# Teaching the Basics of NLP and ML in an Introductory Course to Information Science

**Apoorv Agarwal**
Department of Computer Science
Columbia University
New York, USA
apoorv@cs.columbia.edu

## Abstract

In this paper we discuss our experience of teaching basic Natural Language Processing (NLP) and Machine Learning (ML) in an introductory course to Information Science. We discuss the challenges we faced while incorporating NLP and ML to the curriculum followed by a presentation of how we met these challenges. The overall response (of students) to the inclusion of this new topic to the curriculum has been positive. Students this semester are pursuing NLP/ML projects, formulating their own tasks (some of which are novel and presented towards the end of the paper), collecting and annotating data and building models for their task.

## 1 Introduction

An introductory course to Information Science has been taught at Columbia University for over a decade. The main goal of the course is to introduce undergraduates at our university to applications of Computer Science. For most students, this is their first course in the Computer Science department. The course has no pre-requisites such as higher mathematics or programming. In fact, through a survey we found that about 10% of the class did not know the meaning of a programming language.

Traditionally, the computer science applications that have been taught in this course include HTML (creating a website), Spreadsheets, Database Systems, World Wide Web and the Internet, Algorithms and programming in Python. Given the importance of understanding how humans are building *smart* machines and the amount of excitement around Natural Language Processing (NLP) and Machine Learning (ML) applications, we decided to include a social media analysis application – sentiment analysis of Twitter – in the curriculum last year. The overall response to this inclusion has been positive. One outcome of this inclusion is that the students are now able to build basic models for popular NLP applications such as sentiment analysis of Twitter, spam detection of emails, and document classification. But a more significant outcome of this inclusion is that the students seemed to have gained a general idea of how machine learning works, as a result, they find Watson playing Jeopardy! against the humans, and Google's self-driving car less "magical".

There were two main challenges in incorporating an introduction to NLP and ML to the curriculum: (1) we wanted to include this topic without compromising the traditionally covered material, which put a constraint on the number of lectures we could use for introducing NLP and ML and (2) we were required to abstract away from the inherently mathematical jargon used to explain NLP and ML. In this paper we present the way we met these challenges. We present our lecture, homework and examination design that enabled us to get some of the most important ideas of NLP and ML across in one lecture. The students performed exceptionally well on the NLP/ML section of the examination. Moreover, some students are pursuing projects related to these topics formulating their own tasks, collecting and annotating data, and building models to answer their hypotheses. These are signs that undergraduates with a broad spectrum of educational backgrounds and interests are not only capable of tackling the basics of NLP and ML, but that they may even be doing so with relish.

There has been successful and fruitful effort by researchers in the NLP community to share their experiences and course design through this workshop in the past (Lee, 2002; Eisner, 2002; Liddy and McCracken, 2005; Freedman, 2005; Zinsmeister, 2008). Steven Bird (2008) notes that an introductory course needs to serve some common, basic needs – "For some students, it will be the **first step in a pathway** leading to specialized courses, graduate research, or employment in this field. For students who do not continue, the introductory course will be their **main exposure to the field**. Naturally, this course is also a prime **opportunity to promote the field** to newcomers and encourage them to pursue advanced studies in this area." We share the same motivation (as (Bird, 2008)) – our target audience is in fact "newbies" in Computer Science, who may or may not continue with more advanced topics in Computer Science, in which case this course will be their main exposure to the field and thus offers a great opportunity for us to promote the field.

The rest of the paper is structured as follows: In section 2, we give details of the course and student demographics. Section 3 presents the NLP/ML lecture organization and content. In section 4 we present the problems on the mid term examination and performance of the students on the NLP/ML part of the exam. Section 5 describes some of the most interesting student projects that have come out of the course. We conclude in Section 6.

## 2 Student demographics

Students enrolling in this introductory course on Information Science come from a wide variety of academic backgrounds. A majority of the class is undergraduates who have never taken any course in the Computer Science department before. The course is taught over a period of 4 months, consisting of 24 lectures of 75 minute duration each.

Figure 1 and Figure 2 present a distribution of 61 students based on their college rank and major (academic background) respectively.

