Jason M. Eisner

Teaching Statement (August 2013)

Teaching Performance and Feedback

Teaching JHU students has been gratifying because they are willing to *enjoy* hard courses. Students consistently rate my classes both difficult[1] and time-consuming.[2] But they are also highly rated overall.[3] Most students strongly agree that I inspired their interest in the subject matter,[4] that they learned from the lectures and homeworks,[5] and that the class improved their ability to solve problems with computers[6] and with mathematical methods and data.[7]

I was pleased to receive a teaching award in 2005, which was presented with the following quotations from student nominations:

- Armed with a hilarious sense of humor, a great skill for captivating lectures, and an enormous amount of knowledge . . . Dr. Eisner knows how to keep students interested, motivated, and even entertained at the same time. . . . He continually made a strong effort to know how each and every one of the students in his course were doing—asking, by name, not only about the successful students, but about the struggling ones as well.
- Dr. Eisner gives an enormous amount of time and effort to teaching his classes and demands the same effort from his students. . . He is like no other professor I have had. He makes students feel as comfortable as possible in participating in class by finding some good in each comment a student makes.
- I feel a sense of satisfaction every day when I have his class that I have pushed myself to learn something new or to think about something in a way that I had never before.
- In my four years at Hopkins, Dr. Eisner has undoubtedly been the best professor that I've had in any class at this University.
- In my four years at Johns Hopkins, I haven't had a professor who has given so much to his students and who has gotten so much from them.

A few other quotations from students' nomination letters are also worth repeating. These give a more concrete sense of how lectures, assignments, and other elements function in my courses:

- From the first class, and throughout the semester, Prof. Eisner's excitement and enthusiasm in teaching the material stood out. Prof. Eisner took the time to prepare detailed lecture slides to enable his students to better understand the concepts being taught. His assignments were challenging, but were very effective in helping to understand how theoretical concepts have practical applications. He engaged the students in his lectures, making classes interesting with various examples. What must also be mentioned is Prof. Eisner's dedication to his students. He took the time to answer questions carefully and in detail,

[1] Average rating for "intellectual challenge" in my lecture courses was 4.70 / 5 across all ACE surveys that asked this question (Fall 2004–Spring 2011).

[2] I often hear that my courses have this reputation. The workload score in the ACE surveys is "only" 4.22, where 5 is "much higher workload than other courses."

[3] ACE average 4.50 for overall quality.

[4] ACE average 4.74.

[5] ACE averages 4.70 and 4.72, with a teachig effectiveness average of 4.68

[6] ACE average 4.69.

[7] ACE averages 4.44 and 4.49 in the NLP course.

whether they be in class or by email late at night. It was amply evident that he cared greatly about his students understanding the material.

- His assignments are not toy assignments; they are real world problems. In times when engineering is becoming so specialized and detached in some sense from the real world, his problems remind you the real reason a problem is studied.
- The assignments in both Data Structures and Natural L[an]guage Processing are like no other assignments I have had in any other class. A typical assignment is 20 pages long and includes plenty of background reading, carefully constructed supporting programs, relevant commentary and inspiration to the student, and extremely explicit instructions. I am pushed to understand and apply concepts in completing these homework assignments so that by the time I am done, I understand the material infinitely better than I previously had.
- He teaches a very difficult subject, Natural Language Processing, but does so with extreme caution and care that his students understand. He always goes the extra mile to make sure you understand. He is an interesting and captivating lecturer with a good sense of humor, and above all, gives incentives to his students to think out of the box and apply their knowledge to a wide spectrum of problems.

