The evolution of a statistical NLP course

Fei Xia
University of Washington
Department of Linguistics
Box 354340
Seattle, WA 98195-4340
fxia@u.washington.edu

Abstract

This paper describes the evolution of a statistical NLP course, which I have been teaching every year for the past three years. The paper will focus on major changes made to the course (including the course design, assignments, and the use of discussion board) and highlight the lessons learned from this experience.

1 Introduction

In the past two decades, there has been tremendous progress in natural language processing (NLP) and NLP techniques have been applied to various real-world applications such as internet/intranet search and information extraction. Consequently, there has been an increasing demand from the industry for people with special training in NLP. To meet the demand, the University of Washington recently launched a new Professional Masters Program in Computational Linguistics (CLMA). To earn the master's degree, students must take nine courses and complete a final project. The detail of the program can be found in (Bender et al., 2008).

One of the required courses is LING572 (Advanced statistical methods in NLP), a course that I have been teaching every year for the past three years. During the process and especially in Year 3, I have made many changes to course content, assignments, and the usage of discussion board. In this paper, I will describe the evolution of the course and highlight the lessons learned from this experience.

2 Background

LING572 is part of the four-course NLP core sequence in the CLMA program. The other three are LING570 (Shallow Processing Techniques for NLP), LING571 (Deep Processing Techniques for NLP), and LING573 (NLP Systems and Applications). LING570 and LING571 are organized by NLP tasks (e.g., language model, POS tagging, Named-entity tagging, chunking for LING570, and parsing, semantics and discourse for LING571); LING572 is organized by machine learning methods; LING573 is the place where students use the knowledge learned in LING570-572 to build NLP systems for some real applications (e.g., question answering and information extraction).

The prerequisites for LING572 are (1) at least one college-level course in probability and statistics, (2) strong programming skills, and (3) LING570. The quarter is ten weeks long, with two 80-minute sessions per week. The class size is relatively small, with ten to twenty students. Most students in LING572 are from the CLMA program and are taking LING571 and other NLP courses at the same time. About a half of the students are from computer science background, and the remaining are from linguistics or other humanity background.

3 Course content

It would be impossible to cover all the major machine learning (ML) algorithms in one quarter; therefore, one of the decisions that I made from the very beginning is that the course would focus on major classification algorithms and spend only one week showing how these algorithms can be applied to sequence labeling problems. I believe that once

Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics (TeachCL-08), pages 45–53,
Columbus, Ohio, USA, June 2008.

students have a solid grasp of these algorithms, it would be easy for them to learn algorithms for other kinds of learning problems (e.g., regression, clustering, and ranking).

The next question is what classification algorithms should be included in the syllabus. Table 1 shows the topics covered each year and the number of sessions spent on each topic. The topics can be roughly divided into three units: (1) supervised learning, (2) semi- and unsupervised learning, and (3) other related topics.

3.1 Year 1

The teaching plan for Year 1 turned out to be too ambitious. For instance, six supervised algorithms were covered in four weeks (i.e., 8 sessions) and four semi-/unsupervised algorithms were covered in 2.5 weeks. Such a tight schedule did not leave sufficient time for students to digest all the important concepts and equations.

3.2 Year 2

In Year 2, I reduced the amount of time spent on Unit (2). For instance, I spent only one session discussing the main ideas in the EM algorithm, without going through the details of the mathematic deduction and special cases of the algorithm such as the backward-forward algorithm and the inside-outside algorithm. Other changes were made to other units, as shown in the second column of Table 1.

3.3 Year 3

In the first two years, my lecturing style was similar to the tutorials given at major NLP conferences (e.g., ACL) in that I covered a lot of material in a short period of time and expected students to digest all the details after class. This approach did not work very well because our students came from very diverse backgrounds (e.g., linguistics, literature, computer science) and many of them were not familiar with mathematic concepts (e.g., Lagrangian, dual problem, quadratic programming, hill climbing) that are commonly used in machine learning. Most of the students were also new to NLP, taking only one quarter of NLP-related courses before taking LING572.

Based on this observation, I made major changes to the syllabus in Year 3: I reduced the lecture material and spent more time on discussion and illustrative examples. For example, on average 1.25 sessions were spent on a supervised algorithm in Year 2 and that number increased to 2.17 sessions in Year 3. I also added one session on information theory, which provides theoretic foundation for many learning methods. Because of this change, some important topics had to be cut, as shown in the last column of Table 1. Fortunately, I was able to incorporate some of the removed topics to two other courses (i.e., LING570 and a seminar on semi- and unsupervised learning) that I was teaching in the same year. The feedback from students indicated that the new course plan for Year 3 was more effective than the ones for the previous two years.