Figure 1 shows that a large majority of students are freshman and sophomores (50%). While these students have an idea of what they would like to major in, they are not required to finalize their majors until the final semester of their sophomore year. This introductory course is therefore a great opportunity to promote the field by exposing the students to some of the most exciting applications of Computer Science. In the first class of the course, we showed the students a video of Watson playing the popular gameshow Jeopardy! against the humans. It was surprising that only a few students knew of Watson. But even the ones who knew about it were excited and curious to learn how Watson actually works.

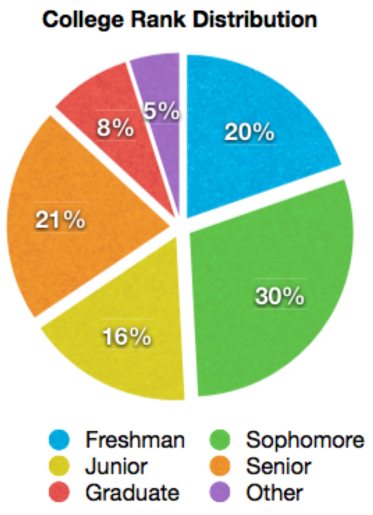

Figure 1: Student distribution based on College rank. 20% Freshman, 30% Sophomore, 16% Junior, 21% Senior, 8% Graduate and 5% Other

Figure 2 presents a distribution of students based on the majors they are pursuing or intend to pursue. For this figure, we grouped the reported majors into the following broader categories: Math/Engineering (Math, Computer Science, Information Science, Electrical Engineering), Basic sciences (Biology, Zoology, Chemistry, Physics, Neuroscience), Political Science, Social Science, Language (German, French, Yiddish, English, Linguistics), Arts and Humanities (including Literature, Film, Theatre), Regional studies (Asian, American, Middle Eastern), and Other (Finance, History, International Relations, Marketing, Philosophy, Psychology).

The distribution of majors shows that the students come from a wide variety of academic backgrounds. Only about 12% of the students are pursuing or intend to pursue a major in Math/Engineering. There is a large majority of students who have only taken

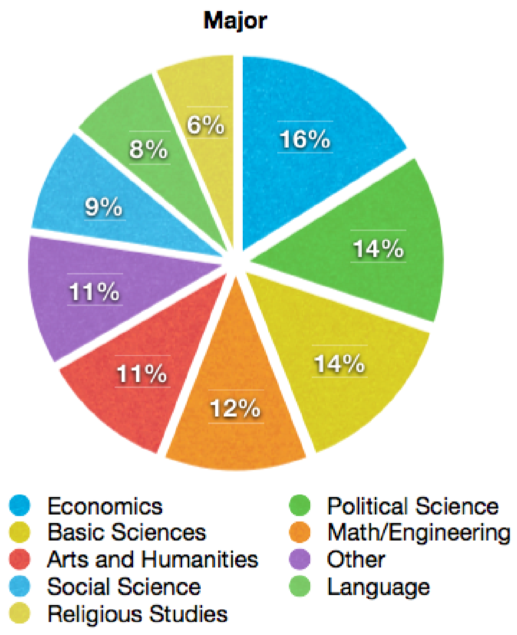

Figure 2: Student distribution based on majors. 16% Economics, 14% Political Science, 14% Basic Sciences, 12% Math/Engineering, 11% Arts and Humanities, 11% Other, 9% Social Science, 8% Language, 6% Regional Studies.

SAT level mathematics. The majority of these students have never used any programming language before. Therefore, one of the main challenges of teaching this course, especially introducing NLP and ML, was to abstract away from mathematical jargon and convey the fundamentals through the use of analogies and concrete illustrations.

## 3 Lecture organization and content

To meet the aforementioned challenges, we spent one lecture introducing the class to some basic concepts of NLP and ML. Through homework and examination, we introduced the students to more NLP applications that also helped them appreciate the strengths and weaknesses of the simple ML technique we introduced in class. We geared the Python part of course towards text processing preparing the students to implement an end-to-end pipeline of a popular NLP application on another homework.

