# Teaching NLP to Computer Science Majors via Applications and Experiments

**Reva Freedman**
Department of Computer Science
Northern Illinois University
DeKalb, IL 60115
rfreedman@niu.edu

## Abstract

Most computer science majors at Northern Illinois University, whether at the B.S. or M.S. level, are professionally oriented. However, some of the best students are willing to try something completely different. NLP is a challenge for them because most have no background in linguistics or artificial intelligence, have little experience in reading traditional academic prose, and are unused to open-ended assignments with gray areas. In this paper I describe a syllabus for Introduction to NLP that concentrates on applications and motivates concepts through student experiments. Core materials include an introductory linguistics textbook, the Jurafsky and Martin textbook, the NLTK book, and a Python textbook.

## 1 Introduction

Northern Illinois University is a large public university (25,000 students) located about 60 miles west of Chicago. Most computer science majors come from the suburbs and exurbs of Chicago or small towns near the university. Their preferred career path is generally to obtain a programming job in local industry, preferably in a hi-tech area. Most students take the Introduction to NLP course out of a desire to do something different from their required courses.

In this paper I describe the issues I have found in teaching NLP to this population, and the syllabus I have developed as a result. Since the students enjoy programming and see system development as the core issue of computer science, I concentrate on applications and their structure. I motivate many of the issues involved using data and systems from the web and in-class experiments. I explicitly teach the linguistics background that they need.

## 2 Student background

I started from the following assumptions derived from several years of teaching Introduction to Artificial Intelligence and Introduction to NLP at NIU.

Linguistic background:

1. Students have never studied linguistics.
2. Students are not familiar with the common syntactic constructions of English taught in traditional English grammar, and are often unsure about parts of speech.
3. Students have little experience with languages other than English.

Programming:

4. Students are not familiar with programming languages other than conventional imperative languages such as C++, Java, and .NET.
5. Students like to program and to build working systems.
6. Students expect to have programming languages explicitly taught in class.

Academic approach:

7. Students live on the web and are uncomfortable having to use offline reference materials.
8. Students are not comfortable with or interested in traditional academic prose or research papers.
9. Students are taking the course for fun and to do something different. They are unlikely to need specific NLP content in their future careers.



*Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics (TeachCL-08)*, pages 114–119,
Columbus, Ohio, USA, June 2008. 

10. Students taking NLP are unlikely to have time in their program to take another artificial intelligence course (although there are exceptions).

## 3 Course goals

From these presuppositions I have developed the following general principles to provide a positive experience for both students and teacher:

1. Teach the linguistic content explicitly, at a level suitable for beginners.
2. Concentrate on applications, using them to motivate algorithms.
3. Concentrate on student involvement at all levels: in-class experiments, take-home experiments to be discussed in class, and practical programming projects.
4. Concentrate on a few basic principles that are repeated in many contexts, such as rule-based vs. Bayesian approaches and the role of world knowledge in working systems.

From these presuppositions I have developed a syllabus that maintains student interest, provides students a basic background in NLP, and also provides them with useful skills and knowledge that they may not otherwise encounter in their program of study.

The course has three goals:

1. Give students a general background in the issues involved in handling both speech and written text, some of the most common applications, and some of the most widely used algorithms.
2. Provide students with a productive experience in a modern programming language.
3. Teach students a number of useful concepts that they might not otherwise come across in their course of study. These topics include:

- Bayes' Law
- Dynamic programming
- Hidden Markov models
- Regular expressions and finite-state machines
- Context-free grammars

The following sections of the paper describe the most important units of the course, showing how they use the principles stated above to contribute to these goals.

## 4 Introducing NLP

The first goal of the course is to define the NLP task and explain why it is harder and less determinate than many of the problems they have studied in their other courses.

I start by encouraging students to list all the meanings they can for "I made her duck", based on the five meanings given by Jurafsky and Martin (2000, section 1.2). For a view of a system that can deal with such issues, I then introduce Figure 1.1 of Bird, Klein, and Loper (2008, henceforce referred to as the NLTK textbook), which shows a pipeline architecture for a spoken dialogue system. I use this opportunity to discuss each component and possible data representations.

## 5 Providing linguistic background

I introduce three kinds of background knowledge, related to speech, words and sentences, and human factors issues.

