Appears in *Proceedings of AAAI-98/ICML-98 Workshop on Learning for Text Categorization and the AAAI-98 Workshop on Recommender Systems,*pp.49-54 and pp.70-74, Madison, WI, July 1998

Book Recommending Using Text Categorization with Extracted Information

Raymond J. Mooney
Paul N. Bennett
Department of Computer Sciences
University of Texas
Austin, TX 78712-1188
{mooney,pbennett}@cs.utexas.edu

Loriene Roy
Graduate School of Library
and Information Science
University of Texas
Austin, TX 78712-1188
loriene@gslis.utexas.edu

Abstract

Content-based recommender systems suggest documents, items, and services to users based on learning a profile of the user from rated examples containing information about the given items. Text categorization methods are very useful for this task but generally rely on unstructured text. We have developed a book-recommending system that utilizes semi-structured information about items gathered from the web using simple information extraction techniques. Initial experimental results demonstrate that this approach can produce fairly accurate recommendations.

Introduction

There is a growing interest in *recommender systems* that suggest music, films, and other items and services to users (e.g. `www.bignote.com`, `www.filmfinder.com`) (Maes 1994; Resnik & Varian 1997). These systems generally make recommendations using a form of computerized matchmaking called *collaborative filtering*. The system maintains a database of the preferences of individual users, finds other users whose known preferences correlate significantly with a given patron, and recommends to a user other items enjoyed by their matched patrons. This approach assumes that a given user's tastes are generally the same as some other user of the system and that a sufficient number of users and ratings are available. Learning individualized profiles from descriptions of examples (*content-based recommending* (Balabanovic & Shoham 1997)), on the other hand, allows a system to uniquely characterize each patron without having to match their interests to someone else's.

Learning for text-categorization has been applied to content-based recommending of web pages (Pazzani, Muramatsu, & Billsus 1996) and newsgroup messages (Lang 1995). We have been exploring book recommending by applying text-categorization to semi-structured text extracted from the web. Our current prototype system, LIBRA (Learning Intelligent Book Recommending Agent), uses a database of book information extracted from web pages at Amazon.com. [1] Users provide 1–10 ratings for a selected set of training books; the system then learns a profile of the user and produces a ranked list of the most recommended titles. Experimental results on 1,000 literary fiction books rated by two separate users demonstrate that the system makes reasonably accurate recommendations—producing moderate correlations after 20 examples and strong correlations after 60 examples.

Unlike previous research on recommending web pages and news postings, the text used to represent examples is structured into fields such as author, title, abstract, and subject terms. This structured text is extracted from Amazon's book-description web pages using a simple *information extraction* system (Lehnert & Sundheim 1991; Cardie 1997). The resulting examples are then represented using set-valued features (Cohen 1996a; 1996b) with a feature for each slot whose value is the set of words appearing in that slot. Such an approach selects and organizes a subset of the information presented on a web-page or other document and can produce a more concise, structured, and useful representation of examples. Such an approach seems more appropriate for recommending particular types of items (like books, music, software, etc.) for which semi-structured descriptive text is available.

With respect to evaluation, we believe it is important to evaluate the continuous rankings produced by recommender systems rather than just the "thumbs-up/thumbs-down" predictions. A ranking provides more information and a user usually wants to pursue only a few of the most highly-rated examples. We use Spearman's ranked correlation coefficient to compare the system's ranking of the test examples to the ranking imposed by the user's 1-10 ratings. We believe this provides more appropriate information than binary classification accuracy and is a useful way to evaluate recommenders.

System Description

Extracting Information and Building a Database

First, an Amazon subject search is performed to obtain a list of book-description URL's of relevant titles. Cur-

[1] Current book-recommending systems at Amazon.com and BarnesAndNoble.com apparently use some form of collaborative filtering.

rently we have assembled databases for science fiction (2,600 titles) and literary fiction (3,061 titles). LIBRA then downloads each of these pages and uses a simple pattern-based information-extraction system to extract data about each title. The current slots utilized by the recommender are: title, authors, synopses (including excerpts from published reviews), and subject terms. A number of other slots are also extracted (e.g. publisher, date, ISBN, price, related titles, customer ratings and reviews, etc.) but are currently not used by the recommender.

