Content-Based Book Recommending Using Learning for Text Categorization

Raymond J. Mooney
Department of Computer Sciences
Loriene Roy
Graduate School of Library and Information Science
University of Texas
Austin, TX 787 12
Email : mooney@cs.utexas.edu, loriene@gslis.utexas.edu

ABSTRACT

Recommender systems improve access to relevant products and information by making personalized suggestions based on previous examples of a user's likes and dislikes. Most existing recommender systems use collaborative filtering methods that base recommendations on other users' preferences. By contrast, content-based methods use information about an item itself to make suggestions. This approach has the advantage of being able to recommend previously unrated items to users with unique interests and to provide explanations for its recommendations. We describe a content-based book recommending system that utilizes information extraction and a machine-learning algorithm for text categorization. Initial experimental results demonstrate that this approach can produce accurate recommendations.

KEYWORDS: Recommender systems, information filtering, machine learning, text categorization

INTRODUCTION

There is a growing interest in *recommender systems* that suggest music, films, books, and other products and services to users based on examples of their likes and dislikes [40, 19, 2,461. A number of successful startup companies like Firefly, Net Perceptions, and LikeMinds have formed to provide recommending technology. On-line book stores like Amazon and BarnesAndNoble have popular recommendation services, and many libraries have a long history of providing *render's advisory* services [4, 29]. Such services are important since readers' preferences are often complex and not readily reduced to keywords or standard subject categories, but rather best illustrated by example. Digital libraries should be able to build on this tradition of assisting readers by providing cost-effective, informed, and personalized automated recommendations for their patrons.

Existing recommender systems almost exclusively utilize a form of computerized matchmaking called *collaborative filtering* or *social filtering* [1, 16, 39, 27]. 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 person other items enjoyed by his or her matched patrons. This approach assumes that a given user's tastes are generally the same as another user of the system and that a sufficient number of user ratings are available. Items that have not been rated by a sufficient number of users cannot be effectively recommended. Unfortunately, statistics on library use indicate that most books are utilized by very few patrons [20]. Therefore, collaborative approaches naturally tend to recommend popular titles, perpetuating homogeneity in reading choices. Also, since significant information about other users is required to make recommendations, this approach raises concerns about privacy and access to proprietary customer data. Finally, although newly introduced items are frequently of particular interest to users, it is impossible for a collaborative approach to recommend items that no one has yet rated or purchased.

Learning individualized profiles from descriptions of examples *(content-based recommending* [5]), on the other hand, allows a system to uniquely characterize each patron without having to match his or her interests to another's, Items are recommended based on information about the item itself rather than on the preferences of other users. This also allows for the possibility of providing explanations that list content features that caused an item to be recommended, potentially giving readers confidence in the system's recommendations and insight into their own preferences. Finally, a content-based approach can allow users to provide initial subject information to aid the system.

Machine learning for text-categorization has been applied to content-based recommending of web pages [37] and newsgroup messages [23]; however, to our knowledge has not previously been applied to book recommending. A content-based approach was employed in one of the first book recommending systems [41, 42]; however, the system developer had to laboriously hand-label each book with values for a pre-selected set of features and users had to provide specific traits about themselves in addition to evaluating recommended books. Content-based approaches have also been applied to recommending other items, such as movies [6, 2]. However, these studies tend to employ very limited sets of features (e.g. actors, directors, genres, ratings) compared to the thousands of distinct words present in even short descriptive texts (e.g. abstracts and reviews). We have been exploring content-based book recommending by applying automated text-categorization methods 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. Therefore, the system's current content information about titles consists of textual meta-data rather than the actual text of the items themselves. Users provide 1–10 ratings for a selected set of training books; the system then learns a profile of the user using a Bayesian learning algorithm and produces a ranked list of the most recommended additional titles from the system's catalog.

As evidence for the promise of this approach, we present initial experimental results on several data sets of books randomly selected from particular genres such as mystery, science, literary fiction, and science fiction and rated by different users. We use standard experimental methodology from machine learning and present results for several evaluation metrics on independent test data including rank correlation coefficient and average rating of top-ranked books.

The remainder of the paper is organized as follows. Section 2 provides an overview of the system including the algorithm used to learn user profiles. Section 3 presents results of our initial experimental evaluation of the system. Section 4 discusses topics for further research, and section 5 presents our conclusions on the advantages and promise of content-based book recommending.