Another change I made was that I divided the teaching material into three types: (1) the essential knowledge that students should fully understand, (2)

Table 1: Content changes over the years (-: topics not covered, *: topics moved to LING570 in the previous quarter, †: topics moved to an NLP seminar)

	Y1	Y2	Y3
(1) Supervised learning:			
kNN	-	1	1
Decision Tree	1	1	1
Decision List	1	1	-
Naive Bayes	-	1	2
Maximum entropy (MaxEnt)	2	2	4
Support vector machine (SVM)	-	-	4
Transformation-based learning (TBL)	2	1	1
Bagging	1	1	-
Boosting	1	2	-
subtotal	**8**	**10**	**13**
(2) Semi-/unsupervised learning:			
Semisupervised	2	1	1†
Unsupervised	3	1	-†
subtotal	**5**	**2**	**1**
(3) Other topics:			
Introduction	1	1	1
Information theory	-	-	1
Feature selection	-	1	1
System combination	1	-	-
Relation between FSA and HMM	1	-	-*
Multiclass → binary	-	1	-*
Beam search	-	1	-*
Student presentation	1	1	-
Recap, summary	3	2	3
subtotal	**7**	**7**	**6**
Total	20	19	20

more advanced topics that students should be aware of but do not have to understand all the details, and (3) related topics that are not covered in class but are available on *additional slides* for people who want to learn more by themselves. Taking MaxEnt as an example, Type (1) includes the maximum entropy principle, the modeling, GIS training, and decoding; Type (2) includes regularization and the mathematic proof that shows the relation between maximum likelihood and maximum entropy as provided in (Ratnaparkhi, 1997), and Type (3) includes L-BFGS training and the similarity between SVM and MaxEnt with regularization (Klein, 2007). Making this distinction helps students focus on the most essential part of the algorithms and at the same time provides additional material for more advanced students.

4 Reading material

One challenge of teaching a statistic NLP course is the lack of good textbooks on the subject; as a result, most of the reading material come from conference and journal papers. The problem is that many of the algorithms covered in class were originally proposed in non-NLP fields such as machine learning and applied mathematics, and the original papers are often heavy in mathematical proofs and rarely refer to the NLP tasks that our students are familiar with. On the other hand, NLP papers that apply these algorithms to NLP tasks often assume that the readers are already familiar with the algorithms and consequently do not explain the algorithms in detail.

Because it is hard to find a suitable paper to cover all the theoretic and application aspects of a learning algorithm, I chose several papers for each algorithm and specified the sections that the students should focus on. For instance, for Maximum Entropy, I picked (Berger et al., 1996; Ratnaparkhi, 1997) for the basic theory, (Ratnaparkhi, 1996) for an application (POS tagging in this case), and (Klein and Manning, 2003) for more advanced topics such as optimization and smoothing.

For the more sophisticated learning methods (e.g., MaxEnt and SVM), it is very important for students to read the assigned papers beforehand. However, some students choose not to do so for various reasons; meanwhile, other students might spend too much time trying to understand everything in the papers. To address this problem, in Year 3 I added five reading assignments, one for each of the following topics: information theory, Naive Bayes, MaxEnt, SVM, and TBL. Each assignment consists of simple questions such as the one in Appendix A. Students were asked to turn in their answers to the questions before class. Although the assignments were very simple, the effect was obvious as students started to ask in-depth questions even before the topics were covered in class.

5 Written and programming assignments

In addition to the reading assignments mentioned above, students also have weekly assignments. For the sake of clarity, we divide the latter into two types, *written* and *programming* assignments, depending on whether programming is required. Significant changes have been made to both types, as explained below.

5.1 Year 1

In Year 1, there were three written and six programming assignments. The written assignments were mainly on mathematic proof, and one example is given in Appendix B. The programming assignments asked students to use the following existing packages to build NLP systems.

1. Carmel, a finite state transducer package written by Jonathan Graehl at USC/ISI.

2. fnTBL (Ngai and Florian, 2001), an efficient implementation of TBL created by Ngai and Florian at JHU.

