Adam Meyers: Computer Science Teaching Statement

I learned to program while earning my PhD in linguistics at New York University (NYU). Formalizing linguistic concepts sufficiently to be coded helped me understand the linguistic concepts – learning to think computationally, helped me think formally. Writing papers about Computer Science (CS) research has improved my research, causing me to compare my work to previous work and to plan future experiments that were logically necessary to write the paper. In other words, thinking narratively helped me put computational ideas in context. I do my best to share both of these lessons with my students, in theory, forming a symbiotic relationship between narrative thinking and computational thinking.

Summary of Experience

My career in Natural Language Processing (NLP) began at IBM Research in 1989, while earning my linguistics PhD from NYU. I worked in research positions for about 25 years. For the other 9 years, I held teaching positions in NYU’s CS department. While in research positions, I taught 4 graduate classes as an adjunct: 2 Syntax (Linguistics NYU) and 2 NLP (CS NYU and Montclair University Linguistics). As an NYU Clinical Associate professor, I taught undergraduate NLP for many years and in 2022, began teaching a graduate NLP class as well. Apart from NLP, I have also taught Introduction to Computers and Programming for many years and 2 semesters of, ‘Programming Tools for the Data Scientist’. I also taught Computer Music at a pre-college class funded by an NSF grant which resulted in a conference paper presented at ACM Creativity and Cognition 2009.

Aided by my research experience, I have mentored many graduate, undergraduate students and post docs. I have been on several dissertation committees at NYU and advised two completed dissertations as an outside reader: Anabela Barreiro, 2009, Universidade do Porto and John Ortega, Universidad de Alicante 2021. My mentoring has resulted in several academic papers, some authored by students whom I mentored and others which I co-authored with those students.

Introduction to Computers and Programming

I have come to realize that computer scientists take a certain point of view towards tasks, which Jeanette Wing (Wing 2005, http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf) and others are calling “computational thinking”. This has shaped the way many people, in all walks of life, are solving all sorts of problems, not just programming problems. It is changing the way artists do art; businesses conduct business, scientists do science, etc. Techniques for planning algorithms can even provide insight into planning (and budgeting) non-computational projects (e.g., business plans). I believe that all students should get a taste of computational thinking, because it is part of a new paradigm of thought that is changing the world. Thus teaching students to program for the first time is very exciting to me. Students sometimes feel culture shock during the “Introduction to Computers and Programming” class, as they learn programming as a general method for solving problems.

I believe that teaching beginners to program requires a studio-like environment where students try out the concepts shortly after they learn them. NYU has adopted a “flipped classroom” approach for Introduction to Programming in which students complete much of the homework in class with support from tutors and the professor. Thus, my classes consist of alternations between in-class homework and conventional joint class activities (lectures, cooperative problem solving, review, etc.) The in-class homework problems reinforce related material right after it is taught. Other homework problems are completed at home or with the help of NYU tutors that are available at non-class times.
**Natural Language Processing**

In my Natural Language Processing (NLP) classes, I teach the combination of CS and descriptive linguistics needed for both basic NLP technologies and NLP applications. Basic technologies include parsing, part of speech tagging and semantic role labeling. NLP applications include machine translation, information extraction and ad-hoc information retrieval. During the first half of the semester, homework assignments include the creation of systems for performing tasks on pre-marked data sets, using primarily supervised machine learning (ML), e.g., HMM part of speech tagging, Information Retrieval, Maximum Entropy based Noun Group Chunking and semantic role labeling. During the second half of the semester, students work on individualized final projects (rather than specified tasks). The final project is completed in stages. There is typically a project proposal, a first draft, a conference-style talk and a final conference-length paper. In addition to the above structure, I assign each final project group a Teaching Assistant mentor to meet with on a weekly basis and track their progress. I have found that this helps keep students focused on doing a good job, including following good experiment design, completing work on a regular basis, and collaborating in an effective way.

Completed final projects have included corpus annotation (by students or by Amazon Mechanical Turk) with evaluation, implemented programs with evaluation, and projects that systematically evaluate existing systems. I encourage students to create simple base-line systems in addition to more complex systems that include experimental ideas. I also encourage students to do extensive error analysis. This ensures that the student gets results, even if they do not have a high-performing system. Students, with my support, have continued research after the term ended and developed these projects into published conference and workshop papers. Students from my classes often join my research efforts.

**Programming Tools for the Data Scientist**

The goal of this class is to build up the students’ computational literacy so they are ready to take NYU’s “Introduction to Data Science” class. It is geared for students minoring in Data Science who may have had only one previous programming class (e.g., “Introduction to Programming”) and one Data Science class (“Data Science for Everyone”). The class includes Python instruction not covered in the first programming class, shell commands and (bash) shell scripts, preparing data (compilation, curation and annotation), evaluation techniques, experimental methodology and measurement, basic machine learning principles, and the introduction to a few specific areas within data science (natural language processing, image processing, etc.). An effort is made to cover computational literacy in a way that supports, but does not duplicate the other introductory data science classes.

**Summary**

I believe that all educated people should be exposed to computer programming, as it represents a paradigmatic shift in human thought. In addition, many students studying STEM are missing out on the connection between writing and research. It turns out that the process of writing papers, especially constructing arguments, can cause students to do better experiments, write better code and integrate their work with other research. I believe that this omission can be filled, at least for CS students, by introducing them to the connection between writing programs and doing research papers. I believe that many students studying undergraduate data science, jump to process high level materials (e.g., about social or natural science) very quickly in a way that makes them unaware of certain basics, basics that will help them write better structured programs and perform well-formed experiments. Thus for all of these classes, I am hoping to educate students in a way that will help them beyond the simple content of the class.