SYSTEM DESCRIPTION

Extracting Information and Building a Database

First, an Amazon subject search is performed to obtain a list of book-description URLs of broadly relevant titles. LIBRA then downloads each of these pages and uses a simple pattern-based information-extraction system to extract data about each title. Information extraction (IE) is the task of locating specific pieces of information from a document, thereby obtaining useful structured data from unstructured text [24, 12]. Specifically, it involves finding a set of substrings from the document, called *fillers*, for each of a set of specified *slots*. When applied to web pages instead of natural language text, such an extractor is sometimes called a *wrapper* [22]. The current slots utilized by the recommender are: title, authors, synopses, published reviews, customer comments, related authors, related titles, and subject terms. Amazon produces the information about related authors and titles using collaborative methods; however, LIBRA simply treats them as additional content about the book. Only books that have at least one synopsis, review or customer comment are retained as having adequate content information. A number of other slots are also extracted (e.g. publisher, date, ISBN, price, etc.) but these are not used by the recommender. We have initially assembled databases for literary fiction (3,061 titles), science fiction (3,813 titles), mystery (7,285 titles), and science (6,177 titles).

Since the layout of Amazon's automatically generated pages is quite regular, a fairly simple extraction system is sufficient. LIBRA's extractor employs a simple pattern matcher that uses pre-filler, filler, and post-filler patterns for each slot [8]. In other applications, more sophisticated information extraction methods and inductive learning of extraction rules might be useful [9].

The text in each slot is then processed into an unordered bag of words (tokens) and the examples represented as a vector of bags of words (one bag for each slot). A book's title and authors are also added to its own related-title and related-author slots, since a book is obviously "related" to itself, and this allows overlap in these slots with books listed as related to it. Some minor additions include the removal of a small list of stop-words, the preprocessing of author names into unique tokens of the form first-initial_last-name and the grouping of the words associated with synopses, published reviews, and customer comments all into one bag (called "description").

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 or picking selections at random. The user is asked to provide a discrete 1–10 rating for each selected title.

The inductive learner currently employed by LIBRA is a bag-of-words simple Bayesian text classifier [30] extended to handle a vector of bags rather than a single bag. Recent experimental results indicate that this relatively simple approach to text categorization performs as well or better than many competing methods [18, 28]. LIBRA does not attempt to predict the exact numerical rating of a title, but rather just a total ordering (ranking) of titles in order of preference. This task is then recast as a probabilistic binary categorization problem of predicting the probability that a book would be rated as positive rather than negative, where a user rating of 1–5 is interpreted as negative and 6–10 as positive. As described below, the exact numerical ratings of the training examples are used to weight the training examples when estimating the parameters of the model.

Specifically, we employ a multinomial text model [28], in which a document is modeled as an ordered sequence of word events drawn from the same vocabulary, V. The "naive Bayes" assumption states that the probability of each word event is dependent on the document class but independent of the word's context and position. For each class, c_j, and word or token, $w_k \in V$, the probabilities, $P(c_j)$ and $P(w_k|c_j)$ must be estimated from the training data. Then the posterior probability of each class given a document, D, is computed using Bayes rule:

$$P(c_j|D) = \frac{P(c_j)}{P(D)} \prod_{i=1}^{|D|} P(a_i|c_j)$$

where a_i is the ith word in the document, and $|D|$ is the length of the document in words. Since for any given document, the prior $P(D)$ is a constant, this factor can be ignored if all that is desired is a ranking rather than a probability estimate. A ranking is produced by sorting documents by their odds ratio, $P(c_1|D)/P(c_0|D)$, where c_1 represents the positive class and c_0 represents the negative class. An example is classified as positive if the odds are greater than 1, and negative otherwise.

In our case, since books are represented as a vector of "documents," d_m, one for each slot (where s_m denotes the mth slot), the probability of each word given the category and the slot, $P(w_k|c_j, s_m)$, must be estimated and the posterior category probabilities for a book, B, computed using:

$$P(c_j|B) = \frac{P(c_j)}{P(B)} \prod_{m=1}^{S} \prod_{i=1}^{|d_m|} P(a_{mi}|c_j, s_m)$$

where S is the number of slots and a_{mi} is the ith word in the mth slot.