3. A MaxEnt toolkit written by Le Zhang, available at http://homepages.inf.ed.ac.uk/s0450736.

To complete the assignments, the students needed to study some functions in the source code to understand exactly how the learning algorithms were implemented in the packages. They would then write pre- and post-processing tools, create data files, and build an end-to-end system for a particular NLP task. They would then present their work in class and write a final paper to report their findings.

5.2 Year 2

In Year 2 I made two major changes to the assignments. First, I reduced the number of written assignments on theoretic proof. While such assignments strengthen students' mathematic capability, they were very challenging to many students, especially the ones who lacked mathematics and statistics training. The assignments were also not as effective as programming assignments in understanding the basic concepts for the learning algorithms.

The second major change was to replace the three packages mentioned above with Mallet (McCallum, 2002), a well-known package in the NLP field. Mallet is a well-designed package that contains almost all the learning methods covered in the course such as Naive Bayes, decision tree, MaxEnt, boosting, and bagging; once the training and test data were put into the Mallet data format, it was easy to run all these methods and compared the results.

For the programming assignments, in addition to reading certain Mallet functions to understand how the learning methods were implemented, the students were also asked to extend the package in various ways. For instance, the package includes a text-user-interface (TUI) class called *Vectors2Classify.java*, which produces a classifier from the training data, uses the classifier to classify the test data, compares the results with gold standard, and outputs accuracy and the confusion matrix. In one assignment, students were asked to first separate the code for training and testing, and then add the beam search to the latter module so that the new code would work for sequence labeling tasks.

While using Mallet as a black box is straightforward, extending it with additional functionality is much more difficult. Because the package did not have a detailed document that explained how its main classes should be used, I spent more than a week going through hundreds of classes in Mallet and wrote a 11-page guide based on my findings. While the guide was very helpful, many students still struggled with the assignments, especially the ones who were not used to navigating through other people's code and/or who were not familiar with Java, the language that Mallet was written in.

5.3 Year 3

To address these problems, in Year 3, we changed the focus of the assignments: instead of studying and extending Mallet code, students would create their own package from scratch and use Mallet only as a reference. For instance, in one assignment, students would implement the two Naive Bayes models as described in (McCallum and Nigam, 1998) and compare the classification results with the results produced by the Mallet Naive Bayes learner. In the beam search assignment, students' code would include modules that read in the model produced by Mallet and calculate $P(y \mid x)$ for a test instance x and a class label y. Because the code no longer needed to call Mallet functions, students were free to use whatever language they were comfortable with and could treat Mallet as a black box.

The complete assignments are shown in Appendix C. In summary, students implemented six learners (Naive Bayes, kNN, Decision Tree, MaxEnt, SVM, TBL),[1] beam search, and the code for feature selection. All the coding was completed in eight weeks in total, and the students could choose to either work alone or work with a teammate.

6 Implementation issues

All the programming assignments in Year 3, except the one for MaxEnt, were due in a week, and students were expected to spend between 10 and 20 hours on each assignment. While the workload was demanding, about 90% of students completed the assignments successfully. Several factors contribute to the success:

- All the learners were evaluated on the same classification task.[2] The input and output data format were very similar across different learners; as a result, the code that handled input and output could be reused, and the classification results of different learners could be compared directly.

[1] For SVM, students implemented only the decoder, not the trainer, and they would test their decoder with the models produced by libSVM (Chang and Lin, 2001).

[2] The task is a simplified version of the classic 20-newsgroup text classification task, with only three out of the 20 classes being used. The training data and the test data consist of 900 and 100 examples from each class, respectively.

- I restricted the scope of the assignments so that they were doable in a week. For instance, the complexity of a TBL learner highly depends on the form of the transformations and the type of learning problem. In the TBL assignment, the learner was used to handle classification problems and the transformation had the form *if a feature is present in an instance, change the class label from A to B*. Implementing such a learner was much easier than implementing a learner (e.g., fnTBL) that use more complex transformations to handle sequence labeling problems.

- Efficiency is an important issue, and there are often differences between algorithms on paper and the code that implements the algorithms. To identify those differences and potential pitfalls that students could run into, I completed all the assignments myself at least a week before the assignments were due, and shared some of my findings in class. I also told students the kind of results to be expected, and encouraged students to discuss the results and implementation tricks on the discussion board.