We started the lecture by introducing a concrete and motivating NLP application – sentiment analysis of Twitter. In line with Reva Freedman's (2005) observation, we found that starting with a concrete application is important. We first defined sentiment analysis as the task of building a machine that is able classify the *polarity* of *opinions* in text into one of two categories: positive and negative.[1] We motivated this application by briefly discussing some of its use cases: predicting the outcome of a presidential election, gauging how a company or a product of a company is performing in the market, finding on average *how people are feeling* based on gender, location, age and weather.[2]

After posing the question – *how would a machine learn to predict if a tweet has a positive or a negative sentiment* – we first drew an analogy of how humans learn new concepts. Humans learn through examples and counter-examples. When we see a new object or learn a new concept, our instinct is to compare the new with the familiar. Our first attempt is to find similarities and dissimilarities between this new object with the objects we have already seen. Similarly, to *train* a machine, we first need to provide it with some *labeled examples*. For the task at hand, *examples* are tweets and their *labels* are manually annotated sentiment polarity (positive or negative). Using these training examples, the machine *learns* patterns of words that signify a particular sentiment.

We started with a small list of words, calling them "features". The training data and features are presented in Table 1. We asked the students to fill out each cell in Table 1 by putting a 1 if a tweet contains a particular word and 0 if it does not contain that word. We mentioned that this process is called "feature extraction", in which we convert unstructured data into a structured representation. This representation is structured because each tweet is represented as an ordered and fixed *list of features*.

We asked the students how they would calculate the similarity between two tweets. And we got an obvious answer – count the number of words they have in common.

The next question we asked was "how might the machine calculate the similarity using the structured representation?" The answer to this question was less obvious but once we gave them the formula,

[1]We defined the italicized words and gave examples to help students understand the definitions. We intentionally kept the definition of sentiment analysis simple and restricted to classifying polarity of opinions into positive and negative categories.

[2]http://www.wefeelfine.org

| Tweet ID | Tweet | good | bad | not | pretty | great | Label |
|---|---|---|---|---|---|---|---|
| T1 | It's a good day :) | 1 | 0 | 0 | 0 | 0 | +1 |
| T2 | The weather is pretty bad | 0 | 1 | 0 | 1 | 0 | -1 |
| T3 | Alice is pretty | 0 | 0 | 0 | 1 | 0 | +1 |
| T4 | Bieber is not all that great | 0 | 0 | 1 | 0 | 1 | -1 |
| S1 | It is a good day for biking | 1 | 0 | 0 | 0 | 0 | ? |
| S2 | The situation is not pretty | 0 | 0 | 1 | 1 | 0 | ? |
| S3 | Such a great show :) | 0 | 0 | 0 | 0 | 1 | ? |

Table 1: Training and test data used in class to illustrate how a machine will learn to predict the polarity of tweets.

the students were able to grasp it quickly. We introduced the formula as a bit-wise multiplication of list of features followed by the summation of the resulting bits.

$$Sim(T, S) = \sum_{i=1}^{d} t_i \times s_i$$

where $T, S$ are tweets, $d$ is the number of features in the list of features, $t_i, s_i$ are the $i^{th}$ bit of tweets $T$ and $S$ respectively.

The next question we asked was given a tweet, whose polarity is unknown (an *unseen* tweet), how might they use the training data to predict its polarity. This was a harder question, and though we did not expect an answer, we posed this question nonetheless to serve as a pause in the lecture and indicate that a key idea was coming.

Before revealing the secret sauce, we made the analogy of how humans would do a similar task. Given two kinds of fish, say sea bass and salmon, the way we would classify a new fish into one of these two categories would be by comparing "features" of the new fish with the features of sea bass and with the features of salmon followed by observing if the new fish is "closer" to sea bass or salmon. Similarly, the machine will compare the list of features of the unseen tweet with the list of features of the positive and the list of features of the negative tweets and compute a *similarity score* that will allow the machine to make a prediction about the polarity of this unseen tweet.

We then introduced the following formula:

$$s = \sum_{i=1}^{N} Sim(T_i, S) \times Label_i$$

where $N$ is the total number of training examples, $T_i$ is the $i^{th}$ training example, $S$ is the test tweet and $Label_i$ is the human annotated polarity of $T_i$.

The machine uses this score to make a final prediction. If the score is less than or equal to 0, the machine predicts the polarity of the tweet as negative. If the score is greater than 0, the machine predicts the polarity of the tweet as positive.

We illustrated this by working out a few examples of how the machine will go about predicting the polarity of the following unseen tweets:

1. "It is a good day for biking"
2. "The situation is not pretty"
3. "Such a great show :)"

We worked out the first example on the board and asked the students to work out the remaining two on their own. Following is the way in which we worked out the first example on the board.