### 5.1 Background for speech processing

To provide essential background for discussing speech processing, I introduce the concepts of *phone* and *phoneme*. I also teach give a brief introduction to the IPA so that I can use it in examples. I use the following sections from Stewart and Vaillette (2001), a textbook for introductory linguistics classes:

File 3.1: International Phonetic Alphabet (IPA)
File 3.2: English consonants
File 3.3: English vowels
File 3.5: English transcription exercises
File 4.1: Phones vs. phonemes

These sections were chosen to provide the background students need while providing maximum opportunities for interaction. Students have found this approach more accessible than the rather terse treatment in Jurafsky and Martin (2000, ch. 4). I do the following activities, familiar to teachers of introductory linguistics classes, in class:

• Putting one's fingers on the glottis to experience the difference between voiced and unvoiced

consonants
• Putting one's hand in front of one's mouth to experience the difference between aspirated and unaspirated consonants
• Reading IPA transcription in pairs

I also introduce students to the idea that both pronunciation and other areas of human language generation are affected by context. For example, using Figure 5.7 of Jurafsky and Martin (2000) as a guide, I try to generate as many as possible of the sixteen most common pronunciations of *because* shown in that figure.

## 5.2 Background for text processing

As background for the text processing section, I lecture on a few core aspects of syntax and related topics that will be needed during the semester. These topics include the following:

- What is a word?
- How many parts of speech are there?
- Lexical ambiguity
- Syntactic ambiguity, including PP attachment, attachment of gerunds, and coordination ambiguity
- Difference between syntactic structure and intention

## 5.3 Background in human factors issues

This section includes several topics that experience has shown will be needed during the semester.

The first is the difference between descriptive and prescriptive linguistics. I take class polls on various sociolinguistic issues, including pronunciation, word choice and sentence structure, using File 10.10: Language variation from Stewart and Vaillette (2001) as a basis.

I take a poll on the pronunciation of the word *office*, choosing that word since the distribution of its first vowel is sensitive both to geography and speaker age. The poll gives me an opportunity to introduce some of the human factors issues related to corpus collection and the issue of statistical significance. We also examine some data collection tasks found on the Internet, using them to discuss experimental design and how it relates to the data collected.

Finally, I begin a discussion on the difference between rule-based and statistical systems that will recur frequently during the semester. This is a good place to discuss the importance of separating training data and test data.

# 6 Python

## 6.1 Basic Python

The next step is to teach basic Python so that there will be time for some practice programs before the first major programming project. As computer science majors, the students tend to find that the treatment in the NLTK textbook does not answer enough of their technical questions, such as issues on argument handling and copying of objects vs. references to them.

I give several lectures on Python, including the following topics:

- Basic data structures
- Basic control structures
- Functions and modules
- Objects
- File handling

I have found Lutz (2008) to be the most readable introductory textbook. I use Chun (2007) as a reference for topics not covered by Lutz, such as regular expressions and some of the I/O options.

## 6.2 Using Python for basic language handling

This unit basically covers the material in chapters 2, 3, and 6 of the NLTK textbook. The goal is to show students how easily some of these problems can be handled with an appropriate programming language. Many of them are quite uncomfortable with the idea of a list not implemented with pointers, but in the end they cope well with a language that does not have all the baggage of C++.

I give a simple assignment that involves finding the most common words in a corpus. A secondary purpose of this assignment is to reinforce the earlier lecture on the difficulty of defining a word. I lard the input text for the assignment with problematic cases such as hyphenated multiword expressions, e.g., "the orange-juice based confection."

## 7 Rule-based dialogue systems using regular expressions

Since later in the course we will be comparing rule-based systems to statistics-based systems, this is an appropriate time to introduce rule based systems. We experiment in class with Eliza, trying both to make it work and make it fail. I give out a list of versions available on the web, and students can easily find more. In class I often use the emacs built-in version.

I then give out copies of the original Eliza paper (Weizenbaum, 1966), which contains the original script in an appendix. If time permits, I also discuss PARRY (Parkison, Colby and Faught, 1977), which has a much more linguistically sophisticated design but there is no simulator available for it.

I introduce regular expressions at this point for two reasons. In addition to being required for continued use of the NLTK textbook, regular expressions are an important idea that is not otherwise included in our curriculum. We experiment with Rocky Ross' interactive web site (Pascoe, 2005) and occasionally with other simulators. I also assign a simple homework using regular expressions in Python.

The first major project in the course is to write an shallow interactive written dialogue system, i.e., an Eliza-type program. Students have the choice of choosing a more realistic, limited domain, such as a database front-end, or of picking a specific case (e.g., a linguistic issue) that they would like Eliza to handle. This project is implemented in Python as a rule-based system with heavy use of regular expressions. Before they write their code, students do a five-minute presentation of their domain, including a sample conversation. After the projects are due, they present their results to the class.