Implementing machine learning algorithms is often an art, as there are many ways to improve efficiency. Two examples are given below. While such tricks are well-known to NLP researchers, they are often new to students and going through them in class can help students to speed up their code significantly.

The trainer for TBL

As described in (Brill, 1995), a TBL trainer picks one transformation in each iteration, applies it to the training data, and repeats the process until no more good transformations can be found. To choose the best transformation, a naive approach would enumerate all the possible transformations, for each transformation go through the data once to calculate the net gain, and choose the transformation with the highest net gain. This approach is very inefficient as the data have to be scanned through multiple times.[3]

A much better implementation would be to go through the training data only once, and for each feature in each training instance, update the net gains of the corresponding transformations accordingly.[4] Students were also encouraged to read (Ngai and Florian, 2001), which proposed another efficient implementation of TBL.

The decoder for Naive Bayes

In the multi-variate Bernoulli event model for the text classification task (McCallum and Nigam, 1998), at the test time the class for a document d is chosen according to Eq (1). If we calculate $P(d|c)$ according to Eq (2), as given in the paper, we have to go through all the features in the feature set F. However, as shown in Eq (3) and (4), the first product in Eq (3), denoted as $Z(c)$ in Eq (4), is a constant with respect to d and can be calculated beforehand and stored with each c. Therefore, to classify d, we only need to go through the features that are present in d. Implementing Eq (4) instead of Eq (2) reduces running time tremendously.[5]

$$c^* = arg\ max_c P(c)P(d|c) \tag{1}$$

$$P(d|c) = \prod_{f \notin d}(1 - P(f|c)) \prod_{f \in d} P(f|c) \tag{2}$$

$$= \prod_{f \in F}(1 - P(f|c)) \prod_{f \in d} \frac{P(f|c)}{1 - P(f|c)} \tag{3}$$

$$= Z(c) \prod_{f \in d} \frac{P(f|c)}{1 - P(f|c)} \tag{4}$$

7 Discussion board

A discussion board is one of the most effective vehicles for outside-class communication and collaboration, as anyone in the class can start a new conversation, read recent posts, or reply to other peo-

[3] Let N_f be the number of features and N_c be the number of classes in a classification task, the number of transformations in the form we specified above is $O(N_f N_c^2)$, which means that the learner has to go through the data $O(N_f N_c^2)$ times.

[4] For each feature t in each training instance x, if x's current label y_c is different from the true label y, there would be only one transformation whose net gain would be affected by this feature in this instance, and the transformation is *if t is present, change class label from y_c to y*. If y_c is the same as y, there would be $N_c - 1$ transformations whose net gain would be affected, where N_c is the number of classes.

[5] This trick was actually pointed out by a student in my class.

ple's posts.[6] Furthermore, some students feel more comfortable posting to a discussion board than raising the questions in class or emailing the instructor. Therefore, I provided a discussion board each time LING572 was offered and the board was linked to the course website. However, the board was not used as much as I had hoped in the first two years.

In Year 3, I took a more pro-active approach: first, I reminded students several times that emails to me should be reserved only for confidential questions and all the non-confidential questions should be posted to the discussion board. They should also check the discussion board at least daily and they were encouraged to reply to their classmates' questions if they knew the answers. Second, if a student emailed me any questions that should go to the discussion board, I would copy the questions to the board and ask the sender to find my answers there. Third, I checked the board several times per day and most of the questions raised there were answered within an hour if not sooner.

As a result, there was a significant increase of the usage of the board, as shown in Table 2. For instance, compared to Year 2, the average number of posts per student in Year 3 more than quadrupled, and at the same time the number of emails I received from the students was cut by 65%. More importantly, more than a half of the questions posted to the board were answered by other students, indicating the board indeed encouraged collaboration among students.

A lesson I learned from this experience is that the success of a discussion board relies on active participation by its members, and strong promotion by the instructor is essential in helping students take advantage of this form of communication.

Table 2: The usage of the course discussion board

	Y1	Y2	Y3
# of students	15	16	11
# of conversations	13	47	116
Total # of posts	42	149	589
# of posts by the instructor	7	21	158
# of posts by students	35	128	431
Ave # of post/student	2.3	8	39.2

8 Course evaluation

Students were asked to evaluate the course at the end of the quarter using standard evaluation forms. The results are shown in Table 3.[7] For (1)-(11), students were asked to answer the questions with a 6-point scale: 0 being *Very Poor* and 5 being *Excellent*. The question for (12) is "*on average how many hours per week have you spent on this course?*"; The question for (13) is "*among the total average hours spent on the course, how many do you consider were valuable in advancing your education?*" The values for (1)-(13) in the table are the average of the responses. The last row, Challenge and Engagement Index (CEI), is a score computed from several items on the evaluation form, and reported as a decile rank ranging from 0 (lowest) to 9 (highest). It reflects how challenging students found the course and how engaged they were in it.

