected points per turn, this is far from optimal play. Playing to score is not the same as playing to win. Indeed, to win more often, one must be willing to lose with a lower score, taking great risks when the opponent has a great score advantage.

At any time, when faced with the decision to roll or hold, one should choose the action that maximizes the expected probability of winning. Let $P_{i,j,k}$ be the player's probability of winning if the player's score is $i$, the opponent's score is $j$, and the player's turn total is $k$. In the case where $i + k \geq 100$, $P_{i,j,k} = 1$ because the player can simply hold and win. In the general case where $0 \leq i, j < 100$ and $k < 100 - i$, the probability of an optimal player winning is

$$P_{i,j,k} = \max(P_{i,j,k,roll}, P_{i,j,k,hold}),$$

where $P_{i,j,k,roll}$ and $P_{i,j,k,hold}$ are the probabilities of winning if one rolls and holds, respectively. These probabilities are given by:

$$\begin{aligned} P_{i,j,k,roll} &= \frac{1}{6}\left((1 - P_{j,i,0}) + \sum_{r=2}^{6} P_{i,j,k+r}\right) \\ P_{i,j,k,hold} &= 1 - P_{j,i+k,0} \end{aligned}$$

The probability of winning after rolling a 1 or holding is the probability that the other player will not win beginning with the next turn. All other outcomes are positive and dependent on the probabilities of winning with higher turn totals.

### Dynamic Programming

Sutton and Barto [1998] describe three fundamental classes of algorithms for reinforcement learning: dynamic programming (DP), Monte Carlo methods, and temporal-difference (TD) learning. TD learning can be viewed as a hybrid of DP and Monte Carlo methods, so DP methods are a natural starting point for teaching reinforcement learning.

Sutton and Barto define dynamic programming (DP) as a collection of algorithms for solving Markov decision processes (MDPs) [Sutton and Barto 1998, p. 89], including the value iteration algorithm. However, in computer science, the term dynamic programming commonly refers more generally beyond MDPs to a common programming pattern where one "[solves] an optimization problem by caching subproblem solutions (memoization) rather than recomputing them" [Cormen et al. 2001].[9]

This important general programming pattern is usually taught in a data structures and algorithms course, but we use it as a bridge for teaching value iteration. In Cormen et al. [2001], the development of a dynamic programming algorithm is described as having four steps as follows.

(1) Characterize the structure of an optimal solution.
(2) Recursively define the value of an optimal solution.
(3) Compute the value of an optimal solution in a bottom-up fashion.
(4) Construct an optimal solution from computed information.

This is, in short, a recursive solution with caching and reuse of subproblem solutions.

A recursive solution structure assumes that subproblem dependencies are acyclic. However, equations for optimal play of Pig are cyclic. Suppose two players roll a 1 in succession. Then nothing has changed about the state of the game; players have cycled back to the same situation. Such cyclic possibilities in play mean that we cannot solve optimal play of Pig using recursion.

However, a slight modification to the rules of Pig, described in Boyan [1998], yields an acyclic set of equations for optimal play. We call this variation Progressive Pig, because one progresses towards the score goal each turn. Progressive Pig is identical to Pig except that a player always scores at least 1 point each turn as follows.

—If the player rolls a 1, the player scores 1 point and it becomes the opponent's turn.
—If the player rolls a number other than 1, the number is added to the player's turn total and the player's turn continues.

[9]One can view value iteration as a type of dynamic programming if the cached subproblem solutions are state value estimates indexed by iteration, and the number of iterations is fixed.

—If the player holds, the greater of 1 and the turn total is added to the player's score and it becomes the opponent's turn.

Thus the equations for the probability of winning Progressive Pig with optimal play are:

$$
\begin{aligned}
P'_{i,j,k} &= \max\left(P'_{i,j,k,roll}, P'_{i,j,k,hold}\right),\\
P'_{i,j,k,roll} &= \frac{1}{6}\left(\left(1 - P'_{j,i+1,0}\right) + \sum_{r=2}^{6} P'_{i,j,k+r}\right),\\
P'_{i,j,k,hold} &= 1 - P'_{j,i+\max(k,1),0}.
\end{aligned}
$$

In our curricular materials, we first illustrate the DP programming pattern in Java with the computation of Fibonacci numbers. Then we further illustrate DP with the computation of optimal play for Progressive Pig, as an approximation to optimal Pig play. We then challenge students to apply DP to the computation of optimal play for the following Pig-related exercises:

—*Pig Solitaire*. To win, reach a given goal score $g$ within $n$ turns.

—*Pig Solitaire 2*. Maximize one's score in $n$ turns.[10]

—*THINK Solitaire*. Compute a policy to maximize expected score in the game of THINK. THINK is identical to Pig, except that
  —two standard dice are rolled. If neither shows a 1, their sum is added to the turn total.
  —if a single 1 is rolled, the player's turn ends with the loss of the turn total.
  —if two 1's are rolled, the player's turn ends with the loss of the turn total and score.
  —each player gets only five turns, one for each letter of THINK.
  —the highest score at the end of five turns wins.

## Value Iteration

The value iteration algorithm iteratively improves estimates of the value of being in each state. In describing value iteration, we follow Sutton and Barto [1998], which we also recommend for further reading. We assume that the world consists of states, actions, and rewards. The goal is to compute which action to take in each state so as to maximize future rewards. At any time, we are in a known state $s$ of a finite set of states $\mathcal{S}$. There is a finite set of actions $\mathcal{A}$ that can be taken in any state. For any two states $s, s' \in \mathcal{S}$ and any action $a \in \mathcal{A}$, there is a probability $P^a_{s,s'}$ (possibly zero) that taking action will cause a transition to state $s'$. For each such transition, there is an expected immediate *reward* $R^a_{s,s'}$.