Since the layout of Amazon's automatically generated pages is quite regular, a fairly simple extraction system is sufficient. LIBRA's extraction system is hand-written and employs a pattern matcher that utilizes pre-filler, filler, and post-filler patterns as described by Califf & Mooney (1998). In other applications, more sophisticated information extraction methods and inductive learning of extraction rules might be useful (Cardie 1997). The text in each slot is then processed into an unordered set of words/tokens and the examples represented as a vector of set-valued features.

Learning a Profile

Next, the user selects and rates a set of training books. By searching for particular authors or titles, the user can avoid scanning the entire database. The user is asked to provide a discrete 1–10 rating for each selected title. The rating assigned to each book is interpreted as its category.

The inductive learner currently employed by LIBRA is a fairly simple feature-based naive Bayesian classifier extended to efficiently handle set-valued features. Each word appearing in a given slot is treated as a binary feature and hash tables are used to efficiently store and index conditional probabilities for only the words actually occurring in each slot in the training data.[2] Probabilities are smoothed using Laplace estimates (Kohavi, Becker, & Sommerfield 1997), which also provides nonzero probabilities for any novel words encountered during testing. Calculation with logarithms of probabilities is used to avoid underflow. Finally, in order to avoid considering every possible slot and word combination during testing, the system precomputes the posterior probability of each category assuming the value for each feature is the empty set. During testing, the system simply adjusts this default probability to account for the words actually present in the given example. This trick makes testing time linear in the size of an example rather than linear in the size of the entire vocabulary.

[2]Note that using each word as a binary feature is different from using the probability that a given word is identical to one randomly selected from all the text in a given category (Mitchell 1997; Joachims 1997).

Producing, Explaining, and Revising Recommendations

Once a profile is learned from the training data, it is used to predict the rating of the remaining books and then the N top-scoring recommendations are presented to the user. After computing the posterior probability of each of the ten ratings categories for a test example, the system calculates an expected value for the rating, $\sum_{i=1}^{10} iP(i)$, where $P(i)$ is the posterior probability for category i. We use the expected value rather than simply choosing the most probable category in order to better represent the continuity of scores. Consider the case where $P(3) = 0.35$, $P(9) = 0.32$, and $P(10) = 0.33$; Even though 3 is the most probable category, the "closeness" of the other categories makes it more likely that the example would fall toward the high end. Using the expected value of 7.23 addresses this issue. When using this 10-category model to predict a binary category (positive: rating > 5; negative: rating ≤ 5), we classify an example as positive if and only if $\sum_{i=6}^{10} P(i) > \sum_{i=1}^{5} P(i)$. LIBRA can also be trained specifically for binary categorization with the posterior odds (Pearl 1988) of the positive category used to rank the test examples. A third option, which we will call the *weighted binary* approach, maps the user's 1 - 10 rating r into a weight, w_r, in the closed interval [0,1], where $w_r = \frac{r-1}{9}$. The general formula for this is $w_r = \frac{r-min}{max-min}$, where $0 \leq min \leq r \leq max$ and $max \neq min$. Then, if a word occurs in n training examples given a rating of r, it is counted as occurring nw_r times in positive examples and $n(1 - w_r)$ in negative examples. The ranked predictions are once again produced by ordering based on posterior odds of positive. Unless otherwise stated, we have adopted the first approach in our experiments.

The system also has a limited ability to "explain" its recommendations by listing the M features that most contributed to its high rank. For example, when trained for binary categorization, the system presented the following explanation for a particular recommendation:

> The Gods Themselves by Issac Asimov classified as POSITIVE because:
> words:award(4.20), words:earth(4.20), words:terrify(4.20), words:truth(3.71), words:Nebula(2.96), words:Hugo(2.96), words:alien(2.96), words:die(2.96), words:scientist(1.25), author:Asimov(1.08).

