Making Research Tools Accessible for All AI Students

Zachary Dodds, Christine Alvarado, and Sara Owsley Sood

Harvey Mudd College Computer Science
301 Platt Boulevard
Claremont, CA 91711
dodds@cs.hmc.edu, alvarado@cs.hmc.edu,

Pomona College Computer Science
185 East Sixth Street
Claremont, CA 91711
sara@cs.pomona.edu

Abstract

The AI community is constantly creating and improving sophisticated tools for visualizing, understanding, and investigating AI algorithms. Because these tools are designed by and for researchers, they do not always present a welcoming interface to students learning the discipline for the first time. This paper presents a set of short assignments that scaffold three freely available libraries of use in AI investigations. We have found that activities such as these give undergraduate students the confidence and capabilities crucial to future success with open-ended AI investigations. In addition, the self-guided tasks free classroom time from software details that are better learned in a hands-on setting.

Research tools: risks vs. rewards

First and foremost, we teachers of undergraduate AI strive to instill excitement and self-confidence among our students. We want them to be able to read and appreciate the context of current AAAI and HCI research. In addition, AI electives often ask students to gain hands-on experience by implementing a medium-sized project, sometimes of their own design.

To be efficient in balancing background knowledge and hands-on experience, students must build atop the resources – libraries, toolsets, and algorithms – that AI researchers constantly create and improve. Examples include graphical modeling tools (Bilmes and Zweig, 2002), robot control libraries and computer vision infrastructures (Montemerlo et al., 2003; Vaughan et al. 2003), user input and support processing of HCI data (Myers et al., 2000), and support for natural language work (Bird, 2005), to name but a few. Such resources complement from-scratch algorithm development in class: students can explore far broader swaths of AI far more deeply by standing on the shoulders of prior work.

Yet even well-established research tools do not always make ideal pedagogical platforms. Such tools might emphasize performance and comprehensiveness over accessibility. They may have deep dependence chains that make setup or configuration challenging for students hoping to run on their own computers. Documentation may be incomplete or inaccessible to first-time users.

Similar drawbacks haunt students' first exposure to computer programming. A language's feature set, the IDE-compiler-interpreter setup, and the mental models assumed by the documentation all conspire against the tentative student. We thus emulate the thematic approaches of today's transformative CS 1 curricula (Bayliss and Strout, 2006; Blank, 2006, Guzdial, 2003, Kumar et al., 2008). A gently-graduated set of motivating tasks wraps skill-practice and fundamental concepts. Later assignments then leverage students' confidence and independence in order to deepen computational investigations and understanding.

Contributions

This paper provides a detailed overview of three assignments used in AI projects and AI classes at the Claremont Colleges:

- **RTS** The *Reasoning Through Search* library (RTS) offers students a means to explore – and innovate with – emotional classification of text in only one semester.
- **WPF** A journal-like drawing application can motivate students to delve into the *Windows Presentation Foundation*, a fundamental resource for investigating pen-based computing and user interfaces.
- **OpenCV** We also outline an introduction to the OpenCV library of computer vision routines through the implementation of the game of *Set*. (Set, 1991).

Each of these tasks pushes students to gain familiarity with a useful AI framework that might otherwise seem daunting. At the Claremont Colleges, students have shown that they subsequently delve far beyond the confines of these introductory tasks in pursuit of more advanced, open-ended projects. These carefully structured exercises might be adapted – or adopted as-is – in other courses and contexts. We hope that these examples bring other such techniques, either already in use or yet-to-be-created, to the attention of the AI education community.

RTS – Reasoning Through Search

Emotional computation is a rising subfield of AI, with an end goal of creating machines that have higher emotional intelligence (Picard, 1997). Researchers are taking many different approaches towards developing machines with higher emotional intelligence, including empowering machines to detect emotion, to understand and manage and emotional connection with a user, and to express emotion in their communication with a user (Sood, 2008).

A large area of research towards this goal involves building systems that detect emotional content in text. While classification problems often yield to well-known machine learning approaches, emotional classification has an added complexity in that words carry different emotional connotations when used in different contexts or when describing objects in different domains. Describing a politician as "cold" typically carries a negative meaning, while a "cold" beverage is typically positive. Given this disparity, building a general emotional classification tool is a daunting task. In fact, using a Naïve Bayes approach to emotional classification without a domain specific approach only yields 66% accuracy.