The courses themselves are described on my CV. For my tenure case in 2007, I excerpted the following selection of comments from anonymous course evaluations, which again convey how my pedagogical choices have worked out in practice (for good and for ill):

- This was an amazing course, although I think I am now failing all my other courses due to the workload. [600.226 Data Structures]
- Dr. Eisner is hilarious. Very interesting lectures. Makes difficult material fun. Always answers questions well and is enthusiastic in class. [600.226 Data Structures]
- The homeworks were very challenging, yet interesting and very engaging, dare I say fun. [600.226 Data Structures]
- We love random anecdotes. [600.226 Data Structures, never dreaming that his/her comment would *become* a random anecdote]
- Excellent instructor—Dr. Eisner is clearly brilliant and a very engaging lecturer. He expects a lot but it's worth it. [600.325 Declarative Methods]
- The class discussions were typically great. Eisner did have a tendency to go on tangents, but he was good about getting back to the material in a timely fashion most of the time. HWs tended to reinforce the material very well. [600.325 Declarative Methods]
- The lectures were very well organized and presented. The homeworks were very clear and guided students through a learning process rather than just testing recall. [600.425 Declarative Methods]
- A uniquely structured overview of an area I'd previously paid very little attention to. It was very enlightening. [600.425 Declarative Methods]
- Powerpoint presentation & lectures were phenomenal. [600.425 Declarative Methods]
- Strong relation between lectures and homeworks. [600.425 Declarative Methods]
- Eisner is a great professor, knows how to teach & develop a course. Assignments were helpful, well thought out. [600.425 Declarative Methods]
- The course covers material seldom encountered in other CS courses. Jason & Blatz [the TA] are both excellent at teaching & answering questions. Despite this being the very first time it's taught, the course was superbly organized & run; notifications always came promptly, and the slides are awesome! [600.425 Declarative Methods]

- Excellent and updated slides; thorough presentations; well-maintained course web page; genuine interest in the students (e.g. learning names); clear and detailed homework assignments. [600.465 Natural Language Processing]
- The professor is the best professor at Hopkins, period. He knows his stuff, he puts a lot of effort to make the lectures both interesting and challenging, has never managed to bore me. He answers questions promptly and goes the extra length to make sure you've understood. His assignments are extremely interesting Real World problems, and inspire a lot of interest in the subject matter (that is already exciting by default). [600.465 Natural Language Processing]
- Probably the most rewarding course I've taken during my time at Hopkins thus far. The material is very interesting, and Dr. Eisner has a knack of making it all seem very intuitive. Of course, it doesn't seem that way when it's time to implement the material in projects, but that just makes it more rewarding when the code finally works. [600.465 Natural Language Processing]
- The professor is a great lecturer. The amount of preparation he put into the slides, spreadsheets, and assignments is just staggering. [600.465 Natural Language Processing]
- Several assignments in this course could easily be semester projects in other courses. [600.465 Natural Language Processing]
- Extremely interesting, well-structured, stimulating lectures. Concerned about students. Good top-down representation in terms of topic detail. Ability to develop a strong interest in NLP and provided a medium-depth understanding of various models and subtleties of challenges. **This class reminds me why getting an education is actually worth it.** [600.465 Natural Language Processing]
- Eisner is everything a professor should be and then some. He is perhaps the best professor I have ever had, grad or undergrad. He is an invaluable asset to Hopkins, the field of Comp Sci, and any student lucky enough to have him in class. [600.465 Natural Language Processing]

Teaching Philosophy

For me, the central fact of teaching is that a complex network of ideas, examples, and techniques already hangs together in the teacher's head. This structure fits together so tightly that to the teacher, it seems compact, natural, and beautiful—a "mode of thought" that will apply to many new problems. The challenge is to unfold (much of) this structure into a linear, semester-long presentation that will fold up correctly again, like a protein, in a student's head. Then the student will "own" the major ideas of the course, each reinforced by many connections to other ideas (within and outside the course), multiple formalizations, generalizations, examples, and anecdotes.

In conveying this network of ideas, every element of the course has its place:

- **Lectures** serve both to convey technical intuitions and to excite interest. The two secrets to great lecturing, a colleague once told me, are clarity and a large personality.

  For clarity, I draw on my own mental habits. Much of my everyday technical thinking consists of searching for the "right" way to express some problem or solution. One derives much insight from a concise yet precise formulation, a crucial example or counterexample, or an apt visualization. Hence preparing for a class feels like research to me (well, research plus a lot of visual tinkering in Powerpoint). I put all my lecture slides online, and they have been used by faculty at several other universities.