1. First the machine converts the test tweet S1 = "It is a good day for biking" into the same structured representation as that of the training tweets. The list of features for S1 is [1,0,0,0,0] (see Table 1).
2. Then the machine compares the list of features for S1 with each of the training tweets as follows:
   (a) Comparing the list of features for tweets T1 and S1, the machine finds the bit-wise multiplication of their feature lists $[1, 0, 0, 0] \times [1, 0, 0, 0] = [1, 0, 0, 0]$. Then the machine adds all the bits $1+0+0+0 = 1$. We point out there is only one word in

common between the two tweets (namely "good"). The similarity score between the first training example and the test example $s_1 = 1 \times (+1) = 1$.

(b) Similarly, comparing the feature lists for T2 and S1, we get a similarity score $s_2 = ([0,1,0,1,0] \times [1,0,0,0,0]) \times (-1) = 0$

(c) Comparing the feature lists for T3 and S1, we get a similarity score $s_3 = ([0,0,0,1,0] \times [1,0,0,0,0]) \times (+1) = 0$

(d) Finally, comparing the feature lists for T4 and S1, we get a similarity score $s_4 = ([0,0,1,0,0] \times [1,0,0,0,0]) \times (-1) = 0$

3. Next, the machine adds all the similarity scores together to get an aggregated score for the test tweet $s = s_1 + s_2 + s_3 + s_4 = 1$. Since $s > 0$, the machine predicts this test tweet T1, "It is a good day for biking", has a positive polarity.

Having the students work out the other two examples in class on their own and interacting with their neighbors, they began to see the meaning of pattern recognition. Bringing their attention to Table 1, we pointed out that the word "good" is associated with a positive polarity by virtue of appearing in a positively labeled tweet. The word "pretty" is associated with a *neutral* polarity because it appears both in a positive and in a negative tweet. This means that the machine has learned that it cannot make a prediction simply based on the word "pretty". The test tweet "The situation is not pretty" makes this point explicit. This tweet is classified correctly as negative but only because of the presence of the word "not", which appears in a negative tweet.

In summary, through these worked out examples, we were able to drive home the following points:

1. The machine automatically learns the connotation of words by looking at how often certain words appear in positive and negative tweets.

2. The machine also learns more complex patterns that have to do with the conjunction and disjunction of features.

3. The quality and amount of training data is important – for if the training data fails to encode a substantial number of patterns important for classification, the machine is not going to learn well.

Students asked the following questions, which helped us build on the aforementioned points.[3]

1. Good and great are synonyms. Shouldn't we count them as one feature?

2. Could we create and use a dictionary that lists the prior polarity of commonly used words?

3. If the prediction score for the tweet is high, does that mean we the machine is more confident about the prediction?

4. In this approach, the sequence of words does not matter. But clearly, if "not" does not negate the words containing opinion, then won't the machine learn a wrong pattern?

5. If we have too many negative tweets in our training data (as compared to the positive tweets), then would the machine not be predisposed to predict the polarity of an unseen tweet as negative?

Building on these concepts, we had the students work through an end-to-end example of classifying movie reviews into positive and negative on their homework. What appeared to be a promising machine learning technique in class, seemed to fail for this task. They realized that classifying movie reviews is much harder because of the words used in plot descriptions that mislead the classifier. We used examples from the seminal paper by Peter Turney (2002) for this homework problem.

## 4 Problem and performance on the Mid term examination

We further built on the fundamentals, by asking the students to classify emails into "important" and "unimportant" by using the same machine learning technique (used for sentiment analysis of Twitter) on their mid term examination. This helped them see that the ML technique learned in class may be used, in general, for other NLP applications. As Heike Zinsmeister (2008) notes, redundancy and iterative re-introduction could be helpful for students,

[3]Questions are reformulated for succinctness and clarity.

we found that by having the students work out different NLP applications using the same ML approach helped them grasp the concepts better and appreciate the strengths and weaknesses of this simple ML approach.

Table 2 presents the training data along with the features. Following are the problems from their midterm examination.

