Throw Down an AI Challenge

Todd W. Neller
Gettysburg College
Department of Computer Science
Gettysburg, Pennsylvania 17325
tneller@gettysburg.edu

Ingrid Russell
University of Hartford
Department of Computer Science
West Hartford, Connecticut 06117
irussell@hartford.edu

Zdravko Markov
Central Connecticut State University
Department of Computer Science
New Britain, Connecticut 06050
markovz@ccsu.edu

Abstract

Students will be more likely motivated to pursue a field of study if they encounter relevant, exciting, and interesting challenges early in their studies. The NSF CCLI project "Machine Learning Laboratory Experiences for Introducing Undergraduates to Artificial Intelligence" uses machine learning as a unifying theme to teach diverse artificial intelligence (AI) topics in the context of an introductory AI course. In this paper, we extend our work to include interesting AI challenges suitable for the context of introductory CS1 and CS2 courses.

Introduction

In Dale Carnegie's motivational classic *How to Win Friends & Influence People* [1], the reader is urged to "throw down a challenge". Noting the work of behavioral scientist Frederic Herzberg, he observes "The one major factor that motivated people was the work itself. If the work was exciting and interesting, the worker looked forward to doing it and was motivated to do a good job." Applying such ideas to student enrollment, we propose that a student will be more likely motivated to pursue a field of study if they encounter relevant, exciting, and interesting challenges early in their studies.

As instructors seek the ideally structured syllabus to communicate foundational computer science concepts, it is vital that student experiences include engaging challenges that provide a sense of the problem-solving possibilities of our discipline. A student may miss these possibilities amidst unmotivated programming drills. As teachers, we must ask ourselves, "Is this how I became a computer scientist?" Very likely, we trace the beginning of our enjoyment of the discipline not to a concept or programming language, but to intriguing challenges and the satisfaction of meeting those challenges.

AI instructors are immersed in some of the most exciting and interesting challenges of the field. Robotics and games are just two areas with demonstrable attraction for high-school students and non-majors. Lego Mindstorms, Alice, and Scratch[1] provide good access to robotics, interactive 3D graphics, and interactive web applications, respectively. One ongoing challenge is how to take representative challenges from AI areas beyond robotics and make them accessible at early stages of education.

In this paper, we describe a machine-learning themed approach to teaching AI, related projects that have been developed and deployed, and suggestions for introductory computer science exercises related to each project.

Machine Learning Theme

It is believed by many faculty members 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 [7]. 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 [2,5]. Recently, work has been done to address the diversity of topics covered to create a theme-based approach. Russell and Norvig present an agent-based approach [23]. A number of instructors have been working to integrate Robotics into the AI course [2,3,5,6,8,9].

Our approach, Machine Learning Laboratory Experiences for Introducing Undergraduates to Artificial Intelligence (MLExAI), 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. MLExAI has several objectives:

[1] http://mindstorms.lego.com/, http://www.alice.org/, and http://scratch.mit.edu/, respectively.

- Enhance the student learning experience in the AI course.
- Increase student interest and motivation to learn AI.
- Introduce students to an increasingly important research area, thus motivating them to pursue further study.
- 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 has been 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. 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 [25]. 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.

In the next section, we describe each of these six projects in turn, and present additional relevant project suggestions that are suitable for engaging students at an earlier stage of study.

Projects

We have developed six hands-on laboratory projects that can be closely integrated into a one-semester AI course. Details on the various projects have been published in [11,12,14,15,19,20]. Sample solutions along with support code are also available to all instructors using the material. Additional information and complete project descriptions are available at the project web page at: http://uhaweb.hartford.edu/compsci/ccli/index.htm. We also here present relevant challenges suitable for introductory courses in computer science.

Web User Profiling and Web Document Classification

It is generally acknowledged that the web is the largest collection of electronically accessible documents, which make the richest source of information in the world. One problem with the web is that this information is not well structured and organized so that it could be easily retrieved. Search 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 and try to extract knowledge from raw data available on the web by organizing web pages in well defined structures or by looking into web user activity patterns.

These are the challenges that the web user profiling and the web document classification projects address. The aim of the first project is to allow students to build and experiment with the basic components of an intelligent web browser that can automatically adjust to the preferences of its user by creating a machine learning model of these preferences using data collected from web searches and user feedback. The model is then used for improving the efficiency of new web searches performed by the same or new users. The web document classification project investigates the process of classification of web pages according to their topic. Known topic directory structures are used as training data for machine learning algorithms which then create models of document topics. These models can be further applied to new web documents to determine their topic. In this way, firstly the accuracy of the initial topic structure can be evaluated and secondly, new web documents can be classified into existing topics.