The weight presented for each feature f is $\log(P(f \mid P)/P(f \mid N))$ where P and N represent the positive and negative class respectively.

After reviewing the recommendations, the user may assign their own rating to examples they believe to be incorrectly ranked and retrain the system to produce improved recommendations. As with *relevance feedback* in information retrieval (Salton & Buckley 1990), this cycle can be repeated several times in order to produce the best results.

Experimental Results

Methodology

Data Collection Of the first 5,500 URL's returned from the keyword search "literature fiction" on Amazon, 3,061 were judged as unique (differing ISBN's) *adequate information* pages. An adequate information page contains at least one instance of the following slots: *comments*, *reviews*, or *synopses*. Two sets of 1,000 titles were chosen randomly from these 3,061 titles, and each set was evaluated by one user. The two data sets shared 589 titles in common. Both users were presented with the page in a web browser and entered an integer rating from 1–10, inclusive. *Data Set 1* contained 64% negative ratings (i.e. ≤ 5) compared to 60% for *Data Set 2*. The textual data obtained from Amazon is fairly noisy, including incorrectly indexed synopses and spelling errors, and there is a wide amount of variance in the length and quality of book descriptions.

Performance Measures To evaluate performance, we ran 10-fold cross-validation and examined two performance measures, binary classification accuracy and Spearman's rank correlation coefficient (r_s). Learning curves were generated by training on increasingly larger subsets of the data reserved for training. The statistical significance of differences in average performance were evaluated using a 2-tailed paired t-test. We distinguish throughout this paper between a *rating* and a *ranking*, where a rating is a real number assigned to an example by the user or system; whereas, a ranking is the ordinal place an example occupies in the ordering of examples by their ratings. Using a ranking coefficient as a general performance measure in recommender systems instead of a ratings coefficient has two benefits (1) The system need not provide a mapping into the user's interval of ratings (i.e. 1–10). (2) By translating the ratings to rankings we essentially linearize the data with respect to the dimension we are analyzing. These benefits make it likely that the generality of this measure will make it useful in evaluating many types of systems in addition to accurately judging non-linear but correlated ratings. By using r_s, we are able to capture the extent to which the ranked user scores and ranked system predictions covary. As with other correlation coefficients, r_s ranges from -1 to 1 (inclusive), where -1 is perfectly inversely correlated, 0 denotes no correlation, and 1 signifies perfect direct correlation. A correlation coefficient of 0.3 to 0.6 is generally considered "moderate" and above 0.6 is considered "strong." In order to compute r_s when there are ties in the data, the approach recommended by Anderson & Finn (1996) was used. When there are no ties, this reduces to the form given in most introductory statistics texts (Spatz & Johnston 1984).

Systems and Hypotheses Our current experiments compare a simple binary classifier, a 10-ratings classifier which uses the expected value to predict ratings, and a weighted binary classifier (hereafter referred to as Binary, 10-Ratings, and Weighted Binary, respectively). We expected that with sufficient training data the 10-Ratings method would outperform the Binary classifier on the rank correlation measure since it exploits the users' actual 1–10 rating. However, we expected that the Binary method would perform better on binary classification accuracy since it is specifically designed for that task. Finally, the Weighted Binary approach should perform better than the Binary approach for ranking since it exploits the user's 1–10 ratings, though there is some question as to whether it has the expressiveness to outperform the 10-Ratings Classifier in the limit.

Results

Figures 1 and 2 show the results for running all the systems on *Data Set 1*. Figures 3 and 4 show the results for running all the systems on *Data Set 2*. Overall, the predictions are reasonably accurate even given relatively small training sets (25 examples). Moderate correlations (above 0.3) are produced after about 20 examples and strong correlations (above 0.6) after about 60 examples.