1. Extract features from the emails in the training data, i.e. fill Table 2 with ones and zeros. (5 points)

2. What will be the prediction of the machine for this new incoming email "It is important that you register for this meeting. – your phd advisor". Say if, this is an important email, is the prediction made by your machine correct? (4 + 1 points)

3. What will be the prediction of the machine for this new incoming email "Bill, what up?". Say if, this is an unimportant email, is the prediction made by your machine correct? (4 + 1 points)

4. What is the performance of your current machine learning model on all the test data? (2 points)

5. What other feature(s) will you add to the list of features to improve the performance of your machine learning model? How will this change the prediction of the two incoming emails? What will be the performance of your new model? (3 + (2 + 2) + 1 points)

For problem 5 on the exam, most of the students came up with the answer of adding the words "your" and "advisor" to the list of features. But some students devised more complex features. One student proposed to add the capitalization feature to distinguish between "Bill" and "bill". Another student extended this feature to additionally check if "Bill" is a proper noun or not. The only type of feature we introduced in class was the binary occurrence and non-occurrence of words. It was promising to see the students expand on the preliminary feature set to create novel and more advanced set of features.

The duration of the exam was 75 min and it consisted of 6 extended problems. The first two problems were compulsory and the students were asked to do any two out of the remaining four problems (NLP/ML, Logic Gates, Database Design, Machine Instructions). Each of the remaining four problems was worth 25 points. Table 3 shows their performance on the four problems. The table shows that the students did extremely well on the NLP/ML problem – averaging 20.54 out of 25 with a standard deviation of 4.46. Note, students unanimously attempted the NLP/ML part of the exam – only 2 students scored a zero for this problem as compared to 17, 11 and 23 students, who scored a zero on Logic Gates, Database Design and Machine Instructions respectively.[4]

The performance of students on the mid term examination assured us that they were comfortable with the terminology and the process of machine learning. We decided to build on this foundation by introducing them to basic text processing, indexing, and stemming in the Python part of the course. On their Python homework, they implemented a complete pipeline, starting from creating a vocabulary from the training data, then extracting features, and finally implementing a simple version of the perceptron algorithm to predict sentiment polarity of tweets. The average on this homework was 87.8 out of 115 with about 60% of the students scoring over 100 points.

## 5 Student project descriptions

The most exciting outcome of including NLP and ML to the course has been that some students have signed up for a course project in their demanding curriculum. For the course project, the students were asked to formulate their own tasks, collect and annotate data and build machine learning models for their tasks. Following are the two most novel task formulations (in students' own language) followed by a list of other projects.[5]

Detecting liberal or conservative biases (Allen Lipson and Tyler H. Dratch): Critics on both sides of the political spectrum often accuse their adversaries of employing biased language to promote

[4]The grading was generous and students were given partial credit for their attempt. Therefore, we approximate the number of students who attempted a problem by counting the number of students who scored a zero on that problem.

[5]The project reports are available at www.cs.columbia.edu/∼apoorv/Teaching/ProjectReports

| Email ID | Email | meeting | register | unsubscribe | bill | Label |
|---|---|---|---|---|---|---|
| E1 | Meeting at 4, hurry! – your advisor. | ... | ... | ... | ... | +1 |
| E2 | Free event. To register click here. To unsubscribe click here. | ... | ... | ... | ... | -1 |
| E3 | According to our register, your bill is yet to be paid | ... | ... | ... | ... | +1 |
| E4 | Register for this useless meeting. | ... | ... | ... | ... | -1 |

Table 2: Structured representation of the training data or examples from which the machine will learn to differentiate between important and unimportant emails.

| Problem | Average | Std-dev | Median | Count (Score $< 5$) | Count (Score == 0) |
|---|---|---|---|---|---|
| NLP/ML | 20.54 | 4.46 | 22 | 2 | 2 |
| Logic Gates | 16.94 | 6.48 | 20 | 20 | 17 |
| Database Design | 13.63 | 6.48 | 14 | 14 | 11 |
| Machine Instructions | 12.8 | 6.81 | 14.5 | 27 | 23 |

Table 3: Distribution of scores for 53 students on problems on the mid term exam. Students were required to do any two out of these four problems. Each problem was worth 25 points. Count (Score $< 5$) means the number of students out of 53 that scored less than 5 points on a problem. Average, standard deviation and median values exclude students who scored a 0.

an agenda. Nowhere in politics is the usage of language more contentious than in the immigration debate. Conservatives lambast "illegal aliens"; liberals defend "undocumented workers." Liberals promote a "path to citizenship"; conservatives decry "criminal amnesty". But is this bias also present in major news sources, the supposedly impartial sources of society's information? Or are papers like the New York Times and the Wall Street Journal firmly on opposite sides of the immigration debate? We want to put this question to the test. We are constructing a machine learning algorithm to detect liberal or conservative biases on immigration in the New York Times and the Wall Street Journal.