The web user profiling and the web document classification projects allow students to explore and learn a number of important AI related areas in a consistent way. Among them are advanced search algorithms, information retrieval, knowledge representation, and machine learning. All these areas require students to apply basic Computer Science methods and techniques such as sorting, searching, indexing, and web programming, which are typically covered in the introductory CS1 and CS2 courses. This in turn allows these projects to be used as a framework for teaching core CS topics in a more attractive and motivating way.

Introductory Challenges. Both projects combine a number of important areas related to the introductory CS1 and CS2 courses. Web search engines use various search algorithms to explore the web graph and topic hierarchies organize the web documents in tree structures. At the document collection stage, students have the opportunity to explore these areas. The project also involves manipulation of various data structures and at some points (especially at data collection and feature extraction) students are encouraged to write code for this purpose. The Weka open

source provides an excellent opportunity for this. The system comes with complete Java class documentation of all algorithms and allows using parts of its code separately, modifying or combining code to build stand-alone applications.

The projects have three major stages - data collection, feature extraction and machine learning. In the first two stages, students explore and learn basic information retrieval techniques. In the original projects we suggest that for the implementations and experiments students use the Weka system [24], one of the most popular machine learning systems for educational purposes. By using the Weka data preprocessing tools, (StringToWordVector filter) students create the inverted index (term-document matrix) where each document is a vector of Boolean or numeric values representing the word (term) occurrences in it. For this purpose, however, students can also implement the algorithms on their own which can be a motivating exercise well suited to the CS1 and CS2 courses. Below we describe briefly the basic components of these algorithms, each one of which can be also used as a separate programming project in CS1 or CS2.

1. *Creating a text corpus from a set of text or web documents*. For this purpose students have to implement an algorithm for tokenizing text files into words (terms) and filtering them by using a separate set of stopwords (short words like, a, an, the, of, on etc., which are excluded from keyword search because they usually occur in all documents). An advanced extension to this project can be adding a stemming algorithm that reduces each word to its base form (stem). Stemming is common in information retrieval and text processing and many stemming algorithms are available (see for example, the IR resources page at http://ir.exp.sis.pitt.edu/resources/).

2. *Creating an index table (inverted index)* that allows documents to be retrieved by the words (terms) they contain. The table also should include the frequency of each term (the number of occurrences of the term in each document). This step requires implementing sorting and search algorithms (e.g. binary search), which are typically covered in CS2.

3. *Creating a term-document table (vector space model)*, where each row corresponds to a document and each column – to a term. Thus if there are *m* terms, each document is represented by an *m*-component vector. Depending on the type of the vector coordinates there are three types of vector space models – Boolean, TF (Term Frequency), and TFIDF (Term Frequency - Inverted Document Frequency). Assume that term t_i occurs in document d_j n_{ij} times. In the Boolean model, the coordinates of vector d_j are 0 (if $n_{ij} = 0$) or 1 (if $n_{ij} > 0$). In the TF model, the i^{th} coordinate of document d_j is TF(i,j) = $\log(1+n_{ij}/\sum_k n_{kj})$, where the sum is for all terms in document d_j. In addition a coefficient IDF(i) is computed for each term t_i as IDF(i)=$\log((1+|D|)/|D_i|)$, where D_i is the set of documents where term t_i occurs and D is the whole document collection. Thus, in the TFIDF framework, the i^{th} coordinate of document d_j is the product TF(i,j)*IDF(i). The IDF factor scales the coordinate values down for terms that occur in many documents and up for rare terms occurring in only few documents. In this way the TFIDF transformation adjusts the document vector, stretching it or shrinking it along some dimensions, according to the term importance or relevance for the specific document also taking into account the term distribution in the whole set of documents.