## 8 Spelling correction and Bayes' Law

Bayes' Law is another core topic that students are generally unfamiliar with, even though statistics is required in our program. To provide a contrast to rule-based systems, and to introduce this core topic, I present Kernighan, Church and Gale's (1990) Bayesian approach to spelling correction, as explained by Jurafsky and Martin (2000, section 5.5).

Kernighan et al. choose as the preferred correction the one that maximizes $P(t|c)P(c)$, where $t$ is the typo and $c$ is a candidate correction. In a previous paper (Freedman, 2005), I discuss in detail an assignment where students choose a corpus and replicate Kernighan's calculations. They then compare their results to results from their favorite word processor.

Students are generally surprised at how similar the results are from what they originally see as an unmotivated calculation. They are always surprised to learn that spelling correction is generally not done by a lookup process. They are also surprised to learn that learn that results were largely independent of the corpus chosen.

I also demonstrate approximating word frequencies by page counts in Google, along with a discussion of the advantages and disadvantages of doing so. In general, students prefer to use one of the NLTK corpora or a corpus obtained from the web.

## 9 Machine translation: rule-based and statistical models

This unit has several purposes. In addition to showing students how the same problem can be attacked in remarkably different ways, including multiple levels of rule-based and statistically-based systems, machine translation gives students a look at a fielded application that is good enough to be viable but sill obviously needs improvement.

To the extent that information is publicly available, I discuss the architecture of one of the oldest machine translation systems, Systran (Babelfish), and one of the newest, Microsoft Live Translator. The latter uses components from MindNet, Microsoft's knowledge representation project, which provides another opportunity to reinforce the importance of world knowledge in artificial intelligence and NLP in particular. It also provides an initial opportunity to discuss the concept of machine learning as opposed to hand-crafting rules or databases.

As the assignment for this unit, students choose a short text in a foreign language. They use multiple web-based translation systems to translate it into English, and analyze the results. In addition to the systems mentioned above, the Reverso system has done well in these experiments.

Popular inputs include administrative text (e.g., citizenship rules) from a bilingual country and

chapter 1 of Genesis. One student started with a French version of the Tolkien poem "... one ring to rule them all..." Although translation of poetry obviously poses different issues than technical text, a fruitful discussion emerged from the fact that two of the systems misparsed one or more of the lines of the poem.

## 10 POS identification, parsing and author identification

This unit of the course covers key sections of chapters 4, 7, 8 and 9 of the NLTK textbook. Although one student originally stated that "I really don't care about parts of speech," students find this material more interesting after seeing how many of the machine translation errors are caused by parsing errors. Still, I only cover POS assignment enough to use it for chunking and parsing.

The application chosen for this unit involves author identification. I introduce students to the basics of the Federalist Papers controversy. Then I discuss the approach of Mosteller and Wallace (1984), which depends largely on words used much more frequently by one author than the other, such as *while* and *whilst*.

I suggest to students that more interesting results could perhaps be obtained if data about items such as part of speech use and use of specific constructions of English were added to the input. As an alternative assignment, I give students transcripts of tutoring by two different professors and invite them to identify the authors of additional transcripts from a test set. A secondary goal of this assignment is for students to see the level of cleanup that live data can require.

This assignment also shows students the relative difficulty level of chunking vs. parsing better than any lecture could. This is useful because students otherwise tend to find chunking too ad hoc for their taste.

I do teach several approaches to parsing since many students will not otherwise see context-free grammars in their studies. Having had the experiences with machine translation systems helps prevent the reaction of a previous class to Earley's algorithm: "we understand it; it's just not interesting." I also frame Earley's algorithm as another example of dynamic programming.

## 11 Speech understanding

Students generally find speech a much more compelling application than written text. In this unit I discuss how basic speech processing works. This unit provides a nice review of the basics of phonology taught at the beginning of the semester. It also provides a nice review of Bayes' Law because the approach used, based on Jurafsky and Martin (2000, ch. 5.7–5.9) uses Bayes' Law in a fashion similar to spelling correction.

The assignment for this unit involves experimenting with publicly available speech understanding systems to see how well they work. The assignment involves comparing two automated 411 systems, Google's new system (1-800-GOOG411), which was built specifically for data collection, and Jingle (1-800-FREE411), which is advertising-supported. I also encourage students to report on their own experiments with bank, airline, and other systems.