As an attempt to build a general-purpose emotional classification system, the RTS (Reasoning Through Search) system was built (Sood, 2007). The system's goal is to classify a given piece of text as a valence/intensity score between -1 and +1 (-1 corresponding to most intensely negative, +1 corresponding to most positive). RTS is trained on 106,000 movie and product reviews, using the number of stars attached to each review as truth data. It uses a combination of Naïve Bayes and Case Based Reasoning approaches with techniques from Information Retrieval to classify text with an accuracy of 78%. The combination of approaches results in a system that preserves data from individual domains, while benefiting from the robustness of a Naïve Bayes approach.

Given the complexity of the task, it is infeasible for a student to complete any class or research project that includes an emotional classification component within a single semester. Emotional computation is a compelling area of research with many interesting applications; however, with a lack of high quality, well-documented open source tools, any research task involving emotional classification is overwhelming. To this end, I make the *RTS* system, including its training data and a paper documenting how it works, available to students for independent research projects, as well as in Introductory AI courses as a way to explore the tradeoffs inherent in various machine learning approaches.

The Task: Visualizing blogger sentiment

A current Pomona College Senior has made use of the *RTS* system in various aspects of her senior thesis project. The goal of her project was to build a system that, when given a particular weblog of interest, could examine that blog, tracking topics addressed and the blogger's sentiment towards those topics as they changed over time, and create a visualization of this information. This project was too large in scope to allow the student to build her own emotional classification system from scratch. For this reason, she used the RTS system and its underlying data in her project.

The Results

The student was able to build a system that can create three different types of visualizations, two of which are shown below. The first, shown in Figure 1, uses the RTS system to analyze each entry in the blog that includes topics of interest – topics that the system discovers on its own. The end result is a dashboard displaying the blogger's sentiment toward that topic when they last wrote about it.

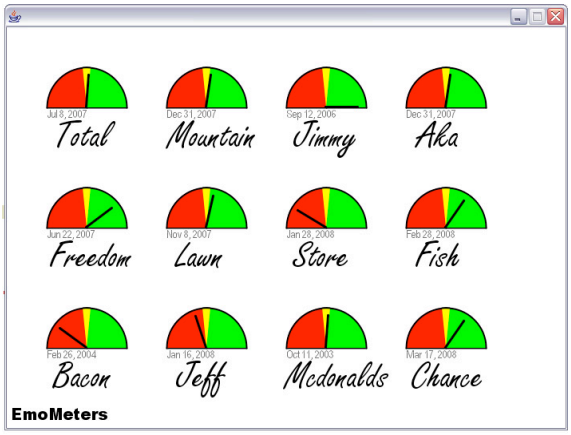

Figure 1 EmoMeters - a visualization of blogger sentiment towards topics of interest.

Figure 2 EmoCloud - a visualization of blogger sentiment surrounding a focus topic.

Figure 2 shows a second visualization, a tag cloud where the centroid is a topic and the other terms are highly emotional terms that occurred in various blog entries that mentioned the centroid topic. Here, the student did not use the RTS system itself, but instead used the data behind RTS in an innovative way. That data enumerates each term's occurrences in positive documents and negative documents; the data includes, in turn, the Bayesian

probability that a document containing each term is positive (and negative). By comparing a term's positive-document and negative-document probabilities, the student was able to quantify how "emotional" the term was. The system decides which words to display in the tag cloud based on this calculation.

By using the RTS system and the data behind it, this student was able to build compelling visualization systems that would not have been possible otherwise. In the end. the RTS tool enabled her to make a contribution towards building systems with emotional intelligence.

A *Foundation* in Pen-Based Computing

Windows Presentation Foundation

With the rise of pen-based computing (PBC), courses dedicated wholly to this topic are beginning to emerge at colleges and universities across the country. These courses aim to teach students fundamental algorithms for working with ink, and often focus in particular on techniques for recognizing hand-drawn input, such as text, gestures or sketches.