4. *Keyword search and document ranking*. For this step, students implement a keyword search engine. The query is considered as a vector represented in the same format as the document vectors (Boolean or TFIDF). Then the documents are sorted by their closeness to the query vector and a number of documents from the top of the list are returned to the user. To implement such a system, students have to use some basic methods from vector algebra such as calculating norm, Euclidean distance, and dot product. Firstly all vectors (including the query vector) are normalized. Then the document vectors are sorted by minimum Euclidean distance or maximum cosine similarity to the query vector. The latter is mostly used in information retrieval. The cosine similarity is the cosine of the angle between the query vector q and the document vectors d_j. When the vectors are normalized this measure is equivalent to the dot product $q.d_j = \sum_i q^i d^i_j$ Students can experiment with different representations (Boolean, TF or TFIDF) and different measures for closeness to the query vector and then compare the results. This can provide them with insights about how web search engines work and motivate them to further study information retrieval and web search. By working on this project, students practice their CS1 knowledge for implementing sorting and learn important practical applications of computational mathematics.

The algorithms described above are used in all information retrieval systems and web search engines and also form the preprocessing phase in text and web document mining. Student can read Chapter 1 of [10], which discusses these methods and algorithms in detail and provides exercises based on the Weka system [24]. By working on these projects students will learn important CS methods and algorithms and at the same time will explore modern AI related applications which can motivate their greater participation in Computer Science.

Character Recognition Using Neural Networks

The goal of this project is to develop a character recognition system based on a neural network model. The project introduces students to the 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. This allows students to understand the role that neural networks play in the more general context of AI techniques and tools.

In particular, students will (1) learn the basics of single-layer neural networks and perceptron type models and their use as pattern associators, (2) understand the limitations of single-layer networks and the basic types of multilayer networks as well as the backpropagation training algorithm, (3) learn the mathematical foundations of the neural network computation, (4) have a better understanding of the nature of problems that neural networks can solve, (5) gain experience in implementing neural network algorithms for solving basic pattern recognition problems, and (6) have a better understanding of the differences between the neural network approach to solving AI problems and those based on classical symbolic knowledge representation, search and learning. The last part of the project provides opportunities for further research for students seeking added challenges.

Introductory Challenge. A pattern associator is one of the simpler neural network models. Students in a CS2 course can write a program to simulate and test a pattern associator. Such a project introduces students to the area of neural networks and provides some insights to the capabilities of such systems at an early stage in their computer science studies, while at the same time reinforcing introductory computer science concepts. Such a project would require concepts typically covered in CS1 and hence could be assigned towards the end of CS1 or at the beginning of CS2. We have written a neural networks tutorial that is also available on line that can be assigned for background material [16].

The architecture of a pattern associator consists of two layers of units, the input layer and the output layer. The activation a_j of an output unit u_j is given by: $a_j = \sum(w_{ij}*a_i)$, where a_i is the activation of u_i and w_{ij} is the weight of the connection from u_i to u_j. A pattern associator is capable of responding with a certain output given a set of input patterns. More importantly, a pattern associator is able to self-modify and learn; i.e., it is capable of modifying its weights in order to learn a certain input/output association. A learning rule determines how such changes in the weights are to be accomplished. A commonly used learning rule with the pattern associator is the Hebb rule. According to the Hebb rule, the change in weight w_{ij} from u_i to u_j is calculated as $\Delta w_{ij} = r*a_i*a_j$, where r is the learning rule, and a_i, a_j are the activations of units u_i and u_j, respectively. Students can implement a basic pattern associator using the Hebb rule and study the performance of the network. Specifically, an instructor can ask students to train the network with various input/output associations provided to the students and to explore and address the following:

1. The effects of factors such as learning rate, straight versus permuted training (randomly selecting the order), and the number of training iterations.
2. What combination of the above factors provides reasonable performance? Use this combination for training the network.
3. Use the best combination arrived at in (2) to test the network's performance using the training patterns.
4. Test the network's ability to generalize what it learned about the training patterns to other similar patterns and explain the results. One way would be to ask students to test the performance of the network when a linear combination of the training patterns is presented.

While reinforcing concepts covered in CS1/CS2, these exercises are aimed at providing a useful exposure to the field of neural networks and artificial intelligence by introducing a simply yet powerful neural network model, the pattern associator. They point to directions of research in an important area of study and highlight some of the problem-solving capabilities of these systems. Details on this and other similar introductory challenges can be found in [17,18]. Project MLExAI material provides additional challenges for students interested in further investigations in this area.

