Computer Science is a discipline combining the practical and the theoretical. Courses from algorithms to operating systems contain this blend: elements that can be expressed purely mathematically and elements tied to the concerns of building systems. In my area of focus, Natural Language Processing, this mixture is particularly important, and for me, the most exciting challenge of teaching is finding ways to join theory and practice such that they support each other and challenge students in complementary ways.

As a graduate student I have had many chances to experience the different facets of teaching in Computer Science. These experiences include:

- **Lecturing and Course Development.** I have co-taught a popular graduate-level class on Natural Language Processing at Columbia University. In addition to giving lectures, I developed the programming assignments and graded for the course.
- **MOOC.** I acted as the sole teaching assistant for Columbia's first massively open online course (MOOC), Natural Language Processing on Coursera, with 30,000 registered students.
- **Tutorials.** I have given well-attended tutorials at major conferences for NLP and Machine Learning on the topic of Lagrangian Relaxation for NLP, the subject of my Ph.D. research.
- **Direct Mentorship.** During graduate school I have worked directly with a half-dozen students – at undergraduate, Masters, and Ph.D. level – as a research mentor.

These varied experiences have given me a holistic view of Computer Science education. It is a discipline that cannot fully be conveyed in a classroom lecture, but one that is supported by practical experience, feedback to questions, and when possible, direct mentorship.

Lecturing and Course Development

My most direct classroom experience has been as a lecturer for a graduate-level course on Natural Language Processing, which I co-taught at Columbia University in the fall of 2013. The course had over eighty enrolled students with backgrounds ranging from undergraduate to Ph.D. student. I taught the lectures in the second half of the class, which consisted of bi-weekly 1.5 hour classes covering topics at the intersection of Machine Learning and NLP. This included lectures on log-linear models, conditional random fields, the structured perceptron, the expectation-maximization algorithm, and semi-supervised learning. These are some of the more mathematically rigorous topics in NLP and so these lectures alternated between slides describing the material and chalkboard work formalizing the models.

In addition to lecturing, I also wrote a new set of programming assignments for the class. These consisted of four independent assignments covering part-of-speech tagging, syntactic parsing, translation alignment, and structured learning. The assignments aimed to provide a similar setup to what a student would encounter in practice. They provided detailed descriptions of the problem domain, sample code, and evaluation scripts, but required the student to develop the code necessary to process input and make predictions. These assignments were used in subsequent versions of the class as well as the online version of the class.

Outside of this class, I have taught several times as a guest lecturer in NLP and Machine Learning at NYU. I have lectured on the topics of context-free grammars, dependency parsing, and dual decomposition for graphical models. I also have taught as a guest lecturer at Stanford on the topic of the Haskell programming language.

In previous years, I was a teaching assistant for Natural Language Processing, as well as a TA for classes in AI, Machine Learning, and Intro to Computer Science. For each of these classes, I graded, wrote assignments, and gave weekly recitation lectures.

MOOC

One of the most challenging and rewarding experiences of graduate school was acting as the sole teaching assistant for Columbia's first massively open online course (MOOC), Natural Language Processing on Coursera. The course attracted over 30,000 students of which over a thousand completed it with a certificate. My role as TA included writing assignments, producing in-lecture questions, answering questions on the forum, and writing an automatic grading system.

While the online environment is clearly different from a classroom, the scope of the course helped shape my understanding of how students learn difficult material. Through daily interaction in the forum, we could observe how different students learned concepts in strikingly different ways. This motivated a class with a multifaceted approach to teaching: utilizing lectures, lecture notes, formative in-lecture questions, problem sets, programming assignments, and detailed forum responses. Despite the limitations of the online environment, many of the students who finished the class had as strong a grasp of the material as students in the classroom.

Tutorials

Another exciting teaching experience was presenting tutorials at conferences. I gave tutorials on Lagrangian relaxation algorithms at two academic conferences: one for NLP (ACL) and one for Machine Learning (NIPS); as well as at invited seminars at Google Research in New York and Mountain View. These tutorials consisted of a three-hour lecture with detailed slides. Additionally I have published the tutorial as a journal article.

While similar to an in-class lecture, the tutorial presentation required teaching to an audience of focused researchers. The aim was to be pedagogical, but still offer useful practical information about NLP and the methods used. The experience taught me how to present material at a deeper level while still being comprehensive enough for those without extensive background.

Direct Mentorship

In my final years as a graduate student, I acted as a mentor for several students working on research projects. These included two undergraduates, three Master's, and an early-stage Ph.D. student. Each week I met individually with each student for at least an hour to discuss ongoing research, edit drafts, and do code reviews. Several of the students have gone on to Ph.D. programs or to tech companies.

One consequence of mentoring was a realization that the programming tools available to an inexperienced student in NLP were severely lacking. This motivated me to write a toolkit, *PyDecode*, that aims to make it easy for students to prototype complex algorithms and utilize optimized libraries for their research. The toolkit is now widely used among the group and is available publicly at `http://www.pydecode.org`. Several papers utilizing the toolkit are currently in submission.

These experiences teaching have given me a broad perspective on the different challenges of Computer Science education. For a student learning the subject, it is critically important to both formally understand critical concepts and feel comfortable implementing them in practice. As a teacher, you have a wide set of tools to help them reach this goal, including lecturing, well-developed assignments, direct feedback, notes, and even software.

Given my background in computer science and practical experience as a software engineer, I would be excited to teach most undergraduate computer science classes. I also would be comfortable teaching graduate classes in Natural Language Processing, Machine Learning, or Artificial Intelligence.