As in many other areas of AI, in order for students to have time to focus on implementing interesting algorithms, they must be provided with a platform for handling mundane tasks, such as collecting and displaying users' hand-drawn strokes. While some instructors still use home-grown software platforms (Hammond, 2007), others leverage Microsoft's new platform: Windows Presentation Foundation, WPF (LaViola, 2007). WPF offers functions for collecting, displaying, saving and loading ink on a Tablet PC. It also provides simple gesture recognition, a lasso selection mode in which users' strokes select existing objects rather than create new strokes, handwriting recognition, and even some advanced ink analysis such as routines for classifying strokes as drawings or text.

However, despite its power, WPF forces students to work in an environment that is likely completely unfamiliar. WPF runs in the .NET platform and is best leveraged using a combination of XAML (eXtensible Application Markup Language) and C#, languages that students have little or no experience with in other classes. Furthermore, the data structures and functions for storing and manipulating ink require some time to become comfortable with.

Creating a Journal Application

To help students become familiar with WPF, XAML and C#, the first assignment in our PBC course is to build a simple drawing application, similar to Windows Journal. This approach is not unique to Harvey Mudd; in fact, our assignment borrows from the first assignment in the PBC class at the University of Central Florida (LaViola, 2007).

The goal of this assignment is to help students acquire the range of skills with WPF that they will need throughout the rest of the class. Thus, this assignment includes pieces that help students learn the following:

- basic application functionality (e.g., loading and saving ink, opening and closing the application),
- working with ink (e.g., collecting ink, changing the pen into an eraser, changing to selection mode),
- working with ink data (e.g., adding timestamps to collected ink points).
- using the built-in recognizers (e.g., recognizing text and simple gestures).

In addition to these required pieces, we also encourage students to implement one or more optional extensions that allow them to express their creativity without becoming overwhelmed. These optional extensions are very open-ended, but might include more advanced controls on the interface, more sophisticated recognition functionality, or more robust data formatting.

Experiences

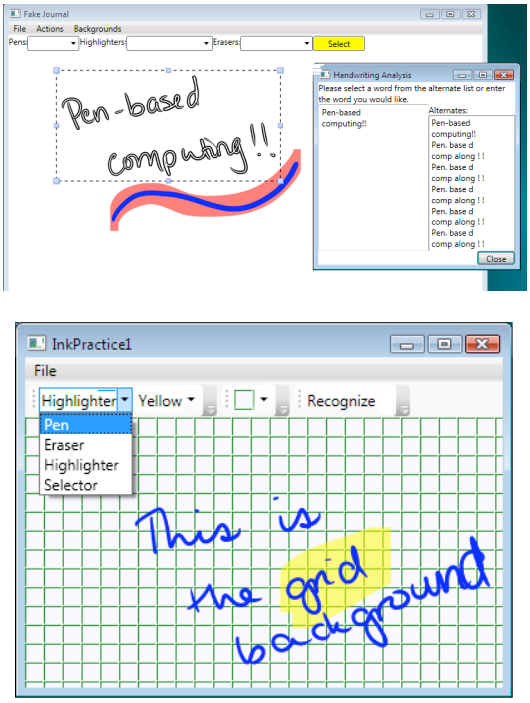

Figure 3 Two examples of student-designed interfaces to the initial course project, a Windows-Journal-like application.

Students have produced a range of implementations for this assignment. Everyone was able to implement the required functionality, but no two interfaces looked the same. For example, some students built complex interface controls consisting of sophisticated drop-down combo-boxes

(Figure 3, bottom), while others developed advanced recognition correction interfaces (Figure 3, top).

The biggest challenge for students on this assignment was to get used to the integration of XAML (the markup language in which all of the interface components are laid out) and C# (used for programming most of the application's functionality). Some students really leveraged the interplay between the two languages, while others chose to implement their application mostly in C#. Both approaches were fine, as they both provided students a framework for their assignments going forward. In fact, many students continued to build off of their code for future assignments.

Attaching timestamps to individual points in the strokes adds another challenge. While point-level timestamps are crucial for many recognition algorithms (Sezgin, 2001), they are not well supported in WPF. Students had to handle individual `PenMoved` events, extract the timing data from these events, and then attach this data to the relevant points via the Extended Properties field of the stroke. This process is not extremely elegant and often leads to several points having the same timestamp.