Solving the N-Puzzle Problem

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. The objective of this project is to introduce the student to Analytical (Explanation-Based) Learning (EBL) 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. Students seeking additional challenging are provided with an opportunity to explore additional topics through independent studies or research projects. [11]

Introductory Challenge. A classic problem, the N-puzzle problem, serves as a good application for illustrating conceptual AI search in a CS 2 course while teaching several CS2 topics [21,22]. In the 8-puzzle version, a 3×3 board consists of 8 tiles numbered 1 through 8 and an empty tile (marked as 0). 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 N-puzzle project can be divided into several parts tackled by students in the following sequence:

- Implementation of a function which given a state (current configuration) generates all new states reachable from that state, and incorporation of that function into a conceptual search program.
- Empirical study of uninformed search algorithms (breadth-first, depth-first, depth-limited, iterative deepening).
- Design and implementation of suitable heuristic functions for the N-puzzle problem (number of tiles in proper places and Manhattan distance among them).
- Empirical study of performance of informed (best-first) search with different heuristic functions.
- Comparison of performance of informed and uninformed search.

Students can do various kinds of experimentation. They may be asked to compare the performance of search algorithms based on size of the search space, length of the solution an algorithm returns, number of times an algorithm backtracks, number of states examined, and elapsed time. These exercises provide for an excellent complexity analysis and illustrate the tradeoff between time efficiency, space efficiency, and quality of the solution found. For example, students discover how the depth-first search memory requirement is less than for breadth-first search (its space complexity is polynomial versus exponential space complexity for the breadth-first search). On the other hand, depth-first search can lead to a dead-end or continue infinitely. In addition, it is not guaranteed to find a solution even if one exists, and if it finds a solution, the solution is not guaranteed to be optimal. However, the polynomial space complexity of the depth-first search makes it a practical choice in larger applications. Breadth-first search, on the other hand, has a large memory requirement since it requires the entire search space to be stored in memory. However, it is guaranteed to find a solution that is nearest to the initial state (the shortest path), if one exists. Following these, related material from Project MLExAI can be used to demonstrate to students how learning improves the performance of search algorithms.

Solving the Dice Game Pig

The jeopardy dice game Pig is very simple to describe, yet the optimal policy for play is far from trivial. The object of Pig is to be the first player to reach 100 points. Each turn, a player repeatedly rolls a die until either a 1 is rolled or the player holds and scores the sum of the rolls. Using the computation of Pig's optimal roll/hold policy as a central challenge problem, we give the student a deep, experiential understanding of dynamic programming and value iteration through explanation, implementation examples, and implementation exercises.

Students first gain insight to optimal Pig play by a documented Java dynamic programming solution to a simple Pig variant with an acyclic state space. This is followed by related dynamic programming exercises. Students are then introduced to the reinforcement learning technique of value iteration, which is demonstrated on a simple coin variant of Pig called Piglet. They are then ready to solve Pig and related games such as Pass the Pigs. Advanced undergraduate research projects are suggested as well.

Introductory Challenge: Pig Implementation. As the simplest representative jeopardy dice game, Pig is frequently used by mathematicians to teach introductory concepts of probability [14]. In the same way, stepwise implementation of Pig can be used to teach a variety of CS1 topics while illustrating bottom-up design with control structures [14]. Consider one possible progression:

- Simulate a single Pig turn where the player holds after rolling 20 or more points.
- Given the current score, simulate a turn where the player additionally holds when the turn total is enough for victory.
- Simulate an entire solitaire game with the same policy.
- Extend this to a two-player game simulation.
- Replace one computer player with a human player, so that the human player competes against this policy.

Along the way, one can ask intriguing questions such as "What is the distribution of outcomes for a 'hold at 20' turn?" or "What is the first player advantage for a 'hold at 20 or goal' game?"

Introductory Challenge: Pig Competition. There is no reason that students must use a "hold at 20 or goal" automated policy in the previous exercises. Indeed, a more intriguing challenge would be to have students create competing policies. Once students understand how to implement an interface, a simple student Pig tournament can spark significant creativity and encourage fun engagement. Many instructors have observed that simple game competitions have been amongst the most engaging, memorable challenges for students.

The instructor should be aware that the only relevant variables for roll/hold decisions are the player scores and the current turn total. However, one could have students design an interface that offers irrelevant information. For example, students making use of the number of rolls so far in a turn fall prey to the Gambler's Fallacy. Such policy

experimentation is common in elementary education exercises, but computer science students can more easily test ideas through substantial simulation.

The relevant resources of our MLExAI project provide a point of continuation for students interested in computing optimal play.