Parameters are estimated from the training examples as follows. Each of the N training books, B_e $(1 \leq e \leq N)$ is given two real weights, $0 \leq \alpha_{ej} \leq 1$, based on scaling its user rating, r $(1 \leq r \leq 10)$: a positive weight, $\alpha_{e1} = (r-1)/9$, and a negative weight $\alpha_{e0} = 1 - \alpha_{e1}$. If a word appears n times in an example B_e, it is counted as occurring $\alpha_{e1}n$ times in a positive example and $\alpha_{e0}n$ times in a negative example. The model parameters are therefore estimated as follows:

$$P(c_j) = \sum_{e=1}^{N} \alpha_{ej}/N$$

$$P(w_k|c_j, s_m) = \sum_{e=1}^{N} \alpha_{ej} n_{kem} / L(c_j, s_m)$$

where n_{kem} is the count of the number of times word w_k appears in example B_e in slot s_m, and

$$L(c_j, s_m) = \sum_{e=1}^{N} \alpha_{ej} |d_m|$$

denotes the total weighted length of the documents in category c_j and slot s_m.

Slot	Word	Strength
DESCRIPTION	ZUBRIN	9.85
DESCRIPTION	SMOLIN	9.39
DESCRIPTION	TREFIL	8.77
DESCRIPTION	DOT	8.67
SUBJECTS	COMPARATIVE	8.39
AUTHOR	D_GOLDSMITH	8.04
DESCRIPTION	ALH	7.97
DESCRIPTION	MANNED	7.97
RELATED-TITLES	SETTLE	7.91
RELATED-TITLES	CASE	7.91
AUTHOR	R_ZUBRIN	7.63
AUTHOR	R_WAGNER	7.63
AUTHOR	H_MORAVEC	7.63
RELATED-AUTHORS	B_DIGREGORIO	7.63
RELATED-AUTHORS	A_RADFORD	7.63
DESCRIPTION	LEE	7.57
DESCRIPTION	MORAVEC	7.57
DESCRIPTION	WAGNER	7.57
RELATED-TITLES	CONNECTIONIST	7.51
RELATED-TITLES	BELOW	7.51

Table 1: Sample Positive Profile Features

These parameters are "smoothed" using Laplace estimates to avoid zero probability estimates for words that do not appear in the limited training sample by redistributing some of the probability mass to these items using the method recommended in [21]. Finally, calculation with logarithms of probabilities is used to avoid underflow.

The computational complexity of the resulting training (testing) algorithm is linear in the size of the training (testing) data. Empirically, the system is quite efficient. In the experiments on the LIT1 data described below, the current unoptimized Lisp implementation running on a Sun Ultra 1 trained on 20 examples in an average of 0.4 seconds and on 840 examples in an average of 11.5 seconds, and probabilistically categorized new test examples at an average rate of about 200 books per second.

A profile can be partially illustrated by listing the features most indicative of a positive or negative rating. Table 1 presents the top 20 features for a sample profile learned for recommending science books. *Strength* measures how much more likely a word in a slot is to appear in a positively rated book than a negatively rated one, computed as:

$$Strength(w_k, s_j) = log(P(w_k|c_1, s_j)/P(w_k|c_0, s_j))$$

Producing, Explaining, and Revising Recommendations

Once a profile is learned, it is used to predict the preferred ranking of the remaining books based on posterior probability of a positive categorization, and the top-scoring recommendations are presented to the user.

The system also has a limited ability to "explain" its rec-

The Fabric of Reality:
The Science of Parallel Universes- And Its Implications
by David Deutsch recommended because:

Slot	Word	Strength
DESCRIPTION	MULTIVERSE	75.12
DESCRIPTION	UNIVERSES	25.08
DESCRIPTION	REALITY	22.96
DESCRIPTION	UNIVERSE	15.55
DESCRIPTION	QUANTUM	14.54
DESCRIPTION	INTELLECT	13.86
DESCRIPTION	OKAY	13.75
DESCRIPTION	RESERVATIONS	11.56
DESCRIPTION	DENIES	11.56
DESCRIPTION	EVOLUTION	11.02
DESCRIPTION	WORLDS	10.10
DESCRIPTION	SMOLIN	9.39
DESCRIPTION	ONE	8.50
DESCRIPTION	IDEAS	8.35
DESCRIPTION	THEORY	8.28
DESCRIPTION	IDEA	6.96
SUBJECTS	REALITY	6.78
TITLE	PARALLEL	6.76
DESCRIPTION	IMPLY	6.47
DESCRIPTION	GENIUSES	6.47

Table 2: Sample Recommendation Explanation

The word UNIVERSES is positive due to your ratings:

Title	Rating	Count
The Life of the Cosmos	10	15
Before the Beginning : Our Universe and Others	8	7
Unveiling the Edge of Time	10	3
Black Holes : A Traveler's Guide	9	3
The Inflationary Universe	9	2

Table 3: Sample Feature Explanation

ommendations by listing the features that most contributed to its high rank. For example, given the profile illustrated above, LIBRA presented the explanation shown in Table 2. The strength of a cue in this case is multiplied by the number of times it appears in the description in order to fully indicate its influence on the ranking. The positiveness of a feature can in turn be explained by listing the user's training examples that most influenced its strength, as illustrated in Table 3 where "Count" gives the number of times the feature appeared in the description of the rated book.