  For personality, I do jazz up the occasional lecture by harking back to my theater days and using props or singing. More often, however, I enliven technical material simply by explaining it

with entertaining metaphors or whimsical examples. A good metaphorical explanation supplies plenty of opportunities for wit. (I try to keep the jokes connected to the material, since they're the only thing the students will still remember in 10 years.)

- **Class discussion.** There is a third secret to great teaching, I think: one must care about the students' learning rather than just one's own wind-up act. I learn all the students' names, watch their reactions, ask questions to check their understanding, and am willing to take time out to discuss even tangential questions, connections, and proposals from the audience.[8] It seems to me that to cultivate a spirit of intellectualism, connections to any field of inquiry have to be fair game.

  My favorite classes are those where I pose a problem at the blackboard and coax the class to come up with a series of increasingly better solutions. I ran the entire sophomore-level Data Structures course this way—helping the students reinvent the basics of algorithm design and analysis. (They missed being first by only a few decades!)

  Since all my lecture slides are available online, class attendance is a useful test of whether I am using classroom time effectively. In the end, the in-class expositions, discussions, and jokes seem to add enough value that students report a 90% attendance rate at my classes.[9] I emphasize on the first day that classroom time is interactive time. A student who is just going to listen passively would be better off saving the JHU tuition and just watching DVDs of the world's best solo lecturers.

- **Manipulatives** are physical objects used to teach elementary school math. We need more of them for higher math. When you really understand a mathematical object, you have an almost tactile sense of how it will behave when you twist it or drop it, and what might break it. I am often frustrated that the students can't see or manipulate the pictures zooming around in *my* head.

  I think the solution is to build interactive software visualizations that students can play with. In 2012, my TA Frank Ferraro and I developed one for log-linear models, aimed at students in NLP and ML. In 2002, I developed well-known visualizations[10] for hidden Markov models (HMMs) and Gaussian mixture models (GMMs), based on spreadsheets so that students can examine the intermediate computations. In 2006, with a team of undergraduates, I built a system for visual exploration of large graphs, which can be useful for understanding the information flow through an algorithm. I am now starting to design a more general platform for rapidly constructing interactive visualizations, to be built atop the Dyna programming language.

  Playing with other kinds of software can also be helpful. I devised a competitive grammar writing game, supported by a few software tools, that exposes students to a lot of important NLP concepts in the course of an afternoon. I've run this game annually since 2001, and it was also adopted by Stanford's NLP MOOC.

  (All of the above innovations have been published.)

- **Puzzles** can draw students into a field through the pleasure of problem-solving. Puzzles can both expose students to concrete data and goad them into discovering abstract concepts. Since

[8] Average ACE rating was 4.90 for "treated all students with respect," 4.78 for "genuinely concerned/interested that I learn the material," and 4.81 for " answered questions thoroughly/well."

[9] Average ACE self-rating over ACE surveys that asked this question.

[10] Used at some other universities, and featured in the leading NLP textbook.

2006, I have been on the problem-writing committee of the North American Computational Linguistic Olympiad, which selects the U.S. team for the International Linguistic Olympiad. Our puzzles have appeared in a new "recreational linguistics" book series from Springer, and I have used them in teaching freshmen and non-computer scientists. Recently I co-authored a paper about how to contruct puzzles of this sort.

- **Warmups** are an idea that I borrowed from the physics instruction community ("just-in-time teaching," "peer instruction") for my sophomore Data Structures class. I created fun half-hour online exercises to prepare students *just before* they started a new topic in class. The short-answer questions (graded for effort only) got the students thinking about particular data structure design issues, by making them discuss relevant examples or analogous problems from the physical world.

  Answers were due an hour before class. I always skimmed them before class to get a sense of where the students stood, and often brought up the warmup questions as we designed new methods during class—perhaps taking a show of hands, or calling on particular students to repeat their interesting or funny answers. My series of warmups has since been used by faculty at over a dozen other institutions.

- **Email** or a **discussion website** such as Piazza is a wonderful way to supplement class discussion. I frequently post thoughts and followups, and the TAs and I try to answer questions promptly. Students often chip in and help answer one another's questions. I've been pleased that both undergrads and grad students often ask advanced questions, including requests for further reading. I invite students to follow my example and share funny, interesting, or useful items that we run across (news clippings, videos, examples, readings, etc.).