We are not just interested in the immediate rewards; we are also interested to some extent in future rewards. More specifically, the value of an action's result is the sum of the immediate reward plus some fraction of the future reward. The discount factor $0 \leq \gamma \leq 1$ determines how much we care about the expected future reward when selecting an action.

[10]This is confirmation of the "hold at 20" policy in disguise.

Let $V(s)$ denote the estimated value of being in state $s$, based on the expected immediate rewards of actions and the estimated values of being in subsequent states. The estimated value of an action $a$ in state is given by

$$\sum_{s'} P^a_{s,s'} \left[R^a_{s,s'} + \gamma\, V(s')\right].$$

Since any action can be chosen, the optimal choice is the action that maximizes this estimated value:

$$\max_a \sum_{s'} P^a_{s,s'} \left[R^a_{s,s'} + \gamma\, V(s')\right].$$

This expression serves as an estimate of the value of being in state $s$, that is, $V(s)$. In a nutshell, value iteration consists of revising the estimated values of states until they converge, that is, until no single estimate is changed significantly. The value iteration algorithm is as follows.

**Value Iteration Algorithm**

```
For each $s \in \mathcal{S}$, initialize $V(s)$ arbitrarily.
Repeat
  $\Delta \leftarrow 0$
        For each $s \in \mathcal{S}$,
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} P^a_{s,s'} \left[R^a_{s,s'} + \gamma\, V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \epsilon$
```

Value iteration repeatedly updates estimates of $V(s)$ for each $s$. The variable $\Delta$ is used to keep track of the largest change for each iteration, and $\epsilon$ is a small constant. When the largest estimate change $\Delta$ is smaller than $\epsilon$, we stop revising our estimates.

When $\gamma = 1$, and all rewards are 0 except for game winning state transitions with a reward of 1, then state values are identical to win probabilities. The computed policy is optimal, maximizing the expected probability of winning against perfect competition. Value iteration's convergence, however, is not guaranteed in general when $\gamma = 1$. Convergence is guaranteed only when $\gamma < 1$ and rewards are bounded [Mitchell 1997, §13.4]. However, in the cases of Piglet and Pig, value iteration happens to converge.

In order to apply value iteration to Piglet, we reduce the two-agent problem to a single agent problem through our previous observation that the probability of winning after the end of a turn is the probability of the opponent losing. That is, the probability of winning after the turn's end is 1 minus the probability that we would win as the other player. With this small modification to the Bellman equations, we illustrate the solution to optimal play for Piglet in Java. We then provide the following exercises.

—*Pig*. Compute optimal play for Pig.

—*Pig Solitaire 3*. Compute optimal play to minimize the number of turns taken to reach a given goal score *g*.

—*Pass the Pigs*. Compute optimal play for Pass the Pigs®. Pass the Pigs® (a.k.a., Pigmania®) is a popular commercial variant of Pig which involves rolling two rubber pigs to determine the change in turn total or score.[11] Students are given roll data for estimating roll probabilities. From this, approximately optimal play can be computed.

Additional curricular materials and full details for all dynamic programming and value iteration exercises are available through the MLExAI project Web site.[12]

## 4.6 Clue Deduction: An Introduction to Satisfiability

4.6.1 *Introduction.* There are many ways to teach knowledge representation and reasoning (KR&R). The approach we describe here has a number of distinguishing features. First, it is goal directed, with the goal being to develop expert artificial intelligence for reasoning about the popular board game Clue®(a.k.a., Cluedo®), a mystery-themed game of deduction. By featuring one of the last half-century's most popular board games in the world, we have chosen a goal that is both fun and accessible. Further, we have chosen to deal exclusively with propositional logic only, covering most core KR&R concepts without first-order logic complexities. Drawing from both AI texts and mathematics journals, we offer a high-quality selection of word problems for propositional logic. Finally, the core student project is extensible in several important ways, creating incentives for further exploration in constraint satisfaction and Boolean satisfiability reasoning engines.

4.6.2 *Project Objectives.* The object of this project is to give the student a deep, experiential understanding of propositional knowledge representation and reasoning through explanation, worked examples, and implementation exercises.

—Gain an understanding of the syntax and semantics of propositional logic, as well as general logic terminology, including "model", "(un)satisfiability", "entailment", "equivalence", "soundness", and "completeness".

—Learn the process of knowledge base conversion to Conjunctive Normal Form (CNF).

—Solve word problems with proof by contradiction (a.k.a., reductio ad absurdum) using resolution theorem proving.

—Represent knowledge so as to complete a program implementation that performs expert reasoning for the game of Clue.

[11]For rules, see http://www.hasbro.com/common/instruct/PassThePigs.PDF.
[12]See http://uhaweb.hartford.edu/compsci/ccli/.

—Compare deductive learning, inductive learning, and knowledge acquisition.

4.6.3 *Project Description.* The murder-mystery themed board game of Clue was invented by Anthony E. Pratt and first produced in the U.K. by Waddingtons as “Cluedo” in 1949. Since that time, Clue became one of the most popular family board games of the last half century.