The Game of Clue

The popular boardgame Clue (a.k.a. Cluedo) is the fun focus problem for this introduction to propositional knowledge representation and reasoning. Students first learn the syntax and semantics of propositional logic, general logic terminology (e.g. "model", "(un)satisfiability", and "completeness"), conversion to conjunctive normal form, and resolution theorem proving. This is followed by an array of logical word problems to be solved with and without the aid of a satisfiability solver (e.g. zChaff).

Students are then introduced to the game of Clue and the relevant game knowledge for propositional reasoning. This culminates in the development of software that performs perfect propositional reasoning about Clue game knowledge. Several possible advanced projects are sketched for students that wish to pursue the topic in more depth. Finally, these concepts are tied into the machine learning theme through a comparison of deductive learning, inductive learning, and knowledge acquisition.

Introductory Challenge. A minimalist approach to introducing knowledge representation and reasoning will generally minimize either representation or reasoning topics. Although the Clue project covers resolution theorem proving, exercises are largely concerned with knowledge representation. Students surveyed about their experience with the project have expressed desire to create their own solvers without the aid of a black-box satisfiability solver.

Fortunately, it is not difficult to code simple propositional reasoning engines. For example, a WalkSAT variant and a set-of-support resolution theorem prover can each be implemented in less than 100 lines of Java code. Given word problems with their conjunctive normal form (CNF) representations, a simple reasoning engine is a feasible project for an introductory course.

One simple input format is a text file with one clause per line, with each clause consisting one or more non-zero integers. Each positive integer corresponds to an atomic sentence (i.e. variable, proposition). A negative integer denotes the corresponding negated atomic sentence. For example consider the following problem: Ann says Bob always lies. Bob says Cal always lies. Cal says Ann and Bob always lie. Given that one always tells the truth or always lies, who is lying? If atomic sentences claiming the truthfulness of Ann, Bob, and Cal are numbered 1, 2, and 3 respectively, one CNF input file encoding would be:

```
-1 -2
1 2
-2 -3
2 3
-3 -1
-3 -2
3 1 2
```

By adding an atomic sentence or its negation and testing for satisfiability, we can show there is a single valid model for this problem. One simple satisfiability testing method is a stochastic local search such as WalkSAT. However, a simpler variant of WalkSAT suitable for introductory courses is possible:

- Generate a random model
- Find all unsatisfied clauses for the model
- *iterations* = 0
- While there are unsatisfied clauses and *iterations* < *max*:
    - Pick a random variable of a random unsatisfied clause and negate it in the model
    - Find all unsatisfied clauses for the new model
    - If more clauses are unsatisfied then revert to the previous model with high probability *p*
    - *iterations* = *iterations* + 1
- Return whether or not clauses are still unsatisfied

Experimenting with parameters *max* and *p*, it is not difficult to quickly and reliably test satisfiability for simple logic problems. Before assigning this algorithm, the instructor may use the Clue project documentation to explain propositional logic basics, including CNF, models, (un)satisfiability, modus ponens, and proof by contradiction. If the instructor supplies CNF encodings of word problems (e.g. those of the MLExAI project), training representation and conversion to CNF is not necessary. With this WalkSAT simplification, students also gain experience with stochastic local search [13].

Conclusions

Introductory level AI challenges allow students to look beyond the programming fundamentals to the engaging challenges that lie beyond. However, such fundamentals are important, so introductory AI challenges should have a relatively low entry cost to be suitably accessible. With this in mind, we have extended and presented here for the first time a number of introductory AI projects we believe will be significant, engaging introductory CS experiences that point beyond the fundamentals, inviting students to explore further challenges in our field.

The core projects we have referenced and extended received positive evaluation, thus the NSF has funded phase 2 of the CCLI project. Twenty faculty members are currently adding to this repository of machine learning laboratory experiences for introducing undergraduates to AI.

Acknowledgments

This work is supported in part by NSF grant DUE CCLI-A&I Award Number 0409497.

References

[1] Carnegie, Dale, *How to Win Friends and Influence People*, revised ed., Simon & Shuster, New York, NY, 1981.

[2] Dodds, Z. et al. (eds.), *Special Issue on Robots and Robotics in Undergraduate AI Education*, AI Magazine, 27(1), AAAI Press, 2006.

[3] Fox, S., Artificial Intelligence and Robotics Lab, http://www.macalester.edu/research/fox/.

[4] Fox, S., Introductory AI for Both Computer Science and Neuroscience Students, In *Proceedings of the 20th International FLAIRS Conference (FLAIRS-2007)*, AAAI Press, May 2007.