- **Textbooks and other readings** can be helpful to students who didn't follow the in-class presentation. However, I always teach an idea first, and let the textbook back that up with another perspective.

  Usually, textbooks play only a secondary role in my classes. Like any active researcher, I prefer my own take on a subject. Moreover, I want the students to feel like active computer scientists, not consumers. Knowledge in my classroom derives from the professor (who knows current research), from the students (many of whom bring relevant knowledge or questions), and from the problem-solving that we do together.

  Sometimes an area is too new or idiosyncratic to even have any textbook, as in my graduate courses, or in my undergraduate Declarative Methods class (which I developed to understand a set of ideas myself).

- **Programming assignments** are the forum where a student masters the material, developing the details behind the classroom intuitions. My assignment handouts are often about 20 pages each, since they present some of these details and lead the students through working out the rest. With 5 or even 10 assignments per semester, the assignment handouts effectively serve as the main textbook.

  I want students to master something useful and know it. Thus, I often have them handle "real" (naturally occurring) data, such as sentences from the newspaper, handwritten digits, and election ballots. This makes for a satisfying assignment, but also requires them to handle various real-world engineering issues. In particular, a naive program might take months to

finish on real-world data; so the students have to engineer their way to an efficient program that finishes by the due date.

I have also experimented with teamwork and other kinds of assignments, including a nifty exercise in which teams compete in real time to write probabilistic grammars of English. In general, a bit of friendly competition can liven things up; for open-ended programming problems, I like to award chocolate for the fastest or most accurate solutions.

- **Term projects** could be a gateway to research. I have used them in only one class (the 400-level version of Declarative Methods). Results have been mixed, perhaps because the busy last month of the term is not enough time to undertake and complete a really interesting project. I am considering creating a new course in which projects are much more central and can already be attempted near the beginning of term.

- **Exams** are mainly useful, in my view, for getting students to consolidate everything they've learned during the semester—to "fold up the protein." To encourage this kind of deep study, I let the students practice on the exams (and solutions) from the past 2 years. I try to write engaging problems that the next year's students will be able to learn from. (Including many rote problems would wound my self-respect.)

Teaching Trajectory

My survey scores in all categories have been stable over time. They have not declined since I got tenure. Nor have they improved (perhaps because they were already high)—but I do put in work to improve my teaching from year to year:

- After giving a lecture, I make notes and revise slides to improve the presentation next time. I similarly improve assignments when I reuse them.

- Every year I feel that there are important ideas that I never laid out properly for the students. So I tend to add new material to a course each time I teach it, dropping other material to make room.

- In 2010, I started reviewing written advice with my TAs at the start of the term. This has improved the quality of TAing and has helped keep the grading on schedule, which students care about.

- In 2010, I started holding office hours in the classroom immediately after class, to encourage students to stay and talk.

- In 2011 I added optional weekly discussion sessions. Typically they are centered around challenging topical problems from past exams, which the students discuss and solve together under TA guidance.

- In 2011, I replaced my email lists with the Piazza discussion platform, discussed above.

- In 2012, I invested time in building a new interactive visualization, as noted above.

- In 2013, I taught a new 20-hour NLP course at the biennial Linguistic Institute of the Linguistic Society of America. Most of these students had not programmmed before. To reach them, I included pencil-and-paper puzzles that I had written for the North American Computational Linguistics Olympiad, and taught them a very high-level declarative programming language—of my own design—which we used to concisely express all the algorithms in lectures and homeworks. I also took a more relaxed attitude toward grading, deadlines, and teamwork. All of these moves were experiments with teaching techniques that I can now bring back to make my JHU course more accessible.

I would be happy to share my framing of the field with a wider audience. Many people have encouraged me to offer a MOOC (massively open online course) or write a textbook. Shortly after Sebastian Thrun at Stanford taught the first MOOC, he contacted me to ask if I would be willing to teach one on Natural Language Processing. I was unable at the time to figure out how to preserve course quality, but some ideas have been taking shape since then. I also have several ideas about idiosyncratic textbooks (or online tutorials) that would be effective.