While the Binary model outperformed both the 10-Ratings model and the Weighted Binary model for binary prediction on *Data Set 1*, the difference between any of the models is not statistically significant. Though from about 55 examples to about 150 examples, the Binary model outperforms both others by a statistically significant amount. Although even in this early region, the statistical significance wavers at various points. On *Data Set 2* the Binary model once again outperformed the 10-Ratings model for binary prediction but not by a significant amount. The Binary model's superior performance to the Weighted Binary model for binary prediction on *Data Set 2* was, however, significant at the 0.05 level. The difference between the Weighted Binary and 10-Ratings model was not significant for binary prediction on *Data Set 2*.

The 10-Ratings model outperformed the Binary method over both data sets on the r_s measure after 900 training examples (significant for *Data Set 1* and *Data Set 2* at the 0.01 and 0.02 level, respectively). However, it is interesting to note that the correlation curves crossover on both data sets indicating that binary categorization is preferable for smaller training sets. The Weighted Binary model also outperformed the Binary method over both data sets on the r_s measure (significant at the 0.01 level for both). There is, however, no significant crossover point between the Weighted Binary classifier and the Binary classifier as the Weighted Binary model was not noticeably outperformed with few training examples. In both data sets the Weighted Binary outperforms the 10-Ratings model early in the learning curve, though only *Data Set 2* contained several sequential points where the difference was significant. At the point with 900 training examples, the difference in the r_s measure between the Weighted Bi-