Results

Based on direct feedback from students after the first assignment, it was successful at meeting its goals. Students reported that it took between 8 and 15 hours to complete, making it appropriate for a one and a half to two week assignment. All students who responded to the post assignment survey (3 of 5) agreed or strongly agreed that the assignment was fun and helped them learn the platform well. In the words of one student:

> *I think it did a good job of covering the bases we'll need while providing some fun distractions as well. I can't really think of anything to change with it.*

However, this introductory assignment and WPF in general do have some limitations that students had to overcome in subsequent weeks. One student commented that even after completing the assignment he was still having trouble knowing how to write clean, elegant code (as opposed to fast, ugly code) in C#. Also, WPF is fundamentally missing some important functionality for working with ink. For example, it does not support a clean way of saving and loading labeled data (i.e., pen strokes that are grouped and associated with a particular label indicating what symbol the strokes represent), so students have had to implement their own code to work around this limitation.

Despite these limitations, students have successfully implemented three additional three-week assignments in WPF exploring fundamental algorithms for ink processing including low-level stroke processing (including finding corners in strokes), gesture recognition, and higher-level sketch recognition. Completing these ambitious assignments in such a short timeframe would not have been possible without being comfortable with WPF and all of the infrastructure it provides for working with ink data.

Getting *Set* with OpenCV

OpenCV

OpenCV is a C and C++ library of computer vision and image processing algorithms that grew from an Intel Labs project to demonstrate the performance of that company's hardware (OpenCV). As OpenCV has matured, robotics and computer vision researchers have transformed it into an open-source project that runs well on Windows, Mac OS X, and Linux systems (Bradsky and Kaehler, 2008). It supports display and image acquisition from many movie and still image formats, as well as most off-the-shelf cameras, including webcams. An active development team, led by Mark Asbach, continues to refine the library for a worldwide audience. One claim to fame was its use as the foundation for Stanley's vision system in 2005.

Despite these features, OpenCV is not immediately accessible even to computationally savvy students. Setup can be nontrivial, as suggested by the large number of tutorials online and the very recent creation of a Mac OS X framework build. Example programs are plentiful, but they highlight single image-processing algorithms, rather than presenting the I/O and data-structure foundation that would open the library to independent experimentation. *Set*, below, motivates precisely this kind of student exploration.

The Task: implementing *Set*

Set is a face-up card game introduced in 1991. It has gained in popularity and notoriety, culminating in its inclusion alongside the crossword in the New York Times online puzzle page. Each card contains an image with four attributes: *the number of symbols* (1-3), *their color* (red, green, or blue), *their shape* (oval, diamond, or squiggle), and *their fill texture* (empty, striped, or filled). The 81 possible combinations of these attributes comprise a complete deck of Set cards: figure 4 shows six of these.

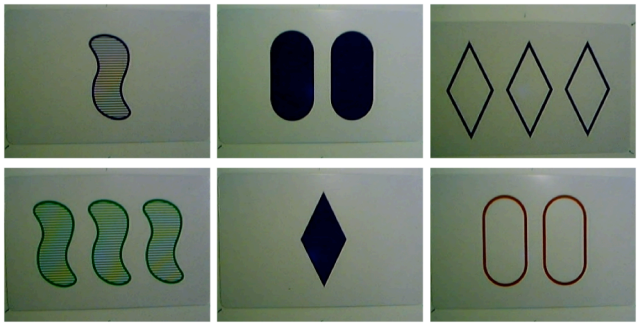

Figure 4 A sample of six of the 81 Set cards. The game's object is to find *sets*, such as the three cards in the top row.

In the game a dozen or so cards are dealt face-up in front of all players. The object is to find three cards from that group in which each attribute's value is identical or distinct among the three cards. For instance, the three cards in the top row of Figure 1 form a set, because

- Each of the three has a distinct *number* of symbols.
- Each of the three has the same *color*, blue.
- Each of the three has a distinct *shape*.
- Each of the three has a distinct *fill texture*.

