David Bamman Teaching Statement

My teaching philosophy is grounded in the increasingly vital role that cross-disciplinary learning has in the life of computer science. While areas like NLP and machine learning have now become mainstays in computer science departments, it's easy to forget that they are the shared ground of several disciplines (CS and linguistics for the former; CS and statistics for the latter); as more and more of these hybrid communities of practice continue to rise – including computational social science, computational journalism and the digital humanities – it's important that we give space to these shared endeavors so the next institutional mainstay has room to take root and thrive.

The courses I've taught at Carnegie Mellon have given me an opportunity to put this philosophy into practice. In Spring 2014, I received the **Alan J. Perlis SCS Student Teaching Award** in recognition of my service; this award is granted to one student each year from the School of Computer Science at Carnegie Mellon "who has shown the highest degree of excellence and dedication" in teaching. This award recognized my service in designing and co-teaching a new interdisciplinary course ("Digital Literary and Cultural Studies") and in serving as TA for the core CS course of "Natural Language Processing."

1 | DIGITAL LITERARY AND CULTURAL STUDIES

In Fall 2013, I designed and co-taught a new interdisciplinary course with Christopher Warren, a professor of English, entitled "Digital Literary and Cultural Studies." This course blended a survey of computational methods and algorithms (including social network analysis, probabilistic graphical models and supervised classification) with in-depth application to the specific domain of Renaissance and Early Modern Studies (ca. 1500–1700).

"Digital Literary and Cultural Studies" brought together students from both Computer Science and the English department to create an environment where each disciplinary side can learn from the other – not just transferring the algorithmic knowledge we have in computer science to another domain, but in working in the other direction as well, leveraging the critical acumen that's fostered in the humanities to reflect back on the assumptions that underlie many of our computational models. In practice, we achieved this in two ways. At the daily level, we split each class lecture in two, with the first half emphasizing a thorough understanding of the core algorithms introduced that day, and the second half discussing disciplinary readings to unpack the assumptions implicit in those models, from the perspective of a humanist. In both halves, the students were engaged participants in the discussion, working through the algorithms and their implications as we walked through them.

At the semester level, the course also involved a collaborative final project, in which all students were encouraged to form cross-disciplinary teams (including at least one person with computational experience and another with domain expertise). Projects that originated out of this collaborative work investigated the changing discourse on electricity from text analysis of Ben Franklin's letters, a comparative computational study of the manuscript traditions of *Piers Plowman*, and an analysis of the changing ingredient networks extracted from a collection of early American cookbooks.

One of my goals in designing this course was to assemble students from a variety of disciplines and teach not only the algorithms themselves, but also a general shared vocabulary for computational collaboration, so that when students sought a collaborator across disciplines (whether CS or English) in the future, they would be empowered being able to have a meaningful, and critical, conversation. This goal informs my broader teaching philosophy: what we teach in such courses is not simply a narrow set of methods, but the groundwork for a potential lifetime of critical engagement with ideas, a skill useful for students on a variety of career paths – not only for computer science researchers but also for the next generation of successful entrepreneurs.

2 | NATURAL LANGUAGE PROCESSING

Within core computer science, I've also held the role of teaching assistant for CMU's "Natural Language Processing" course (jointly taught by Noah Smith and Chris Dyer). NLP is a popular course focusing on the fundamental problems and algorithms in natural language processing, emphasizing dynamic programming; topics include morphological parsing (finite-state automata and transducers), part-of-speech tagging (hidden Markov models), language model smoothing, spelling correction (edit distance), syntactic parsing (CKY), machine translation, and text categorization (Naive Bayes, perceptron); my own lecture in this course covered the topics of word sense disambiguation and coreference resolution. Students work in teams on a competitive final project on question answering; each team develops a system leveraging the techniques learned in class to both generate questions from Wikipedia articles and also answer questions that other teams have generated.

As a teaching assistant in this course, I had the opportunity to cultivate what I consider one of my strengths– explaining complex algorithms in a way that students can not only grasp them, but also more deeply understand their assumptions and limitations, and know when and how to apply them in the real world.

3 | TEACHING INTERESTS

I enjoy teaching; the courses I envision including as part of my curriculum over the next few years span a range of core computer science areas (such as machine learning and natural language processing) and interdisciplinary courses that will build ties across campus with students and faculty in the social sciences and humanities.

Natural Language Processing. NLP provides the foundation for a wide range of text analysis in many domains, enabling students to get beyond simple bag-of-words representations for text and into the more complex structures that are part of language (such as who does what to whom). I envision teaching both introductory courses (for undergraduates) and advanced seminars covering state-of-the-art research in this wide field. Potential textbooks include Jurafsky and Martin, *Speech and Language Processing* (3rd edition forthcoming).

Machine Learning. Machine learning is a necessary course in all contemporary computer science departments; the classification and clustering techniques that comprise its pillars support the kind of large-scale data analysis that is now ubiquitous in both science and industry. As with NLP, I can envision teaching both introductory courses and advanced seminars in this topic. Given the connection to data science, I can also envision teaching a professional master's level course targeted to industry. Potential textbooks include Kevin Murphy, *Machine Learning: A Probabilistic Perspective* (2012).

Computational Social Science. Much of the social sciences already have heavily quantitative aspects, involving empirical measurements and statistical hypothesis tests. As more and more quantitative social data becomes available to us (from online social networks, web data and other sources), we increasingly need computational methods to reason about it at scale. I can envision teaching a course about this burgeoning space, including methods for large-scale data analysis that address real social scientific questions of interest. While I am happy to teach this course on my own, I am also enthusiastic about the possibility of developing a joint course with faculty from a social science discipline.

Computational Humanities. Likewise, the humanities have their own history of using quantitative data that dates at least to 19th-century German philology in the study of language. With the rise of large-scale digitization projects such as Google Books and HathiTrust, the data available for this analysis has exploded; we now have large swaths of the cultural production of humanity available to us in digital form. Many of the text analysis techniques that are useful in this domain overlap with those exploited in computational social science; I envision teaching a separate course directed at the humanities to explore the problems and questions that are unique to this discipline (such as those involving literary theory). As with computational social science, I am happy to teach this course on my own, or to co-develop it with a colleague from a humanities department.

4 | MENTORING

I've also had the opportunity to mentor three undergraduates in research projects, one focusing on improving named entity recognition in books, and two working together to learn relation types (such as *mother*, *brother*, *friend*) between people in several domains. My approach to mentoring takes a long-term view of the student's involvement in computer science research. When working together, I not only target a specific project deliverable (such as a conference deadline), but I also work with them to develop a more well-rounded practicum, stressing the understanding and implementation of a wide range of algorithms while also encouraging them to push further where their interests lie.