[5] Greenwald, L. et al (eds.), Accessible Hands-on Artificial Intelligence and Robotics Education, AAAI Press Technical Report SS-04-01, March 2004.

[6] Harlan, R., Levine, D., and McClarigan, S., The Khepera Robot and the kRobot Class, in *Proceedings of the 32nd SIGCSE technical symposium on Computer Science Education*, ACM Press, New York, NY, February 2001, 105-109. DOI= http://doi.acm.org/10.1145/364447.364553

[7] Hearst, M. (ed), Improving Instruction of Introductory Artificial Intelligence, Technical Report FS-94-05, AAAI Press, 1995.

[8] Kumar, A., Kumar, D., and Russell, I., Non-Traditional Projects in the Undergraduate AI Course. In *Proceedings of the 37th Annual SIGCSE Technical Symposium on Computer Science Education*, ACM Press, March 2006. DOI= http://doi.acm.org/10.1145/1121341.1121491.

[9] Kumar, D., and Meeden, L., A Robot Laboratory for Teaching Artificial Intelligence, In *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education*, ACM Press, New York, NY, 1998, 341-344. DOI= http://doi.acm.org/10.1145/273133.274326.

[10] Markov, Z. and Larose, D. T., *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*, Wiley, 2007.

[11] Markov, Z., Russell, I., Neller, T., and Zlatareva, N., Pedagogical Possibilities for the N-Puzzle Problem, in *Proceedings of the 36th Frontiers in Education Conference*, IEEE Press, November 2006.

[12] Markov, Z., Russell, I., Neller, T., and Coleman, S., Enhancing Undergraduate AI Courses through Machine Learning Projects, In *Proceedings of the 35th Frontiers in Education Conference*, IEEE Press, October 2005.

[13] Neller, T., Teaching Stochastic Local Search, in I. Russell and Z. Markov, eds. *Proceedings of the 18th International FLAIRS Conference (FLAIRS-2005), Clearwater Beach, Florida, May 15-17, 2005*, AAAI Press, pp. 8-13.

[14] Neller, T., Russell, I., Presser, C., and Markov, Z., Pedagogical Possibilities for the Dice Game Pig, *Journal of Computing Sciences in Colleges*, 21(5), April 2006.

[15] Neller, T., Markov, Z., and Russell, I., Clue Deduction: Professor Plum Teaches Logic, In *Proceedings of the 19th International FLAIRS Conference (FLAIRS-2006)*, AAAI Press, May 2006.

[16] Russell, I., Neural Networks: Theory and Applications, *Journal of Undergraduate Mathematics and its Applications (UMAP)*, January 1993.

[17] Russell, I., A Neural Network Simulation Project, *Journal of Artificial Intelligence in Education*, February, 1992, Vol. 3, No. 1.

[18] Russell, I., Neural Networks in the Undergraduate Curriculum, *Journal of Computing in Small Colleges*, April 1991, Vol. 6, No. 4, April 1

[19] Russell, I., Markov, Z., Neller, T., Georgiopoulos, M., and Coleman, S., Unifying Undergraduate AI Courses through Machine Learning Projects, in *Proceedings of the 25th American Society for Engineering Education Conference*, June 2005. DOI= http://doi.acm.org/10.1145/1140124.1140230

[20] Russell, I., Markov, Z., and Neller, T., Teaching AI through Machine Learning Projects, in *Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education*, ACM Press, June 2006. DOI= http://doi.acm.org/10.1145/1140124.1140230

[21] Russell, I., Markov, Z., and Zlatareva, N., Introducing Machine Learning from an AI Perspective", *Proceedings of the 13th International Conference on Artificial Neural Networks*, June 2003.

[22] Russell, I. and Neller, T., Implementing the Intelligent Systems Knowledge Units of Computing Curricula 2001, In *Proceedings of Frontiers in Education Conference*, Boulder, Colorado, November 5-8, 2003, IEEE Press.

[23] Russell, S. J. and Norvig, P., *Artificial Intelligence: a modern approach*, Upper Saddle River, NJ: Prentice-Hall, 2nd ed., 2003.

[24] Witten, I. H. and Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., Morgan Kaufmann, 2005.

[25] Wilensky, U., Abstract Meditations on the Concrete and Concrete Implications for Mathematics Eductation. In Harel, I. and Papert S. (eds.), *Constructionism*, Ablex: Norwood, NJ, 1991, 193-203