In the backstory of the game, Mr. Boddy has been murdered in one of the rooms of his mansion Tudor Close. The game’s object is to be the first to correctly deduce the correct murder suspect, weapon, and room. Three to six players arbitrarily assume roles of possible suspects, but a player’s guilt or innocence does not affect play.

A deck of 21 cards depicts each of six possible suspects, six possible weapons, and nine possible rooms. One of each of these three card types is randomly chosen and placed, unseen by any player, into an envelope called the case file. These three cards are the unknown murder suspect, weapon, and room for this game. The remaining 18 cards are then shuffled and dealt clockwise to players. (In a four- or five-player game, cards will not be dealt evenly.)

Each turn, a player rolls dice to move about the board, visiting rooms, and making suggestions about the contents of the case file. When a player suggests a suspect, weapon, and room, opponents clockwise indicate that they cannot disprove the suggestion until an opponent has one or more of the suggested cards and must reveal one privately to the suggester. Each player may declare one accusation in the game, checking the case file to see if the contents have been correctly named. If correct, the player wins; if not, the player loses and continues to disprove suggestions when possible. Thus, Clue is primarily a game of information, and successful players are skilled at gaining more information than their opponents, using it to deduce or guess further information.

Students often share fond memories of playing Clue as children. Indeed, Clue is often considered a simple children’s game. This is perhaps due to the fact that the game’s detective notepads are too small to make substantial notes, and instructions for use of detective notepads merely encourage the player to simply mark the cards the player has seen. However, most information available in the game concerns cards not seen. It is important to note what is not held by an opponent, or that one of three suggested cards was shown by one opponent to another. In our experiences, few Clue players make such notes or appreciate the logic puzzles each game presents.

Applying logical reasoning to all basic propositional information in the game allows a player to make surprising expert deductions, reaching a winning accusation in a fraction of the time of a novice. More than a children’s game, Clue is revealed as a game of deeper deduction. Between nostalgia and mastery of a popular game, students enjoy a positive, fun experience as they learn fundamental concepts of logic.

## Propositional Logic

Our curricular materials begin with a concise presentation of the syntax and semantics of propositional logic. Examples are based on facts in the game of Clue and a simple liars/truth-tellers problem. A central running example in the introduction, conjunctive normal form, and resolution theorem proving is as follows.

> *Example Problem*. Suppose that liars always speak what is false, and truth-tellers always speak what is true. Further suppose that Amy, Bob, and Cal are each either a liar or truth-teller. Amy says, "Bob is a liar." Bob says, "Cal is a liar." Cal says, "Amy and Bob are liars." Which, if any, of these people are truth-tellers?

Using Clue facts and this problem, we cover atomic sentences, operators ($\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$), literals, propositional logic's BNF grammar, truth assignments, (un)satisfiability, models, validity, tautologies, entailment, and logical equivalence. Step by step, we show that the knowledge base of the example problem above can be expressed as

$$\{A \Leftrightarrow \neg B, B \Leftrightarrow \neg C, C \Leftrightarrow \neg A \wedge \neg B\},$$

where the atomic sentences have the following interpretation:

—$A$ - Amy is a truth-teller.
—$B$ - Bob is a truth-teller.
—$C$ - Cal is a truth-teller.

## Conjunctive Normal Form and Resolution Theorem Proving

Students are next taught the process of converting propositional logic to conjunctive normal form (CNF). This then sets the stage for teaching resolution theorem proving. We describe reductio ad absurdum, that is, proof by contradiction, and generalized modus ponens. We convert our example knowledge base to CNF and show how to prove that Cal is a liar by assuming he is a truth-teller and deriving a contradiction through resolution:

| | | | |
|---|---|---|---|
| (1) | $\{\neg A, \neg B\}$ | | Knowledge base |
| (2) | $\{B, A\}$ | | |
| (3) | $\{\neg B, \neg C\}$ | | |
| (4) | $\{C, B\}$ | | |
| (5) | $\{\neg C, \neg A\}$ | | |
| (6) | $\{\neg C, \neg B\}$ | | |
| (7) | $\{A, B, C\}$ | | |
| (8) | $\{C\}$ | | Assumed negation |
| (9) | $\{\neg A\}$ | (5),(8) | |
| (10) | $\{B\}$ | (2),(9) | |
| (11) | $\{\neg C\}$ | (3),(10) | |
| (12) | $\{\Box\}$ | (8),(11) | Contradiction! |

## SATSolver, DIMACS CNF Format, and zChaff

Our project is structured in such a way that students may do as little as represent knowledge, and may go so far as to implement the entire reasoning system. We make use of zChaff,[13] a free, efficient, complete, DPLL-style satisfiability

[13]See http://www.princeton.edu/%7Echaff/zchaff.html.

solver [Moskewicz et al. 2001]. Any solver may be used that accepts the standard DIMACS CNF format[14] and reports whether the given knowledge base is satisfiable or not. We thus treat the underlying satisfiability reasoning engine as a black box which students may choose to implement for an even richer educational experience.

Thus, the instructor may choose to leave the solver as an abstract black box, or delve into the chosen solver's algorithm. This project is intended as a flexible, customizable starting point that is minimally a knowledge acquisition and representation exercise, but could also motivate a deeper inquiry into reasoning algorithms.