I give at least one anonymous questionnaire every semester. Students generally report that the level of detail is appropriate. They generally vote for more topics as opposed to more depth, and they always vote for more programming assignments and real systems rather than theory.

## 12 Future work

I am considering replacing author identification by question answering, both because it is an important and practical topic and because I think it would provide better motivation for teaching chunking. I am also considering keeping author identification and adding the use of a machine learning package to that unit, since I believe that machine learning is rapidly becoming a concept that all students should be exposed to before they graduate.

My long-term goal is to have students build an end-to-end system. A short-term goal in service of this objective would be to add a unit on text-to-speech systems.

## 13 Conclusions

This paper described a syllabus for teaching NLP to computer science majors with no background in the topic. Students enjoyed the course more and were more apt to participate when the course was oriented toward applications such as dialogue

systems, machine translation, spelling correction and author identification. Students also learned about the architecture of these systems and the algorithms underlying them. Students implemented versions of some of the smaller applications and experimented with web versions of large fielded systems such as machine translation systems.

## Acknowledgments

I thank the authors of Jurafsky and Martin (2000) and Bird, Klein and Loper (2008), whose extensive labor has made it possible to teach this course. I would also like to thank the anonymous reviewers for their suggestions.

## References

Steven Bird, Ewan Klein, and Edward Loper. (2008). *Natural Language Processing in Python*. Available on the web at http://nltk.org/index.php/Book.

Wesley J. Chun. (2007). *Core Python Programming*, 2/e. Upper Saddle River, NJ: Prentice-Hall.

Reva Freedman. (2005). Concrete Assignments for Teaching NLP in an M.S. Program. In Second Workshop on Effective Tools and Methodologies for Teaching NLP and CL, 43rd Annual Meeting of the ACL.

Daniel Jurafsky and James H. Martin. (2000). *Speech and Language Processing*. Upper Saddle River, NJ: Prentice-Hall.

Mark Lutz. (2008). *Learning Python*, 3/e. Sebastopol, CA: O'Reilly.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. (1990). A spelling correction program based on a noisy channel model. In COLING '90 (Helsinki), v. 2, pp. 205–211.

Frederick and Mosteller and David L. Wallace. (1984). *Applied Bayesian and Classical Inference: The Case of* The Federalist *Papers.* New York: Springer. Originally published in 1964 as *Inference and Disputed Authorship: The Federalist.*

Brad Pascoe (2005). Webworks FSA applet. Available at http://www.cs.montana.edu/webworks/projects/theoryportal/models/fsa-exercise/appletCode/fsa_applet.html.

Roger C. Parkison, Kenneth Mark Colby, and William S. Faught. (1977). Conversational Language Comprehension Using Integrated Pattern-Matching and Parsing. *Artificial Intelligence* 9: 111–134.

Thomas W. Stewart, Jr. and Nathan Vaillette. (2001). *Language Files: Materials for an Introduction to Language and Linguistics*, 8/e. Columbus: Ohio State University Press.

Joseph Weizenbaum. (1966). Eliza—A Computer Program for the Study of Natural Language Computation between Man and Machine. Communications of the ACM 9(1): 36–45.

NLP (Neuro Linguistic Programming) has been around in language teaching longer than we may realise. Those teachers who incorporate elements of suggestopedia, community language learning, music, drama and body language into their lessons are already drawing on NLP as it stood twenty years ago. The roots of NLP. NLP and language learning. NLP in the classroom.Â  NLP and related subjects have their sceptics, particularly in terms of general classroom applicability and how NLP is commercially marketed as a method of self-improvement. NLP has been labelled a 'quasi science' and criticised on the grounds of lack of empirical studies, but there are sound reasons why NLP is compatible with current classroom practice. NLP is about recognising patterns. Specific course work in computer programming and programming languages, including recursion, linked structures, searching and sorting techniques, stacking, and queues will be accompanied by other courses in computer science and discrete mathematics. In addition, students will study electronic commerce to understand how to construct a multifunctional website using HTML forms.Â  Computer Science. This is the study of the theoretical foundation for the development of computers and their applications. Computer science considers the mathematical base for computers, as well as flow charting, diagramming, programming systems analysis, systems interface, software development, and related fields. Like many others, our Computer Science program had academic requirements and a competitive application process. This automatically weeds out the weaker students and attracts those who are naturally stronger at math and reasoning. You spend alot of time hacking on late-night homework and projects at the labs.Â  The Computer Science major is usually not ABET accredited unlike the Computer Engineering major in most colleges in the United States.