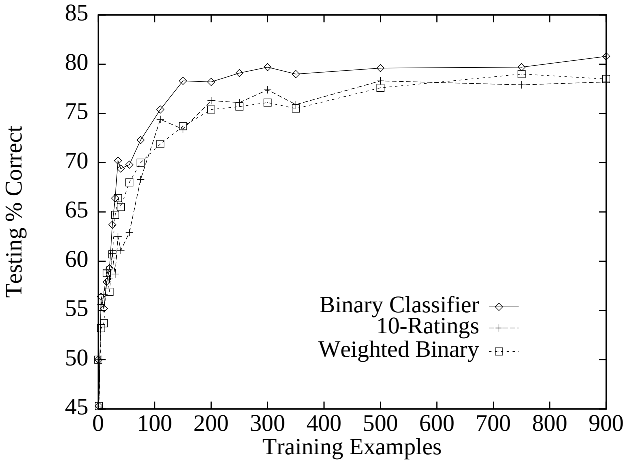

Figure 1: Binary Prediction Accuracy for Data Set 1

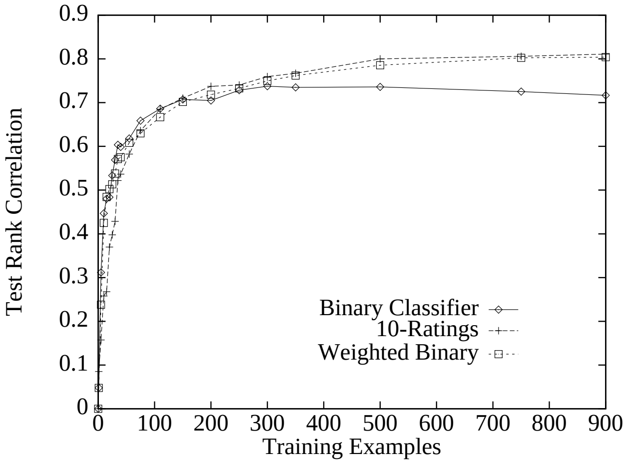

Figure 2: Rank Correlation Coefficient for Data Set 1

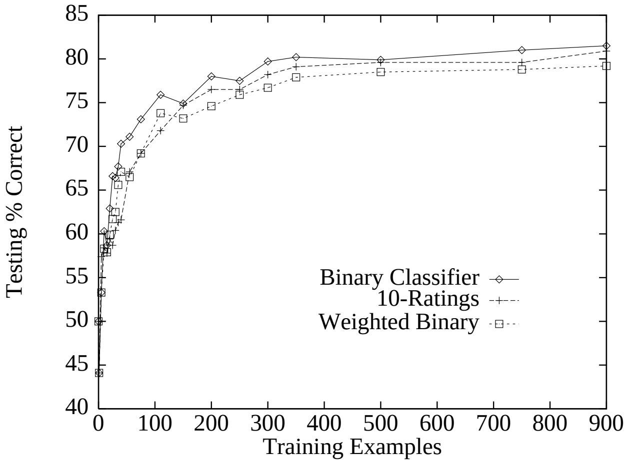

Figure 3: Binary Prediction Accuracy for Data Set 2

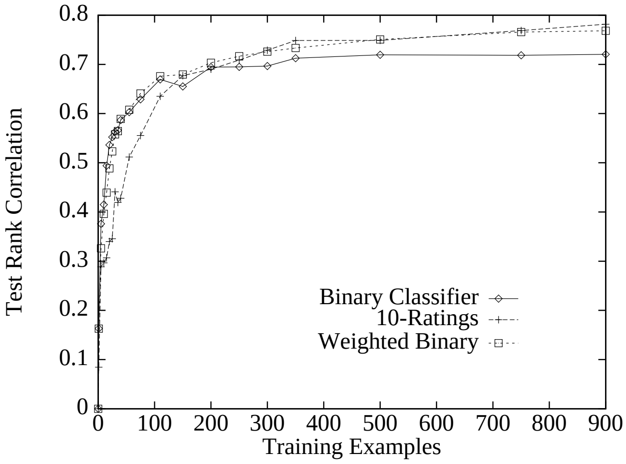

Figure 4: Rank Correlation Coefficient for Data Set 2

nary and the 10-Ratings model is not significant.

Discussion

While the similarity of performance of the various methods on binary prediction is of some note, it is more interesting that in both the binary accuracy curve and the rank correlation coefficient curve, the 10-Ratings model learned more slowly than the Binary model. This results from having more parameters (10 times as many) to learn and relatively sparse, insufficient data to accurately estimate them when there are few training examples. As the Weighted Binary model has less parameters than the 10-Ratings model, we see better performance early in the curve of the Weighted Binary model.

By the end of the rank correlation coefficient curve, there is a significant gain in the use of the 10-Ratings model over the Binary model for ranking. However, the crossover point (at least where it becomes statistically significant) for both data sets occurs after hundreds of training examples. Therefore, since users will often be willing to rate only a relatively small number of examples, obtaining enough ratings to produce good results from the 10-Ratings method could often be impractical. However, as the Weighted Binary model performs comparable to the Binary model early on and comparable to the 10-Ratings model later in the curve, this suggests that the Weighted Binary model may be the best choice. We also have indications that modifications to the 10-Ratings approach or Weighted Binary model look most promising. In the scatter plots for prediction on the *training* examples (Figures 5, 6, and 7), an obvious pattern emerges (we use prediction over the training examples to demonstrate this point because of the greater number of data points (900) and much higher correlations). Clearly, the binary method learns a binary separator for those ratings above five and those at or below five with little order beyond the two-way separator. In contrast, the more expressive 10-Ratings Classifier and Weighted Binary Classifier learn a graduated separation. The ability of the 10-Ratings method and Weighted Binary method to capture this richer model with sufficient training data is supported by the differ-

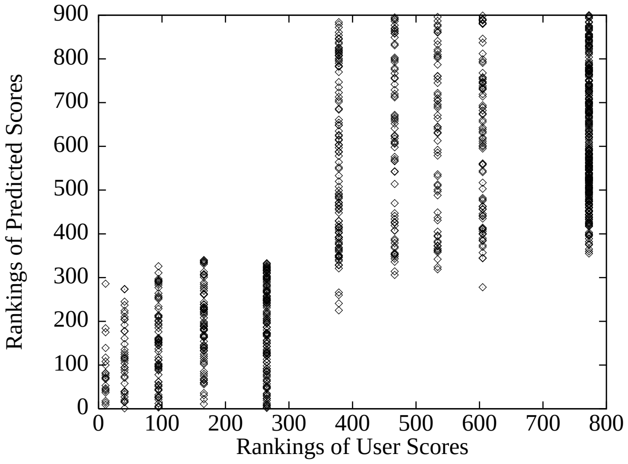

Figure 5: Scatter plot of Ranking for Prediction on the Training Data for One of the Ten Trial Runs Over Data Set 1 Using Binary Classifier (900 training examples)