Experiences

Often students are familiar with *Set* and find the game challenging. As a result, many find the task of building a set-playing program immediately motivating. What's more, writing a Set-player provides a compelling and concrete example of how tasks that we find difficult – identifying legal sets when observing a group of cards – can differ from tasks that computers (or, perhaps, computer programmers) find difficult – determining what symbols are, in fact, on the face of each card.

The task of building a set-playing program begins with a working C++ skeleton file (provided) demonstrating how to include necessary headers, compile and link against the library, and interact with images from a folder, movie file, or camera. In order to avoid – at least initially – difficult or changing lighting, the starting files include a folder of the 81 card images. The assignment guides students to explore facets of the library as they accomplish portions of the overall task:

1. They write code to interact with the keyboard, mouse events, and files; they use these skills to change program parameters such as color thresholds without having to recompile and rerun.
2. With appropriate thresholds, the **`cvBlobsLib`** region-extraction library enables segmentation of each of the symbols on any card. *Number* is often the first characteristic determined.
3. Students have more than 20 computed statistics from which to determine *shape*. Raw area and perimeter tend to fail because the interior of many symbols are segmented as distinct regions due to the lighting, even in the provided images. Convex perimeter and filled area, however, do linearly separate the three shapes.
4. *Color* and *texture* are often addressed at the pixel-level, rather than the region-level. It is important that students gain confidence with both.
5. The assignment requires students to familiarize themselves with the graphics-overlay library of shapes and text – these are skills crucial for future experimentation with vision algorithms. They use these calls to identify each card – and their *Set*s.

These subtasks admit a number of natural extensions. For example, live images of the cards from an attached camera motivate the use of OpenCV's perspective unwarping functions; the addition of a game-playing or parameter-setting interface is easiest through a socket connection, again provided, to OpenCV: this software structure is equally useful for integrating sensors and actuators for robotic projects. Variants such as *Super-Set* offer challenges to the processing *after* identifying the cards.

Results

Teams of 2-3 students report that this assignment ordinarily takes between three and six hours: perhaps a reasonable one-week project, or two weeks with some of the extensions mentioned above.

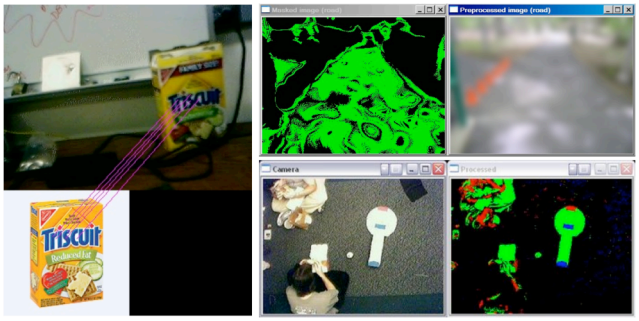

Figure 5 Three student-designed and -implemented vision systems stemming from *Set*: SIFT feature matching (left), sidewalk segmentation (upper right), and off-board observation and control of an iRobot Create trailer-backing robot (lower right)

The resulting facility with OpenCV has helped our students see real-time vision as a useful tool, rather than a source of difficult systems-integration issues (as had been the case). Students in recent offerings of our undergraduate robotics elective have leveraged this initial experience into the systems depicted in figure 5, which

- incorporate the scale-invariant feature transform (Lowe, 2004) in order to robustly recognize objects,
- extract, process, and model campus paths in order to set steering angles for autonomously navigating vehicles,
- use image streams in order to compute the appropriate control for a tractor-trailer-reversing robot,

among many other applications. Although all of this certainly could have been accomplished without the *Set* assignment, having such an introduction seems to have made a difference. Within the course prior to *Set*, only 4 of the 19 (21%) of the open-ended student-designed projects used real-time vision. When the assignment has been used, 14 of the 24 (58%) of those projects did so. Certainly other factors play a role, e.g., improvements in OpenCV, but *Set* has also contributed to this increase.

The assignment page (www.cs.hmc.edu/twiki/bin/view/Robotics/GettingSetWithOpenCV) offers detail sufficient for a student to create a *Set*-playing program even without supervision.