### Propositional Logic Exercises

At this point, students are prepared to exercise knowledge representation, resolution theorem proving, and automated theorem proving through class `SATSolver`, a thoroughly documented Java interface to zChaff. Drawing from a diverse set of artificial intelligence texts, logic texts, and mathematics journals, we selected and adapted eight excellent propositional reasoning problems. Students are asked to approach each problem in four parts as follows.

(a) Express all relevant problem facts as a propositional logic knowledge base. Clearly explain the meaning of the propositional symbols.
(b) Convert the propositional logic knowledge base to CNF.
(c) Use resolution theorem proving to solve the problem.
(d) Solve the problem computationally with a SAT solver.

A student successfully completing a few of these exercises should have a firm grasp of the main concepts up to this point.

### Clue Reasoner

Given the `SATSolver` and exercises in knowledge representation and conversion to CNF, we are now ready to create an expert Clue reasoner. A skeleton class `ClueReasoner` is supplied which contains all but knowledge needed for the knowledge base, so this is essentially a knowledge acquisition and representation exercise.

Each Clue atomic sentence $c_p$ symbolizes the statement "The card $c$ is in place $p$". There is an atomic sentence for each place and card pair. For DIMACS CNF format, we provide a unique integer, that is, a Gödel number, for each atomic sentence as follows. Suppose we have a place index $i_p$ and a card index $i_c$. Then the integer corresponding to $c_p$ is $i_p \times \texttt{numCards} + i_c + 1$. Since $i_c \in [0, \texttt{numCards} - 1]$ and $i_p \in [0, \texttt{numPlayers}]$ (including the case file), then each atomic sentence has a unique number from 1 through $(\texttt{numPlayers} + 1) \times \texttt{numCards}$.

[14]See the project documentation in Section 5.

The knowledge of Clue can be divided into two types: initial knowledge and play knowledge. Initial knowledge is that which is known right before the first turn and includes the following.

—Each card is in exactly one location.
—Exactly one card of each category is in the case file.
—You know your hand of cards.

Play knowledge is knowledge gained during events of the game as follows.

—A player cannot refute a suggestion.
—A player refutes your suggestion by showing you a card.
—A player refutes another player's suggestion by showing them a card privately.
—A player makes an accusation and shares whether or not it was correct.

Whether initial or play knowledge, it is added immediately clause by clause to the knowledge base. Satisfiability testing is done with the knowledge base and each possible literal in turn in order to determine where cards are, are not, or possibly may be located. The results are printed in a table.

In the first iteration of this project, we were surprised to find that the reasoning performance of the ClueReasoner was beyond that of the expert computer players of Atari's commercial product *Clue: Murder at Boddy Mansion*. In one game, several turns before an expert computer player was able to make a correct accusation, the assigned project system was able to deduce the contents of the case file given only the information known to a non-playing observer who never sees any cards. Students should be encouraged by the fact that they can both comprehend and implement a state-of-the-art reasoning system.

### Project Conclusion and Advanced Projects

At the conclusion of the project, students learn about elimination of equivalent, subsumed, and tautology clauses to improve performance. This is followed by a discussion of deductive learning, inductive learning, and knowledge acquisition in the larger context of machine learning. Finally, three advanced projects are suggested.

(1) All previous reasoning omits important initial knowledge: the number of cards dealt to each player. This is common knowledge, and can be very important in practice. One approach is to represent this knowledge in CNF in a simple combinatorially large form or a compact adder-circuit form. A second approach is to perform such reasoning at a meta-level, repeatedly performing all other deductions until the meta-level can deduce nothing more. A third approach is to operate directly with pseudo-Boolean constraints as a constraint satisfaction problem. Indeed, this would provide a good segue to the topic of constraint satisfaction.
(2) The zChaff SAT solver may be replaced with the student's own simple DPLL-type solver. Whereas the project thus far has been largely concerned

with knowledge representation, an implementation of the underlying reasoning engine would be an excellent educational experience as well.

(3) A student might instead opt to experiment with the implementation of a simple stochastic local search (SLS) [Hoos and Stützle 2005]-based reasoning engine, for example, WalkSAT [Selman et al. 1996]. The practical application of incomplete WalkSAT search to Clue reasoning gave students a feel for the tradeoff of speed versus quality of the reasoning result. The more iterations WalkSAT is given, the more likely the algorithm will find satisfying truth assignments when they exist, answering knowledge base queries correctly.

## 5. INTEGRATING THE PROJECTS

The projects presented here allow for varying levels of programming sophistication, with some including more focus on tools allowing flexibility of their use in language-independent AI courses where students can utilize a language they are familiar with, while others have significant programming components. Open-ended questions included in the projects provide several opportunities for students seeking challenges and ideas for capstone or research projects.

The projects are customizable and easily adaptable. At one extreme, students may implement an entire machine learning system that illustrates a core AI topic. At the other extreme, students apply our solution code to understand the computational characteristics of the algorithms. In between is a range of choices, where instructors individually decide how much implementation is best for their students' learning.

The projects allow for varying entry points. While we envision using such labs for a full term of projects, the different entry points into each project will allow faculty to tailor these projects as they deem necessary for their course. We have used the projects as long-term projects using a team-based approach with teams of 2-3 students. The projects are intended to supplement and not replace regular assignments that reinforce concepts. In addition to the term project, smaller assignments throughout the semester were given to reinforce the concepts covered in class.