The Bechdel Test (Michelle Adriana Marguer Cheripka and Christopher I. Young): The Bechdel Test is a measure by which it is possible to identify gender bias in fiction. In order to pass the test, the work of fiction must pass three criteria: there must be two main female characters, they must have a conversation, and they must be speaking about something other than a man. Though primarily used in film, the Bechdel test can also be applied to literature. In previous Bechdel experiments, the results indicated traditional, heteronormative pattern. While a text does not necessarily need to be explicitly feminist in order to pass the test, the test itself is an important gauge for the social roles that societies uphold and perpetuate. This particular experiment was created in order to determine if this trend was consistent across mediums. Considering that children's books provide the foundation for a person's interaction with literature, the test could identify patterns that emerge from an early stages of literature and address their future impact.

Some of the other project proposals are as follows: Gim Hong Lee built a sentiment analysis engine to rate a professor based on his/her reviews available on CULPA.info (Columbia Underground Listing of Professor Ability). Xueying (Alice) Lin built a recommendation system for Yelp. She acquired the data-set from kaggle.com.[6] A group of three students (Roni Saporta, Moti Volpo and Michal Schestowitz) experimented with the breast cancer data-set available at the UCI data-repository.[7] They used scikit-learn's[8] implementation of the logistic regression algorithm.

It is heartening to see that students who had very limited (or no) idea about how machines learn at the start of the course are now formulating tasks and at-

[6]http://www.kaggle.com/c/yelp-recruiting
[7]http://archive.ics.uci.edu/ml/
[8]http://scikit-learn.org/stable/

tempting to build their own machine learning models. What they still do not know, we believe, is that they are mapping each document into a finite dimensional feature space and calculating dot products between feature vectors to calculate similarity between documents. While this math vocabulary is probably required to make more progress and dive deeper into NLP and ML, we believe it is not required to convey the essence of pattern recognition.

## 6 Conclusion

In this paper, we presented a lecture, homework and examination design, through which we were able to get some basic ideas of Natural Language Processing and Machine Learning across to students who came from a wide variety of academic backgrounds, majority of whom did not have an advanced math background. Apart from the challenge of having to abstract away from the inherently mathematical concepts, we faced another challenge at the onset of designing the lecture – we had to deliver the NLP and ML material in one or two lectures so that we do not compromise on the traditionally covered topics.

We believe that the lecture, homework and examination design presented in this paper may be used by lecturers teaching introductory course such as ours or by researchers who are interested in presenting a simplified explanation of NLP and ML to general popular science audiences.

## Acknowledgments

We would like to thank Kapil Thadani, Caronae Howell, Kshitij Yadav, Owen Rambow, Meghna Agarwala, Sara Rosenthal, Daniel Bauer and anonymous reviewers for useful comments.

## References

Steven Bird. 2008. Defining a core body of knowledge for the introductory computational linguistics curriculum. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 27–35, Columbus, Ohio, June. Association for Computational Linguistics.

Jason Eisner. 2002. An interactive spreadsheet for teaching the forward-backward algorithm. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 10–18, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Reva Freedman. 2005. Concrete assignments for teaching NLP in an M.S. program. In *Proceedings of the Second ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 37–42, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Lillian Lee. 2002. A non-programming introduction to computer science via nlp,ir,and ai. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 33–38, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Elizabeth Liddy and Nancy McCracken. 2005. Hands-on NLP for an interdisciplinary audience. In *Proceedings of the Second ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 62–68, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *In the Proceedings of the 40th meeting of Association of Computational Linguisitcs (ACL 2002).*

Heike Zinsmeister. 2008. Freshmen's CL curriculum: The benefits of redundancy. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pages 19–26, Columbus, Ohio, June. Association for Computational Linguistics.