ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, 2002

A non-programming introduction to computer science via NLP, IR, and AI

Lillian Lee
Department of Computer Science
Cornell University
Ithaca, NY, USA, 14853-7501
llee@cs.cornell.edu

Abstract

This paper describes a new Cornell University course serving as a non-programming introduction to computer science, with natural language processing and information retrieval forming a crucial part of the syllabus. Material was drawn from a wide variety of topics (such as theories of discourse structure and random graph models of the World Wide Web) and presented at some technical depth, but was massaged to make it suitable for a freshman-level course. Student feedback from the first running of the class was overall quite positive, and a grant from the GE Fund has been awarded to further support the course's development and goals.

1 Introduction

Algorithmic concepts and programming techniques from computer science are very useful to researchers in natural language processing. To ensure the continued strength of our field, then, it is important to encourage undergraduates interested in NLP to take courses conveying computer science content. This is especially true for students not intending to become computer science majors.

Usually, one points beginning students interested in NLP towards the first programming course (henceforth "CS101"). However, at many institutions, CS101 is mandatory for a large portion of the undergraduates (e.g., all engineering students) and is designed primarily to transmit specific programming skills. Experience suggests that a significant fraction of students find CS101's emphasis on skills rather than concepts unstimulating, and therefore decide not to take further computer science courses. Unfortunately, down the road this results in fewer entering NLP graduate students having been educated in important advanced computer-science concepts. Furthermore, fewer students are introduced to NLP at all, since the subject is often presented only in upper-level computer-science classes.

In an attempt to combat these problems, I created a new freshman-level course, *Computation, Information, and Intelligence*[1], designed to introduce entering undergraduates to some of the ideas and goals of AI (and hence computer science). The premise was that if freshmen first learned something of what artificial intelligence is about, what the technical issues are, what has been accomplished, and what remains to be done, then they would be much more motivated when taking CS101, because they would understand what they are learning programming *for*.

Three major design decisions were made at the outset:

- *No programming*: Teaching elementary programming would be a needless reduplication of effort, since programming pedagogy is already well-honed in CS101 and other such classes. Moreover, it was desirable to attract students having little or no programming experience: the new course would offer them an opportunity for

[1] `http://www.cs.cornell.edu/courses/cs172/2001fa`

initial exploration at a conceptual level. Indeed, for the first edition of the class, students *with* programming experience were actively *discouraged* from enrolling, in order to ensure a more level playing field for those without such background.[2]

• *Emphasis on technically challenging material*: Although no programming would be involved, the course would nevertheless bring students face-to-face with substantial technical material requiring mathematical and abstract reasoning (indeed, topics from graduate courses in NLP and machine learning were incorporated). To achieve this aim, the main coursework would involve challenging pencil-and-paper problems with significant mathematical content.[3]

Of course, one had to be mindful that the intended audience was college freshmen, and thus one could only assume basic calculus as a prerequisite. Even working within this constraint, though, it was possible to craft problem sets and exams in which students explored concepts in some depth; the typical homework problem asked them not just to demonstrate comprehension of lecture material but to investigate alternative proposals. Sample questions are included in the appendix.

• *Substantial NLP and IR content*:[4] Because many students have a lot of experience with search engines, and, of course, all students have a great deal of experience with language, NLP and IR are topics that freshmen can easily relate to without being introduced to a lot of background first.

[2]Students who have programmed previously are more likely to happily enroll in further computer science courses, and thus are already well-served by the standard curriculum.

[3]An alternative to a technically- and mathematically-oriented course would have been a "computers and the humanities" class, but Cornell already offers classes on the history of computing, the philosophy of AI, and the social implications of living in an information society. One of the goals for *Computation, Information, and Intelligence* was that students learn what "doing AI" is really like.

[4]In this class, I treated information retrieval as a special type of NLP for simplicity's sake.

2 Course content

The course title, *Computation, Information, and Intelligence*, reflects its organization, which was inspired by Herb Simon's (1977) statement that "Knowledge without appropriate procedures for its use is dumb, and procedure without suitable knowledge is blind." More specifically, the first 15 lectures were mainly concerned with algorithms and computation (game-tree search, perceptron learning, nearest-neighbor learning, Turing machines, and the halting problem). For the purposes of this workshop, though, this paper focuses on the remaining 19 lectures, which were devoted to information, and in particular, to IR and NLP. As mentioned above, sample homework problems for each of the units listed below can be found in the appendix.