Perspective

These three tasks -- leveraging the *Reasoning Through Search* toolkit and data, building a journal-like application via the *Windows Presentation Foundation*, and writing a Set-player using *OpenCV* -- demonstrate that learning a fundamental AI tool can be seamlessly integrated into the early part of an undergraduate course or project. By combining several layers of Bloom's learning-behavior taxonomy into a single exercise, such assignments increase the efficiency of AI and HCI education. Students can exercise their creativity and deepen their understanding of a topic, even as they familiarize themselves with the low-level details of important software tools.

Tools such as RTS, WPF, and OpenCV are only part of a huge – and constantly growing – group of resources contributed by the many communities that make up AI. We hope that these example tasks offer other instructors easily-adaptable assignments for their students. More important, however, will be our community's creativity in designing and releasing these kinds of *curricular bridges* that can help make the rich research field of AI inviting and accessible for students with a broad variety of backgrounds and interests.

References

Bayliss, J. and Strout, S. 2006. Games as a "flavor" of CS 1. *Proc., SIGCSE '06* Houston, TX, USA, ACM Press, pp 500-504.

Bilmes, J. and Zweig, G. 2002. The Graphical Models Toolkit: an open source software system for speech and time-series processing. *Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*. IEEE Press, pp IV-3916-3919.

Bird, S. 2005. NLTK-Lite: Efficient Scripting for Natural Language Processing. *Proceedings 4th International Conference on Natural Language Processing (ICON)*, Kanpur, India, pp. 1-8.

Blank, D. 2006. Robots make computer science personal. *Communications of the ACM* 49(12) December, 2006, ACM Press, pp 25-27.

Bradsky, G. and Kaehler, A. 2008. *Learning OpenCV: Computer vision with the OpenCV library* O'Reilly Media.

Guzdial, M. 2003. A media computation course for non-majors. In *Proc. ITiCSE '03*, ACM Press, pp 104-8.

Hammond, T. Sketch Recognition Course, Texas A&M, http://faculty.cs.tamu.edu/hammond/courses/SR/2007/

Kumar, D., Blank, D., Balch, T., O'Hara, K., Guzdial, M., and Tansley, S. 2008. Engaging computing students with AI and Robotics. *Proc. of the AAAI Spring Symposium on Using AI to Motivate Greater Participation in CS: Technical Report SS-08-08* AAAI Press, pp. 55-60.

LaViola, J. Topics in Pen-Based User Interfaces Course, UCF. http://www.eecs.ucf.edu/courses/cap5937/fall2007/

Lowe, D. 2004. Distinctive image features from scale-invariant viewpoints. *International Journal of Computer Vision* 60(2), Kluwer, pp. 91-110.

Maclagan, D. and David, B. 2003. The card game Set. *The Mathematical Intelligencer* 25(3), Springer, pp. 33-40.

Montemerlo, M., Roy, N. and Thrun, S. 2003. Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit'. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS '03). Las Vegas, October, 2003, IEEE Press, pp 2436-2441.

Myers, B., Hudson, S.E., and Pausch, R. 2000. Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction* (TOCHI), ACM Press, 7(1), pp 3-28.

OpenCV. http://www.intel.com/technology/computing/opencv/

Picard, R. 1997. *Affective Computing*. MIT Press.

Set. 1991. Set Enterprises. http://www.setgame.com/

Sezgin, T. M., Stahovich, T., and Davis, R. 2001. Sketch Based Interfaces: Early Processing for Sketch Understanding. *Proceedings, Workshop on Perceptive User Interfaces*, Orlando FL.

Sood, S. Owsley. 2008. Emotional Computation in Artificial Intelligence Education. *Proc., AAAI AI Education Colloquium*. Chicago, IL, AAAI Press.

Sood, S., Owsley, S., Hammond, K., and Birnbaum, L. 2007. Reasoning Through Search: A Novel Approach to Sentiment Classification. *Northwestern Univ. Technical Report* NWU-EECS-07-05.

Vaughan, R., Gerkey, B., and Howard, A. 2003. On Device Abstractions For Portable, Resuable Robot Code. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems*. (IROS '03), Las Vegas, USA, October 2003. IEEE Press, pp. 2121-2427.

Wu, D. and Uken, N. 2005. Some exploration of the Game Set. *International Journal of Mathematical Education in Science and Technology* 36(1), Jan.-Feb. 2005, Taylor and Francis, Ltd, pp. 93-95.