After reviewing the recommendations (and perhaps disrecommendations), the user may assign his or her 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 [45], this cycle can be repeated several times in order to produce the best results. Also, as new examples are provided, the system can track any change in a user's preferences and alter its recommendations based on the additional information.

Data	Number Exs	Avg. Rating	% Positive ($r > 5$)
LIT1	936	4.19	36.3
LIT2	935	4.53	41.2
MYST	500	7.00	74.4
SCI	500	4.15	31.2
SF	500	3.83	20.0

Table 4: Data Information

	Rating									
Data	1	2	3	4	5	6	7	8	9	10
LIT1	271	78	67	74	106	125	83	70	40	22
LIT2	272	58	72	92	56	75	104	87	77	42
MYST	73	11	7	8	29	46	45	64	66	151
SCI	88	94	62	49	51	53	35	31	16	21
SF	56	119	75	83	67	33	28	21	12	6

Table 5: Data Rating Distributions

EXPERIMENTAL RESULTS

Methodology

Data Collection Several data sets were assembled to evaluate LIBRA. The first two were based on the first 3,061 *adequate-information* titles (books with at least one abstract, review, or customer comment) returned for the subject search "literature fiction." Two separate sets were randomly selected from this dataset, one with 936 books and one with 935, and rated by two different users. These sets will be called LIT1 and LIT2, respectively. The remaining sets were based on all of the adequate-information Amazon titles for "mystery" (7,285 titles), "science" (6,177 titles), and "science fiction" (3,813 titles). From each of these sets, 500 titles were chosen at random and rated by a user (the same user rated both the science and science fiction books). These sets will be called MYST, SCI, and SF, respectively.

In order to present a quantitative picture of performance on a realistic sample; books to be rated where selected at random. However, this means that many books may not have been familiar to the user, in which case, the user was asked to supply a rating based on reviewing the Amazon page describing the book. Table 4 presents some statistics about the data and Table 5 presents the number of books in each rating category. Note that overall the data sets have quite different ratings distributions.

Performance Evaluation To test the system, we performed 10-fold cross-validation, in which each data set is randomly split into 10 equal-size segments and results are averaged over 10 trials, each time leaving a separate segment out for independent testing, and training the system on the remaining data [30]. In order to observe performance given varying amounts of training data, *learning curves* were generated by testing the system after training on increasing subsets of the overall training data. A number of metrics were used to mea-