We now outline the major topics of the last 22 lectures. Observe that IR was presented before NLP, because the former was treated as a special, simpler case of the latter; that is, we first treated documents as bags of words before considering relations between words.

Document retrieval [3 lectures]. Students were first introduced to the Boolean query retrieval model, and hence to the concepts of index data structures (arrays and B-trees) and binary search. We then moved on to the vector space model[5], and considered simple term-weighting schemes like tf-idf.

The Web [4 lectures]. After noting how Vannevar Bush's (1945) famous "Memex" article anticipated the development of the Web, we studied the Web's global topology, briefly considering the implications of its so-called "bow-tie" structure (Broder et al., 2000) for web crawlers — students were thus introduced to graph-theoretic notions of strongly-connected components and node degrees. Then, we investigated Kleinberg's (1998) hubs and authorities algorithm as an alternative to mere in-link counting:

[5]This does require some linear algebra background in that one needs to compute inner products, but this was covered in the section of the course on perceptrons. Since trigonometry is actually relatively fresh in the minds of first-year students, their geometric intuitions tended to serve them fairly well.

fortunately, the method is simple enough that students could engage in hand simulations. Finally, we looked at the suitability of various random graph generation models (e.g., the "rich-get-richer" (Barabási et al., 1999) and "copying" models (Kumar et al., 2000)) for capturing the local structure of the Web, such as the phenomenon of in-degree distributions following a power law — conveniently, these concepts could be presented in such a way that only required the students to have intuitive notions of probability and the ability to take derivatives.

Language structure [7 lectures]. Relying on students' instincts about and experience with language, we considered evidence for the existence of hidden language structure; such clues included possible and impossible syntactic and discourse ambiguities, and movement, prosody and pause cues for constituents. To describe this structure, we formally defined context-free grammars. We then showed how (a tiny fragment of) X-bar theory can be modeled by a context-free grammar and, using its structural assignments and the notion of heads of constituents, accounted for some of the ambiguities and non-ambiguities in the linguistic examples we previously examined.

The discussion of context-free grammars naturally led us to pushdown automata (which provided a nice contrast to the Turing machines we studied earlier in the course). And, having thus introduced stacks, we then investigated the Grosz and Sidner (1986) stack-based theory of discourse structure, showing that language structures exist at granularities beyond the sentence level.

Statistical language processing [6 lectures] We began this unit by considering word frequency distributions, and in particular, Zipf's law — note that our having studied power-law distributions in the Web unit greatly facilitated this discussion. In fact, because we had previously investigated generative models for the Web, it was natural to consider Miller's (1957) "monkeys" model which demonstrates that very simple generative models can account for Zipf's law. Next, we looked at methods taking advantage of statistical regularities, including the IBM Candide statistical machine translation system, following Knight's (1999) tutorial and treating probabilities as weights. It was interesting to point out parallels with the hubs and authorities algorithm — both are iterative update procedures with auxiliary information (alignments in one case, hubs in the other). We also discussed an intuitive algorithm for Japanese segmentation drawn from one of my own recent research collaborations (Ando and Lee, 2000), and how word statistics were applied to determining the authorship of the Federalist Papers (Mosteller and Wallace, 1984). We concluded with an examination of human statistical learning, focusing on recent evidence indicating that human infants can use statistics when learning to segment continuous speech into words (Saffran et al., 1996).

The Turing test [2 lectures] Finally, we ended the course with a consideration of intelligence in the large. In particular, we focused on Turing's (1950) proposal of the "imitation game", which can be interpreted as one of the first appearances of the claim that natural language processing is "AI-complete", and Searle's (1980) "Chinese Room" rebuttal that fluent language behavior is *not* a sufficient indication of intelligence. Then, we concluded with an examination of the first running of the Restricted Turing Test (Shieber, 1994), which served as an object lesson as to the importance of careful evaluation in NLP, or indeed any science.

3 Experience

Twenty-three students enrolled, with only one-third initially expressing interest in majoring in computer science. By the end, I was approached by four students asking if there were research opportunities available in the topics we had covered; interestingly, one of these students had originally intended to major in electrical engineering. Furthermore, because of the class's promise in drawing students into further computer science study, the GE Fund awarded a grant for the purpose of bringing in a senior outside speaker and supporting teaching assistants in future years.