The table shows that the overall evaluation in Year 2 was worse than the one in Year 1, despite much effort put into improving course design. The main problem in Year 2 was the programming assignments, as discussed in Section 5.2 and indicated in Row (11): many students found the task of extending Mallet overwhelming, especially since some of them had never used Java and debugged a large pre-existing package before. As a result, they spent much time on learning Java and trying to figure out how Mallet code worked, and they felt that was not the best way to learn the subjects (cf. the big gap between the values for Row (12) and (13)).

Based on the feedback from the first two years, in Year 3 I made a major overhaul to the course, as discussed in Sections 3-7 and summarized here:

- The lecture material was cut substantially; for instance, the average number of slides used in a session was reduced from over 60 in Year 2 to below 30 in Year 3. The saved time was spent on class discussion and going through examples on the whiteboard.

- Reading assignments were introduced to help students focus on the most relevant part of the

[6] The software we used is called *GoPost*. It is one of the Web-based communication and collaboration applications developed by the Center for Teaching, Learning, and Technology at the University of Washington.

[7] The complete form has thirty questions, the most relevant ones are listed in the table.

Table 3: Student evaluation of instruction (For Item (1)-(11), the scale is between 0 and 5: 0 is *Very Poor* and 5 is *Excellent*; The scale for Item (14) is between 0 and 9, 9 being *most challenging*)

	Y1	Y2	Y3
Number of respondents	14	15	11
(1) The course as a whole	3.9	3.8	5.0
(2) The course content	4.0	3.7	4.9
(3) Course organization	4.0	3.8	4.8
(4) Explanations by instructor	3.7	3.5	4.6
(5) Student confidence in instructor's knowledge	4.5	4.5	4.9
(6) Instructor's enthusiasm	4.5	4.6	4.9
(7) Encouragement given students to express themselves	3.9	4.3	4.6
(8) Instructor's interest in whether students learned	4.5	4.0	4.8
(9) Amount you learned in the course	3.8	3.3	4.8
(10) Relevance and usefulness of course content	4.3	3.9	4.9
(11) Reasonableness of assigned work	3.8	2.0	3.8
(12) Average number of hours/week spent on the course	7.9	21.9	19.8
(13) How many were valuable in advancing your education	6.2	14.5	17.2
(14) Challenge and engagement index (CEI)	7	9	9

reading material.

- Instead of extending Mallet, students were asked to create their own packages from scratch and many implementation issues were addressed in class and in the discussion board.

- Discussion board was highly promoted to encourage outside-class discussion and collaboration, and its usage was increased dramatically.

As shown in the last column of the table, the new strategies worked very well and the feedback from the students was very positive. Interestingly, although the amount of time spent on the course in Y2 and Y3 was about the same, the students in Y3 felt the assigned work was more reasonable than the students in Y2. This highlights the importance of choosing appropriate assignments based on students' background. Also, while the lecture material was cut substantially over the years, students in Y3 felt that they learned more than the students in Y1 and Y2, implying that it is more beneficial to cover a small number of learning methods in depth than to hurry through a large number of topics.

9 Conclusion

Teaching LING572 has been a great learning experience, and significant changes have been made to course content, assignments, and the like. Here are some lessons learned from this experience:

- A common pitfall for course design is being over-ambitious with the course plan. What matters the most is not how much material is covered in class, but how much students actually digest.

- When using journal/conference papers as reading material, it is often better to select multiple papers and specify the sections in the papers that are most relevant. Giving reading assignments would encourage students to read papers before class and provide guidelines as what questions they should focus on.

- Adding new functionality to an existing package is often difficult if the package is very complex and not well-documented. Therefore, this kind of assignments should be avoided if possible. In contrast, students often learn more from implementing the methods from scratch than from reading other people's source code.

- Implementing ML methods is an art, and pointing out various tricks and potential obstacles beforehand would help students tremendously. With careful design of the assignments and in-class/outside-class discussion of implementa-

tion issues, it is possible to implement multiple learning methods in a short period of time.