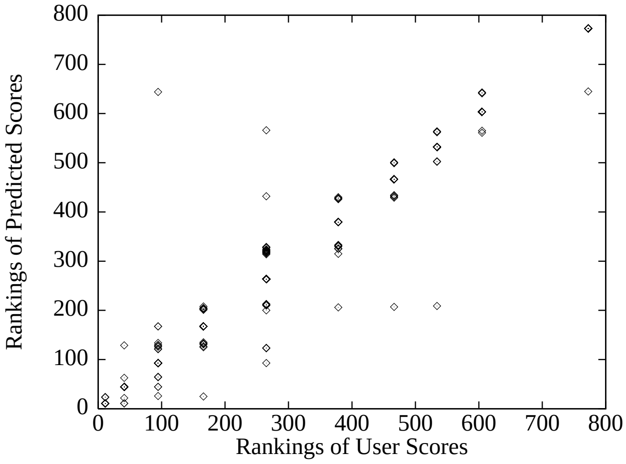

Figure 6: Scatter plot of Ranking for Prediction on The Training Data for One of the Ten Trial Runs Over Data Set 1 Using 10-Ratings Classifier (900 training examples)

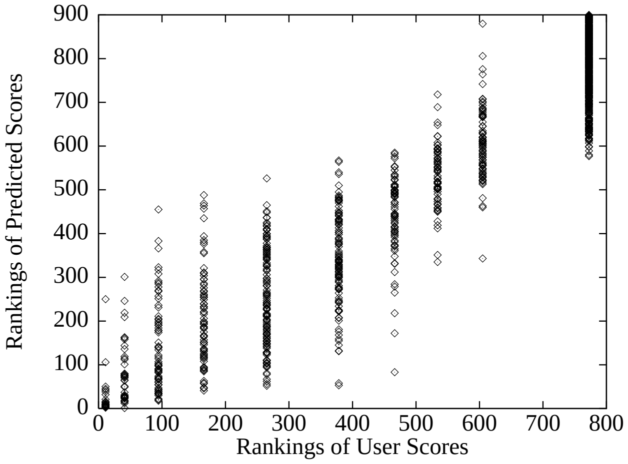

Figure 7: Scatter plot of Ranking for Prediction on the Training Data for One of the Ten Trial Runs Over Data Set 1 Using Weighted Binary Classifier (900 training examples)

ence in the rank correlation coefficients over the test examples. Also, note that there appear to be far fewer datapoints in Figure 6. This is actually the result of having many ties in the predictions the system is producing. There are actually 255 examples at the point $x = 772.5$, $y = 773$ and one at $x = 772.5$, $y = 645$ in Figure 6. Thus the 10-Ratings model is able to fit the training data extremely well (most of the users' ratings are ties since they are integer scores); this suggests that the expressiveness of the 10-Ratings model is leading to overfitting the training data which would further explain why it does not ultimately outperform the Weighted Binary method.

Thus, the results indicate that a model which uses fewer parameters is more likely to perform well with fewer training examples, but a model will only perform better with a large number of training examples if it also preserves the continuity of the user ratings. This is exactly what the Weighted Binary model does. In fact, the way the Weighted Binary model outperforms the 10-Ratings system when there are few training examples is almost definitely a result of having fewer parameters to estimate. Since the Binary model performs similarly early on however, this alone would be worth very little note. What is more interesting is that the Weighted Binary model continues to perform at levels not significantly different than the 10-Ratings predictions after the Binary - 10-Ratings crossover point.

Future Work

The current interface to Libra is through a library of Lisp functions. Producing a user-friendly web-based interface would make the system more accessible.