We see our material as being relatively self-contained and independent of a particular text. Each project includes several deliverables throughout the semester that map into topics covered in the course. Figure 1 includes a sample syllabus used with the projects. Table I presents a mapping of the Web Document Classification project phases to course topics. The project combines a number of important AI areas in a consistent way. Web search engines use advanced search algorithms and information retrieval techniques to find Web pages. They also use knowledge representation techniques to organize and structure the search results and create ontologies (topic directories). The project consists of three phases with several deliverables throughout the semester: Web document collection, feature extraction and data preparation, and machine learning.

The other topics section of the syllabus allows for varying topics based on the project. The Web page associated with each project on the MLeXAI Web

1. *Introduction, Intelligent Agents* - Chapters 1, 2
2. *Problem Solving by Search* - Chapter 3
3. *Heuristic (Informed) Search* - Chapter 4
4. *Constraint Satisfaction* - Chapter 5
5. *Games* - Chapter 6
6. *Knowledge-Based Agents, Propositional and First-Order Logic* - Chapters 7, 8
7. *Inference in First-Order Logic, Logic Programming and Prolog* - Chapter 9
8. *Knowledge Representation* - Chapter 10
9. *Planning* - Chapter 11
10. *Uncertainty and Probabilistic Reasoning* - Chapters 13, 14
11. *Machine Learning: Basic Concepts, Version Space, Decision Trees* - Chapters 18, 19
12. *Machine Learning: Numeric Approaches, Clustering, Evaluation* - Chapter 20
13. *Learning with Background Knowledge - Explanation-Based Learning, Inductive* - Chapter 19
14. *Natural Language Processing* - Chapters 22, 23
15. *Other Topics and Philosophical Foundations* - Chapter 26

Fig. 1. AI syllabus based on Russell and Norvig [2003].

Table I. Mapping of Project Topics

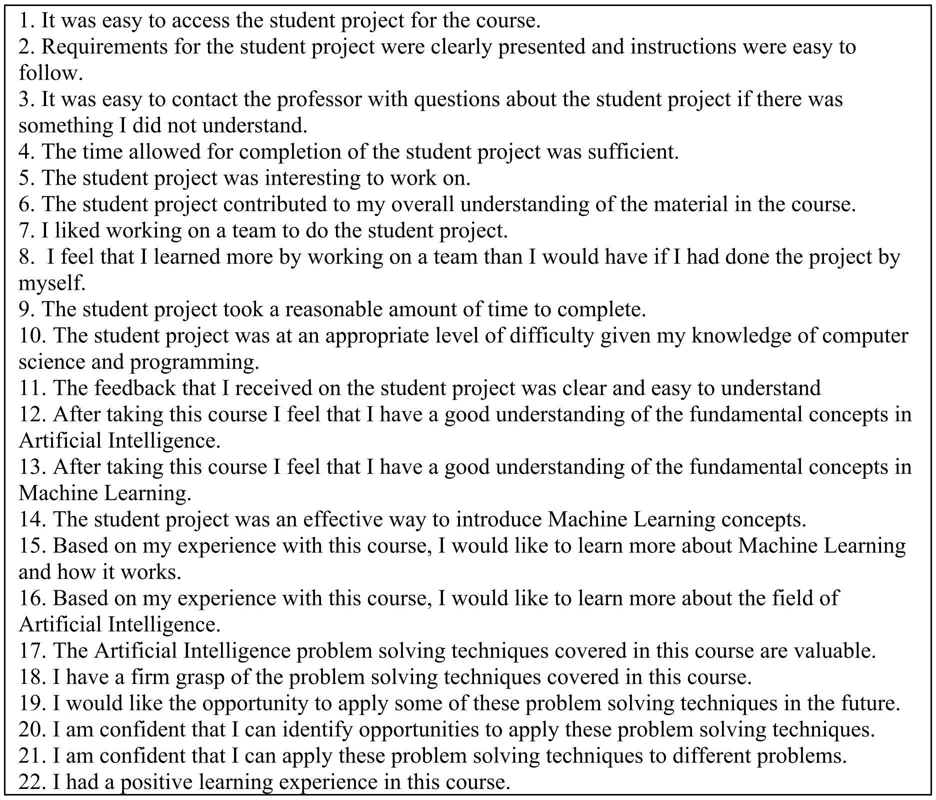
| **Deliverable from the Web Document Classification Project** | | **AI Topic** |
|---|---|---|
| Phase 1: Collecting sets of Web documents grouped by topic | | 2, 3 |
| Phase 2: | Step 1: Keyword Selection | 14 |
| | Step 2: Feature Extraction | 6, 10, 14 |
| | Step 3: Data Preparation | 10, 11, 14 |
| Phase 3: Machine Learning | | 10, 11, 12 |

site includes a sample syllabus that can be used with that specific project. Any additional readings not in the book are included in the project description. In the case of the Web Document Classification, additional readings are available at the Web site of the machine learning system used in the project (Weka) [Weka Home Page].

## 6. ASSESSMENT

The effectiveness of these projects was evaluated with the assistance of internal and external evaluators through a multi-tier evaluation system involving faculty, students, and an advisory board. Led by the project evaluator, the evaluation process involved both a formative evaluation, which guided the development efforts, and a summative evaluation.