One issue that remains to be resolved is the lack, to my knowledge, of a textbook or textbooks that would both cover the syllabus topics and employ a level of presentation suitable for freshmen. For first-year students to learn effectively, some sort of reference seems crucial, but a significant portion of the course material was drawn from research papers that would probably be too difficult. In the next edition of the course, I plan to write up and distributeformal lecture notes.

Overall, although *Computation, Information, and Intelligence* proved quite challenging for the students, for the most part they felt that they had learned a lot from the experience, and based on this evidence and the points outlined in the previous paragraph, I believe that the course did make definite progress towards its goal of interesting students in taking further computer science courses, especially in AI, IR, and NLP.

Acknowledgments

I thank my chair Charles Van Loan for encouraging me to develop the course described in this paper, for discussing many aspects of the class with me, and for contacting the GE Fund, which I thank for supplying a grant supporting the future development of the class. Thanks to Jon Kleinberg for many helpful discussions, especially regarding curriculum content, and to the anonymous reviewers for their feedback. Finally, I am very grateful to my teaching assistants, Amanda Holland-Minkley, Milo Polte, and Neeta Rattan, who helped immensely in making the first outing of the course run smoothly.

References

Rie Kubota Ando and Lillian Lee. 2000. Mostly-unsupervised statistical segmentation of Japanese. In *First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 241–248.

Albert-László Barabási, Réka Albert, and Hawoong Jeong. 1999. Mean-field theory for scale-free random networks. *Physica*, 272:173–187.

Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the web. In *Proceedings of the Ninth International World Wide Web Conference*, pages 309–430.

Vannevar Bush. 1945. As we may think. *The Atlantic Monthly*, 176(1):101–108.

Ralph Grishman. 1986. *Computational Linguistics: An Introduction*. Studies in Natural Language Processing. Cambridge.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Jon Kleinberg. 1998. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 668–677.

Kevin Knight. 1999. A statistical MT tutorial workbook. `http://www.isi.edu/natural-language/-mt/wkbk.rtf`, August.

Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopolan, D. Sivakumar, Andrew Tomkins, and Eli Upfal. 2000. Stochastic models for the web graph. In *Proceedings of the 41st IEEE Symposium on the Foundations of Computer Science*, pages 57–65.

George A. Miller. 1957. Some effects of intermittent silence. *American Journal of Psychology*, 70:311–313.

Frederick Mosteller and David L. Wallace. 1984. *Applied Bayesian and Classical Inference: The Case of the Federalist Papers*. Springer-Verlag.

Jenny R. Saffran, Richard N. Aslin, and Elissa L. Newport. 1996. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928, December.

John R. Searle. 1980. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–457.

Stuart M. Shieber. 1994. Lessons from a restricted Turing test. *Communications of the ACM*, 37(6):70–78.

Herb A. Simon. 1977. Artificial intelligence systems that understand. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, volume 2, pages 1059–1073.

Alan M. Turing. 1950. Computing machinery and intelligence. *Mind*, LIX:433–60.

Appendix: sample homework problems

IR unit

For simplicity, in this question, let the document vector entries be the term frequencies normalized by vector length.

Suppose someone proposes to you to integrate negative information by converting a query $q =$ "$x_1, x_2, \ldots, x_j, \neg y_1, \neg y_2, \ldots, \neg y_k$" to an m-dimensional query vector $\overrightarrow{q}$ as follows: the ith entry q_i of $\overrightarrow{q}$ is:

$$q_i = \begin{cases} 0, & w_i \text{ not in the query} \\ 1, & w_i \text{ is a positive query term} \\ -1, & w_i \text{ is a negative query term} \end{cases}$$

They claim the -1 entries in the query vector will prevent documents that contain negative query terms from being ranked highly.