- Discussion board is a great venue for students to share ideas, but it will be successful only if students actively participate. The instructor can play an important role in promoting the usage of the board.

Many of the lessons above are not specific to LING572, and I have made similar changes to other courses that I am teaching. So far, the feedback from the students have been very positive. Compared to the first two years, in Year 3, students were much more active both in and outside class; they were much more satisfied with the assignments; many students said that they really appreciated all the implementation tips and felt that they had a much better understanding of the algorithms after implementing them. Furthermore, several students expressed interest in pursuing a Ph.D. degree in NLP.

In the future, I plan to replace some of the early ML algorithms (e.g., kNN, Decision Tree, TBL) with more recent ones (e.g., conditional random field, Bayesian approach). This adjustment has to be done with special care, because the early algorithms, albeit quite simple, often provide the foundation for understanding more sophisticated algorithms. I will also fine tune the assignments to make them more manageable for students with less CS/math training.

A Reading assignment example

The following is the reading assignment for MaxEnt in Year 3.

(Q1) Let P(X=i) be the probability of getting an i when rolling a dice. What is $P(X)$ according to the maximum entropy principle under the following condition?
(a) P(X=1) + P(X=2) = 0.5
(b) P(X=1) + P(X=2) = 0.5 and P(X=6) = 0.2

(Q2) In the text classification task, $|V|$ is the number of features, $|C|$ is the number of classes. How many feature functions are there?

(Q3) How to calculate the empirical expectation of a feature function?

B Written assignment example

The following is part of a written assignment for Boosting in Year 1: In the basic AdaBoost algorithm, let h_t be the hypothesis created at time t, $D_t(i)$ be the weight of the i-th training instance, and ϵ_t be the training error rate of h_t. Let the hypothesis weight α_t be $\frac{1}{2} ln \frac{1-\alpha_t}{\alpha_t}$ and the normalization factor Z_t be $\sum_i D_t(i) e^{-\alpha_t y_i h_t(x_i)}$. Prove that $Z_t = 2\sqrt{\alpha_t(1-\alpha_t)}$ for any t.

C Programming assignment examples

In Year 3, there were seven programming assignments, as summarized below:

Hw1: Implement the two Naive Bayes models as described in (McCallum and Nigam, 1998).

Hw2: Implement a decision tree learner, assuming all features are *binary* and using information gain as the split function.

Hw3: Implement a kNN learner using cosine and Euclidean distance as similarity/dissimilarity measures. Implement one of feature selection methods covered in class, and test the effect of feature selection on kNN.

Hw4: Implement a MaxEnt learner. For training, use General Iterative scaling (GIS).

Hw5: Run the svm-train command in the libSVM package (Chang and Lin, 2001) to create a SVM model from the training data. Write a decoder that classifies test data with the model.

Hw6: Implement beam search and reduplicate the POS tagger described in (Ratnaparkhi, 1996).

Hw7: Implement a TBL learner for the text classification task, where a transformation has the form *if a feature is present in a document, change the class label from A to B.*

For Hw6, students compared their POS tagging results with the ones reported in (Ratnaparkhi, 1996). For all the other assignments, students tested their learners on a text classification task and compare the results with the ones produced by pre-existing packages such as Mallet and libSVM.

Each assignment was due in a week except for Hw4 and Hw6, which were due in 1.5 weeks. Students could choose to work alone or work with a teammate.

References

Emily Bender, Fei Xia, and Erik Bansleben. 2008. Building a flexible, collaborative, intensive master's program in computational linguistics. In *Proceedings of the Third ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, Columbus, Ohio, June.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1), March.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Dan Klein and Christopher Manning. 2003. Maxent model, conditional estimation, and optimization. ACL 2003 tutorial.

Dan Klein. 2007. Introduction to Classification: Likelihoods, Margins, Features, and Kernels. Tutorial at NAACL-2007.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *In AAAI/ICML-98 Workshop on Learning for Text Categorization*.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Grace Ngai and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of North American ACL (NAACL-2001)*, pages 40–47, June.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part-of-speech Tagging. In *Proc. of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-1996)*, Philadelphia.

Adwait Ratnaparkhi. 1997. A simple introduction to maximum entropy models for natural language processing. Technical Report Technical Report 97-08, Institute for Research in Cognitive Science, University of Pennsylvania.