This introductory course in Artificial Intelligence was offered six times in three different locations by a total of four different faculty members over a three-year period. The locations included Gettysburg College (G), a small liberal arts college; Central Connecticut State University (C), a large state university; and The University of Hartford (H), a private comprehensive university. Students were asked to evaluate the student projects, which were an integral part of the course, by completing a student survey comprised of 22 ranked questions and two open-ended questions. In addition, a sample of students was interviewed from each location using either a direct (group) interview or a group conference call. Class size varied from a minimum of 12 to a maximum of 25. The percentage of students who completed the questionnaires varied from 50% to 85% of class participants. Figure 2 includes the set of questions and



1. It was easy to access the student project for the course.
2. Requirements for the student project were clearly presented and instructions were easy to follow.
3. It was easy to contact the professor with questions about the student project if there was something I did not understand.
4. The time allowed for completion of the student project was sufficient.
5. The student project was interesting to work on.
6. The student project contributed to my overall understanding of the material in the course.
7. I liked working on a team to do the student project.
8. I feel that I learned more by working on a team than I would have if I had done the project by myself.
9. The student project took a reasonable amount of time to complete.
10. The student project was at an appropriate level of difficulty given my knowledge of computer science and programming.
11. The feedback that I received on the student project was clear and easy to understand
12. After taking this course I feel that I have a good understanding of the fundamental concepts in Artificial Intelligence.
13. After taking this course I feel that I have a good understanding of the fundamental concepts in Machine Learning.
14. The student project was an effective way to introduce Machine Learning concepts.
15. Based on my experience with this course, I would like to learn more about Machine Learning and how it works.
16. Based on my experience with this course, I would like to learn more about the field of Artificial Intelligence.
17. The Artificial Intelligence problem solving techniques covered in this course are valuable.
18. I have a firm grasp of the problem solving techniques covered in this course.
19. I would like the opportunity to apply some of these problem solving techniques in the future.
20. I am confident that I can identify opportunities to apply these problem solving techniques.
21. I am confident that I can apply these problem solving techniques to different problems.
22. I had a positive learning experience in this course.

Fig. 2. Student questionnaire.

Figure 3 presents the results. The number following the letter in the site (university) name indicates whether it was the first or second course offering. In the first offering, students at Colleges C and H were allowed to select one project from the Web document classification, Web user profiling, the eight-puzzle problem, and character recognition. The projects were relatively evenly selected. Students in College G were assigned the dice game Pig and the game of Clue, one during the first half of the semester and the other during the second half. Projects were adapted to be suitable for the period of a few weeks each. The "a" and "b" designations correspond to the two projects. In the second offerings, students in Colleges C and H were asked to select one from the six projects. Students in College G were again assigned the dice game Pig and the game of Clue. Several students at College G pursued independent studies, selecting one of the other four projects. Instructors from the three institutions worked closely and were available to answer student questions from all three institutions. Based on our experiences, we recommend that instructors make available only two to three projects for the students to select from in a given offering.

On the written student evaluation, items were scored from 1 to 5 with 5 being the highest level. The evaluation results, by location, are summarized in

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4) Time allowed for completion of the student project was sufficient | | | | | | | | |
| Site | **H1** | **H2** | **G1a** | **G2a** | **C1** | **C2** | **G2a** | **G2b** |
| Average | 4.15 | 4.86 | 4.5 | 4.25 | 4.27 | 3.75 | 4.5 | 4.57 |
| %A&SA | 77% | 100% | 100% | 88% | 93% | 75% | 100% | 88% |
| 5) Student project was interesting to work on | | | | | | | | |
| Site | **H1** | **H2** | **G1a** | **G2a** | **C1** | **C2** | **G2a** | **G2b** |
| Average | 4.15 | 4.57 | 4.44 | 4.13 | 4.4 | 4.0 | 4.17 | 4.57 |
| %A&SA | 92% | 86% | 83% | 75% | 80% | 75% | 83% | 100% |
| 6) Project contributed to my overall understanding of the material in the course | | | | | | | | |
| Site | **H1** | **H2** | **G1a** | **G1b** | **C1** | **C2** | **G2a** | **G2b** |
| Average | 4.31 | 4.57 | 4.67 | 4.13 | 4.47 | 4.5 | 4.47 | 4.71 |
| %A&SA | 92% | 100% | 100% | 88% | 93% | 100% | 93% | 100% |
| 9) Student project took a reasonable amount of time to complete | | | | | | | | |
| Site | **H1** | **H2** | **G1a** | **G1b** | **C1** | **C2** | **G2a** | **G2b** |
| Average | 4.15 | 4.0 | 4.17 | 3.38 | 4.27 | 3.75 | 4.0 | 4.57 |
| %A&SA | 77% | 86% | 67% | 100% | 93% | 50% | 83% | 100% |
| 12) After taking this course I feel that I have a good understanding of the fundamental concepts in AI | | | | | | | | |
| Site | **H1** | **H2** | **G1a** | **G1b** | **C1** | **C2** | **G2a** | **G2b** |
| Average | 4.38 | 4.29 | 4.17 | 4.5 | 4.27 | 4.5 | 4.17 | 4.57 |
| %A&SA | 92% | 100% | 83% | 100% | 93% | 100% | 83% | 100% |
| 16) Based on my experience with this course, I would like to learn more about AI | | | | | | | | |
| Site | **H1** | **H2** | **G1a** | **G1b** | **C1** | **C2** | **G2a** | **G2b** |
| Average | 4.31 | 4.43 | 4.17 | 4.4 | 4.33 | 4.5 | 4.17 | 4.14 |
| %A&SA | 85% | 71% | 83% | 88% | 80% | 100% | 83% | 88% |
| 22) I had a positive learning experience in this course | | | | | | | | |
| Site | **H1** | **H2** | **G1a** | **G2a** | **C1** | **C2** | **G2a** | **G2b** |
| Average | 4.54 | 4.29 | 4.67 | 4.63 | 4.43 | 4.25 | 4.67 | 4.71 |
| %A&SA | 92% | 100% | 100% | 100% | 87% | 75% | **100%** | 100% |

Fig. 3. Results of student questionnaire.