Show that this claim is incorrect, as follows. Let the entire three-word vocabulary be $w_1 =$ alligator, $w_2 =$ bat, and $w_3 =$ cat, and let $q =$ "alligator, bat, $\neg$cat". Give two documents d_1 and d_2 such that

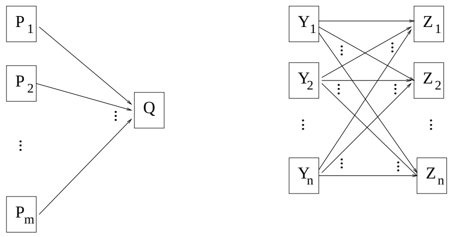
- d_1 and d_2 both contain exactly 8 words (obviously, some will be repetitions);
- d_1 does not contain the word "cat";
- d_2 *does* contain the word "cat"; and yet,
- d_2 is ranked more highly with respect to q than d_1 is.

Explain your answer; remember to show the document vectors corresponding to d_1 and d_2, the query vector, and how you computed them. Make sure the documents you choose and corresponding document vectors satisfy *all* the constraints of this problem, including how the documents get transformed into document vectors.

Web unit

In this question, we engage in some preliminary explorations as to how many "colluding" web pages might be needed to "spam" the hubs-and-authorities algorithm (henceforth HA).

Let m and n be two whole numbers bigger than 1 (m and n need not be equal, although they could be). Consider the following set of web pages (all presumably on the same topic):



That is, all of the m P_i pages point to Q, and all of the n Y_j pages point to all of the n Z_k pages.

(a) Let $m = 5$ and $n = 3$ (thus, $m = 2n - 1$, and in particular $m > n$), and suppose HA is run for two iterations. What are the best hub and authority pages? Explain your answers, showing your computations of the hub and authority scores of every web page (using the tabular format from class is fine).

(b) Now, let n be some whole number greater than 1, and let $m = n^2$. Suppose HA is run for two iterations in this situation. What are the best hub and authority pages? Explain your answers, showing your computations of the hub and authority scores of every web page. (Note: in this problem, you don't get to choose n; we're trying to see what happens in general if there are a quadratic number of colluding web pages. So treat n as an unknown but fixed constant.)

Language structure unit

Recall the Grishman (1986) "next train to Boston" dialog:

(1)	A:	Do you know when the next train to Boston leaves?
(2)	B:	Yes.
(3)	A:	I want to know when the train to Boston leaves.
(4)	B:	I understand.

(a) Using the Grosz/Sidner model, analyze the discourse structure of the entire conversation from the point of view of speaker A. That is, give the discourse segments (i.e., "DS1 consists of sentences 1 and 3, and DS2 consists of sentences 2 and 4"), the corresponding discourse segment purposes, and the intentional structure of the conversation. Then, show what the focus stack is after *each* sentence is uttered. Explain how you determined your answers.

(b) Repeat the above subproblem, but from the point of view of speaker B.

(c) Would your answers to the previous subproblem change if sentence (4) had been "Why are you telling me these things?" Does the Grosz/Sidner model adequately account for this case? Explain.

Statistical language processing unit

In this problem, we explicitly derive a type of power-law behavior in the "Miller's monkeys" model (henceforth MMM) from class. First, a useful fact: for any fixed integers $n > 1$ and $k > 0$,

$$\sum_{i=1}^{k} n^i = \frac{n^{k+1} - n}{n - 1}.$$

In each of the following subproblems, we rank the "words" (remember that "zz" is a "word" in the MMM) by their probability, rather than by corpus frequency. Also, j refers to some fixed but unknown integer greater than 1; hence, your answers should generally be functions of j.

(a) Show mathematically that the number of words that are shorter than j letters long is

$$\frac{26}{25}\left(26^{j-1} - 1\right).$$

(b) Compute the maximum possible rank for a word that is j letters long; explain your answer.

(c) Using your answers to the previous subproblems, find the function $AR(j)$, the *average* rank of a word that is j letters long, showing your work. (For example, you might say "$AR(j) = 4 - j$".)

(d) The probability of a word of length j is $P(j) = \frac{1}{27} \times \left(\frac{1}{27}\right)^j$ (that we aren't combining like terms is meant to be helpful ...). Show mathematically that the $AR(j)$ function you computed above and the probability function $P(j)$ have a particularly simple power-law relationship:

$$AR(j) \approx \alpha \times \frac{1}{P(j)}$$

for some constant α that doesn't depend on j. You may make some reasonable approximations, for example, saying that $\frac{n+1}{n+2}$ is close enough to 1 that we can replace it by 1 for argument's sake; but please make all such approximations explicit.