Comparing different text-categorization algorithms for this application is an obvious area for future research. The ability to produce continuous probability or confidence estimates is an important requirement for presenting ordered recommendations. Algorithms also need to be easily adaptable to the structured (slot-filler) representation produced by information-extraction.

Including other extracted information (e.g. related books, customer ratings and reviews) in the description of examples also needs to be explored. In addition, an examination of the benefits of various methods of feature extraction and selection should be conducted. An experimental comparison of utilizing extracted information to simply using entire pages would be useful in demonstrating the utility of the overall information-extraction approach. Combining information about an item extracted from multiple sources (e.g. Amazon and BarnesAndNoble) is yet another issue.

Allowing a user to initially provide keywords that are of known interest and incorporating this information into learned profiles could also be helpful (Pazzani & Billsus 1997). Combining the current content-based approach with information about other users' ratings (such as those extracted from Amazon) is another interesting direction.

Conclusions

Content-based recommender systems for books and other items is an interesting and challenging application for learning and text categorization. Unlike arbitrary text, descriptive documents about such items can be organized and structured by first using information extraction to assemble relevant information about each item. Representing examples using set-valued features is then one way to allow learning algorithms to exploit the resulting structured information.

We have developed a book recommending system, LIBRA, that utilizes learning and text categorization applied to extracted information. The system employs a simple Bayesian text-categorization algorithm extended to efficiently handle set-valued features. Initial experimental results evaluating the accuracy of its recommendations are very promising. However, the current initial prototype can be improved and extended in many ways in order to improve its accuracy and usability. Eventually such content-based recommender systems based on text-categorization techniques could provide a useful service to consumers overwhelmed by the abundance of choices presented by the modern world.

Acknowledgements

This research was partially supported by the National Science Foundation through grant IRI-9704943 and a research award from the University of Texas Graduate School of Library and Information Science. Sincere thanks go to Tina Bennett who provided the ratings for one of the data sets.

References

Anderson, T., and Finn, J. D. 1996. *The New Statistical Analysis of Data*. New York: Springer-Verlag, Inc.

Balabanovic, M., and Shoham, Y. 1997. Fab: Content-based, collaborative recommendation. *Communications of the Association for Computing Machinery* 40(3):66–72.

Califf, M. E., and Mooney, R. J. 1998. Relational learning of pattern-match rules for information extraction. In *Working Notes of AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, 6–11. Menlo Park, CA: AAAI Press.

Cardie, C. 1997. Empirical methods in information extraction. *AI Magazine* 18(4):65–79.

Cohen, W. W. 1996a. Learning rules that classify e-mail. In *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, 18–25. AAAI Press.

Cohen, W. W. 1996b. Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 709–716.

Joachims, T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 143–151. San Francisco, CA: Morgan Kaufman.

Kohavi, R.; Becker, B.; and Sommerfield, D. 1997. Improving simple Bayes. In *Proceedings of the European Conference on Machine Learning*.

Lang, K. 1995. NewsWeeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, 331–339. San Francisco, CA: Morgan Kaufman.

Lehnert, W., and Sundheim, B. 1991. A performance evaluation of text-analysis technologies. *AI Magazine* 12(3):81–94.

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the Association for Computing Machinery* 37(7):31–40.

Mitchell, T. 1997. *Machine Learning*. New York, NY: McGraw-Hill.

Pazzani, M., and Billsus, D. 1997. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning* 27(3):313–331.

Pazzani, M.; Muramatsu, J.; and Billsus, D. 1996. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 54–61.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA: Morgan Kaufmann Publishers, Inc., revised second printing edition.

Resnik, P., and Varian, H. R. 1997. Introduction (to the special section on recommender systems). *Communications of the Association for Computing Machinery* 40(3):56–59.

Salton, G., and Buckley, C. 1990. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41:288–297.

Spatz, C., and Johnston, J. O. 1984. *Basic Statistics, Tales of Distributions*. Belmont, CA: Wadsworth, Inc., third edition.