Figure 3. The %A&%SA represents the percentage of students who responded with Agree (score of 4) or Strongly Agree (score of 5). Results reveal that, overall, students felt that the project was easy to access and that the requirements were clearly presented (Questions 1-2). They uniformly felt that the professors in all three locations were easy to contact (Question 3).

For the most part, students felt that the time allotted for the project was sufficient and that the feedback they received was clear and easy to understand (Questions 9 and 11). They also felt that the student project was at an appropriate level of difficulty given their abilities (Question 10).

Students expressed their enthusiasm in a variety of ways. They felt that the project was interesting to work on and contributed to their overall understanding of the material in the course (Questions 5-6). By the end of the semester, most felt that they had a good understanding of the fundamentals of AI and machine learning (Questions 12-13). The majority of the students surveyed indicated that they would like to learn more about both AI and Machine Learning (Questions 15-16). They also indicated that they would like further opportunities to apply the problem-solving techniques in the future (Question 19).

The majority of students was enthusiastic about the projects, and felt that they were an effective way to introduce Machine Learning (Question 14). Their responses indicated a high level of confidence in their ability to apply these problem-solving techniques in other situations (Questions 18, 20, 21). In summarizing their experience, students stated that they had a very positive learning experience in the course (Question 22). These sentiments were reinforced by their responses to Question 23, an open-ended question that asked what they liked best about the course. One student commented "It was nice to see the technique work on something that is used in our lives." A second student noted "The student project allowed us to use what we know to try to solve an interesting question."

Students were more ambivalent about their experience with the team aspect of the student projects. Some students liked the teams very much and felt that team participation enhanced their knowledge of the course (Questions 7-8). A minority of students, however, gave very low ratings on the questions pertaining to the team experience. They elaborated further on Question 24, an open-ended question regarding what they liked least about the course. One student noted "I did not like the fact that it was a team project one bit, but that's because I'm completely against the enforced team approach to education in general." Another (amusingly enough) commented "I hate working in groups. I spent half of the time explaining to my partner why he was wrong. . . ". The project evaluator noted to participating faculty during the most recent project board meeting that the students who liked teamwork the least are probably the very ones who need to practice it the most.

The group interviews and conference calls provided further elaboration for responses provided in the written evaluation. Overall, students indicated that they had a minimal amount of prior knowledge or experience with either AI or Machine Learning (Questions 1-2). Most students indicated that they found the team projects to be a very valuable learning experience (Question 3). Those that did not were typically individuals who either did not like teamwork in general or who happened to have a problem with a teammate.

The different faculty who taught the course addressed potential teamwork problems in a variety of ways. For example, one faculty member had the teams select a mandatory weekly meeting time that all team members agreed to in advance. The same faculty member established a process whereby a team could "kick out" a nonfunctioning team member. Three of the four participating faculty devoted extra time to talking with the class about the establishment of teams, the role of teams, and the management of teams. They also provided counseling to teams and individual team members throughout the semester. In general, the participating faculty members felt that the team experience was an important component of the course. Typically, students will work as a part of a team in the workplace. Thus, they need to develop skills in teamwork, communication, and conflict management.

Students were very enthusiastic about the organization of the course, the number of topics covered, and the hands-on nature of the team project (Question 4). They liked being able to use AI principles to solve real problems. In fact, the majority stated that this was the most positive aspect of the course.

Further, most students indicated that the projects enhanced their understanding and gave them new ways to think about the problems.

In terms of course weaknesses, students in earlier sections of the course indicated that the workload was overly burdensome. Faculty addressed this problem in subsequent iterations of the course by screening the list of topics and adjusting the number of projects required. Students also raised issues about the documentation for code, their familiarity (or lack thereof) with the programming language used, and the timing of feedback on assignments. Again, most of these issues were addressed in later iterations of the course.

As with the written evaluation, students indicated a high level of satisfaction with the course and with the method of instruction. By the end of the semester, they had a much better understanding of the importance of AI and its broad applicability. Many also indicated the desire to pursue additional study or research in AI and machine learning.

A sample of student comments from the evaluation forms follows.

> *I liked acquiring knowledge about machine learning techniques and being able to implement a system and see it work. This gave me a concrete understanding of the concepts.*
>
> *I liked the fact that the project I worked on pertained to the Internet and web document classification. It presented a useful real world application of machine learning. Often, examples in the book or other projects lack real world usefulness.*
>
> *Working on the project was a great experience. I was able to see how various AI concepts tie together in developing a machine learning system. I was amazed by the wide range of applications of machine learning in various aspects of our lives.*
>
> *The project was very interesting and challenging. Working in groups was a good experience.*
>
> *The project was really neat. I was challenged to strengthen my deductive reasoning skills by "formalizing the process by which I derive solutions." The problems associated with "satisfiability" are fun to work out, but they also provide me with an intellectual challenge.*

## 7. CONCLUSION

We presented experiences over a three-year period using a unifying theme of machine learning as an effective way to tie together the various AI concepts while at the same time emphasizing AI's strong tie to computer science. We presented a suite of adaptable, hands-on laboratory projects that can be easily integrated into the introductory AI course and thus provide a framework for enhancing the student experience with learning the AI topics. Each of our projects involves the development of a machine learning system for a specific application. While we are using machine learning as a theme in our project, we believe we are creating a framework and a successful model that can be adapted to other themes and other elective courses in the curriculum.