Data	N	Acc	Rec	Pr	Pr3	Pr10	F	Rt3	Rt10	r_s
LIT1	5	63.5	49.0	50.3	63.3	62.0	46.5	5.87	6.02	0.31
LIT1	10	65.5	51.3	53.3	86.7	76.0	49.7	6.63	6.65	0.35
LIT1	20	73.4	64.8	62.6	86.7	81.0	62.6	7.53	7.20	0.62
LIT1	40	73.9	65.1	63.6	86.7	81.0	63.4	7.40	7.32	0.64
LIT1	100	79.0	70.7	71.1	96.7	86.0	70.5	8.03	7.44	0.69
LIT1	840	79.8	62.8	75.9	96.7	94.0	68.5	8.57	8.03	0.74
LIT2	5	59.0	57.6	52.4	70.0	74.0	53.3	6.80	6.82	0.31
LIT2	10	65.0	64.5	56.7	80.0	82.0	59.2	7.33	7.33	0.48
LIT2	20	69.5	67.2	63.2	93.3	91.0	64.1	8.20	7.84	0.59
LIT2	40	74.3	72.1	68.9	93.3	91.0	69.0	8.53	7.94	0.69
LIT2	100	78.0	78.5	71.2	96.7	94.0	74.4	8.77	8.22	0.72
LIT2	840	80.2	71.9	78.6	100.0	97.0	74.8	9.13	8.48	0.77
MYST	5	73.2	83.4	82.1	86.7	89.0	81.5	8.20	8.40	0.36
MYST	10	75.6	87.9	82.4	90.0	90.0	83.8	8.40	8.34	0.40
MYST	20	81.6	89.3	86.4	96.7	91.0	87.3	8.23	8.43	0.46
MYST	40	85.2	95.4	85.9	96.7	94.0	90.3	8.37	8.52	0.50
MYST	100	86.6	95.2	87.2	93.3	94.0	90.9	8.70	8.69	0.55
MYST	450	85.8	93.2	88.1	96.7	98.0	90.5	8.90	8.97	0.61
SCI	5	62.8	63.8	46.3	73.3	60.0	51.1	6.97	6.17	0.35
SCI	10	67.6	61.9	51.2	80.0	67.0	54.3	7.30	6.32	0.37
SCI	20	75.4	66.0	64.2	96.7	80.0	63.1	8.37	7.03	0.51
SCI	40	79.6	69.5	68.7	93.3	80.0	68.3	8.43	7.23	0.59
SCI	100	81.8	74.4	72.2	93.3	83.0	72.3	8.50	7.29	0.65
SCI	450	85.2	79.1	76.8	93.3	89.0	77.2	8.57	7.71	0.71
SF	5	67.0	38.3	32.9	40.0	29.0	28.2	5.23	4.34	0.02
SF	10	64.6	49.0	28.9	53.3	36.0	31.5	5.83	4.72	0.15
SF	20	71.8	45.8	37.4	66.7	37.0	37.8	6.23	5.04	0.21
SF	40	72.6	58.9	40.1	70.0	43.0	43.0	6.47	5.26	0.39
SF	100	76.4	65.7	46.2	80.0	56.0	52.4	7.00	5.75	0.40
SF	450	79.2	82.2	49.1	90.0	63.0	60.6	7.70	6.26	0.61

Table 6: Summary of Results

sure performance on the novel test data, including:

- *Classification accuracy* (Acc): The percentage of examples correctly classified as positive or negative.
- *Recall* (Rec): The percentage of positive examples classified as positive.
- *Precision* (Pr): The percentage of examples classified as positive that are positive.
- *Precision at Top 3* (Pr3): The percentage of the 3 top ranked examples that are positive.
- *Precision at Top 10* (Pr10): The percentage of the 10 top ranked examples that are positive.
- *F-Measure* (F): A weighted average of precision and recall frequently used in information retrieval: $F = (2 \cdot Pr \cdot Rec)/(Pr + Rec)$
- *Rating of Top 3* (Rt3): The average user rating assigned to the 3 top ranked examples.
- *Rating of Top 10* (Rt10): The average user rating assigned to the 10 top ranked examples.
- *Rank Correlation* (r_s): Spearman's rank correlation coefficient between the system's ranking and that imposed by the users ratings ($-1 \leq r_s \leq 1$); ties are handled using the method recommended by [3].

The top 3 and top 10 metrics are given since many users will be primarily interested in getting a few top-ranked recommendations. Rank correlation gives a good overall picture of how the system's continuous ranking of books agrees with the user's, without requiring that the system actually predict the numerical rating score assigned by the user. A correlation coefficient of 0.3 to 0.6 is generally considered "moderate" and above 0.6 is considered "strong."

Basic Results

The results are summarized in Table 6, where N represents the number of training examples utilized and results are shown for a number of representative points along the learning curve. Overall, the results are quite encouraging even when the system is given relatively small training sets. The SF data set is clearly the most difficult since there are very few highly-rated books.

The "top n" metrics are perhaps the most relevant to many users. Consider precision at top 3, which is fairly consistently in the 90% range after only 20 training examples (the exceptions are LIT1 until 70 examples[1] and SF until 450 examples). Therefore, LIBRA's top recommendations are highly likely to be viewed positively by the user. Note that the "% Positive" column in Table 4 gives the probability that a randomly chosen example from a given data set will be positively rated. Therefore, for every data set, the top 3 and top 10 recommendations are always substantially more likely than random to be rated positively, even after only 5 training examples.

Considering the average rating of the top 3 recommendations, it is fairly consistently above an 8 after only 20 training examples (the exceptions again are LIT1 until 100 examples and SF). For every data set, the top 3 and top 10 recommendations are always rated substantially higher than a randomly selected example (cf. the average rating from Table 4).