We have built a community of over 40 faculty members interested in using our material, some of whom have already started using it. Others plan to do so during the upcoming academic year. Those who have used the material have written very positively about their experiences. Furthermore, some of the instructors who used our material already have contributed their adaptation back to our curricular material. Phase 2 of MLeXAI, funded by another grant from NSF, is currently ongoing and includes the development of additional projects. Indeed, 20 additional projects are being created and used by faculty around the country. Phase 2 of the MLeXAI project will further investigate the impact of theme and problem-based learning on student outcomes and examine whether the results of Phase 1 can be extended and generalized by 20 faculty across the country.

## REFERENCES

BOYAN, J. A. 1998. Learning evaluation functions for global optimization. Tech. rep. CMU-CS-98-152, Carnegie Mellon University.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN C. 2001. *Introduction to Algorithms* 2nd Ed. The MIT Press, Cambridge, MA.

DODDS, Z. ET AL., (EDS.) 2006. Robots and robotics in undergraduate AI education. *AI Mag. 27*, 1. AAAI Press.

FOX, S. 2007. Introductory AI for both computer science and neuroscience students. In *Proceedings of the 20th International FLAIRS Conference (FLAIRS'07)*.

GREENWALD, L., (ED.) 2004. Accessible hands-on artificial intelligence and robotics education. Tech. rep. AAAI Press.

HARLAN, R., LEVINE, D., AND MCCLARIGAN, S. 2001. The Khepera robot and the kRobot class. In *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education (SIGCSE'01)*. 105–109.

HEARST, M., ED. 1995. Improving instruction of introductory AI. Tech. rep. FS-94-05, AAAI Press.

HOOS, H. H. AND STÜTZLE, T. 2005. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann.

KLASSNER, F. 2006. Launching into AI's *October Sky* with robotics and lisp. *AI Mag. 27*, 1. AAAI Press.

KNIZIA, R. 1999. *Dice Games Properly Explained*. Elliot Right-Way Books, Lower Kingswood, UK. 129.

KUMAR, D. AND MEEDEN, L. 1998. A robot laboratory for teaching artificial intelligence. In *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'98)*. 341–344.

KUMAR, A., ET AL. 2006. Non-traditional projects in the undergraduate AI course. In *Proceedings of the 37th Annual SIGCSE Technical Symposium on Computer Science Education (SIGCSE'06)*.

MARKOV, Z. AND LAROSE, D. 2007. *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*. Wiley, New York. (Chapter 1 is available for free download from http://media.wiley.com/product_data/excerpt/56/04716665/0471666556.pdf).

MARKOV, Z., RUSSELL, I., NELLER, T., AND ZLATAREVA, N. 2006b. Pedagogical possibilities for the *N*-puzzle problem. In *Proceedings of the 36th Frontiers in Education Conference (FEC'06)*. IEEE Press.

MITCHELL, T. M. 1997. *Machine Learning*. McGraw-Hill, New York.

MITCHELL, T. M. 2006. The discipline of machine learning. Tech. rep. CMU-ML-06-108, Carnegie Mellon University.

MOSKEWICZ, M., MADIGAN, C., ZHAO, Y., ZHANG, L., AND MALIK, S. 2001. Chaff: Engineering an efficient sat solver. In *Proceedings of the 39th Design Automation Conference (DAC'01)*.

NELLER, T., PRESSER C., RUSSELL, I., AND MARKOV, Z. 2006a. Pedagogical possibilities for the dice game Pig. *J. Comput. Sci. Col. 21*, 6, 149–161.

NILSSON, N. 1998. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers.

RUSSELL, I. AND NELLER, T. 2003. Implementing the intelligent systems knowledge units of computing curricula 2001. In *Proceedings of the Frontiers in Education Conference (FIE'03)*. IEEE Press.

RUSSELL, S. AND NORVIG, P. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, NJ.

RUSSELL, I., MARKOV, Z., NELLER, T., AND COLEMAN, S. 2005a. Enhancing undergraduate AI courses through machine learning projects. In *Proceedings of the 35th Frontiers in Education Conference (FIE'05)*. IEEE Press.

RUSSELL, I., MARKOV, Z., NELLER, T., GEORGIOPOULOS, M., AND COLEMAN, S. 2005b. Unifying undergraduate AI courses through machine learning projects. In *Proceedings of the 25th American Society for Engineering Education Conference (ASEE'05)*.

SELMAN, B., KAUTZ, H., AND COHEN, B. 1996. Local search strategies for satisfiability testing. In *Proceedings of the DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26 (DIMACS'96)*, 521–532.

SUTTON, R. S. AND BARTO, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

WEKA HOME PAGE. http://www.cs.waikato.ac.nz/∼ml/weka.

WILENSKY, U. 1991. Abstract meditations on the concrete and concrete implications for mathematics education. In I. Harel and S. Papert, Eds. *Constructionism*. Ablex, Norwood, NJ. 193–203.

WYATT, R. 2000. Curriculum descant. *ACM Intell. Mag. 11*, 2. ACM Press.

Received August 2009; revised March 2010; accepted May 2010