Looking at the rank correlation, except for SF, there is at least a moderate correlation ($r_s \geq 0.3$) after only 10 examples, and SF exhibits a moderate correlation after 40 examples. This becomes a strong correlation ($r_s \geq 0.6$) for LIT1 after only 20 examples, for LIT2 after 40 examples, for SCI after 70 examples, for MYST after 300 examples, and for SF after 450 examples.

Results on the Role of Collaborative Content

Since collaborative and content-based approaches to recommending have somewhat complementary strengths and weaknesses, an interesting question that has already attracted some initial attention [5, 6, 17] is whether they can be combined to produce even better results. Since LIBRA exploits content about related authors and titles that Amazon produces using collaborative methods, an interesting question is whether this *collaborative content* actually helps its performance. To examine this issue, we conducted an "ablation" study in which the slots for related authors and related titles were removed from LIBRA's representation of book content. The resulting system, called LIBRA-NR, was compared to the original one using the same 10-fold training and test sets. The statistical significance of any differences in performance between the two systems was evaluated using a 1-tailed paired *t*-test requiring a significance level of $p < 0.05$.

Overall, the results indicate that the use of collaborative content has a significant positive effect. Figures 1, 2, and 3, show sample learning curves for different important metrics for a few data sets. For the LIT1 rank-correlation results shown in Figure 1, there is a consistent, statistically-significant difference in performance from 20 examples onward. For the MYST results on precision at top 10 shown in

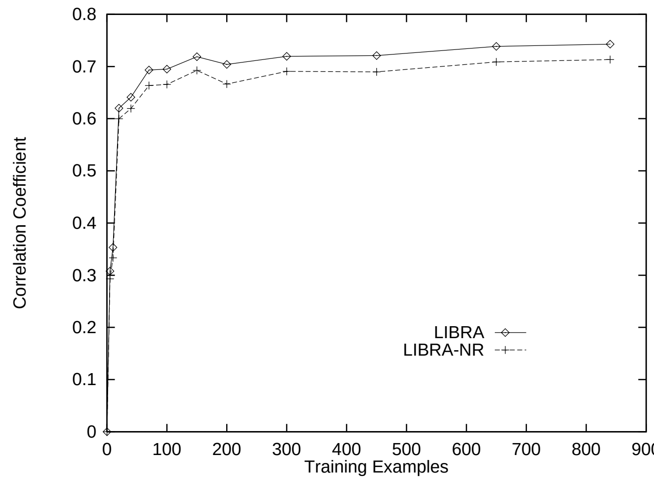

Figure 1: LIT1 **Rank Correlation**

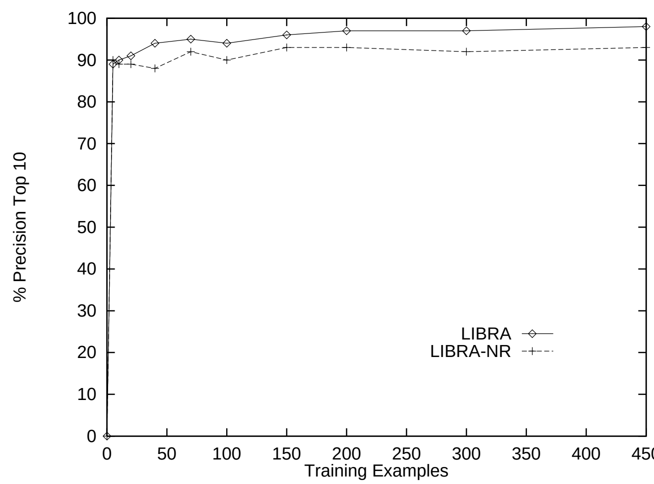

Figure 2: MYST **Precision at Top 10**

Figure 2, there is a consistent, statistically-significant difference in performance from 40 examples onward. For the SF results on average rating of the top 3, there is a statistically-significant difference at 10, 100, 150, 200, and 450 examples. The results shown are some of the most consistent differences for each of these metrics; however, all of the datasets demonstrate some significant advantage of using collaborative content according to one or more metrics. Therefore, information obtained from collaborative methods can be used to improve content-based recommending, even when the actual user data underlying the collaborative method is unavailable due to privacy or proprietary concerns.

FUTURE WORK

Experimental Evaluation and Comparisons

The initial experiments reported in the previous section provide evidence that the current system can produce reasonably accurate recommendations. However, more realistic evaluations with larger numbers of users are needed to fully demonstrate the utility of the approach. Therefore, we are in the process of conducting user studies in which readers train the

[1] References to performance at 70 and 300 examples are based on learning curve data not included in the summary in Table 6.

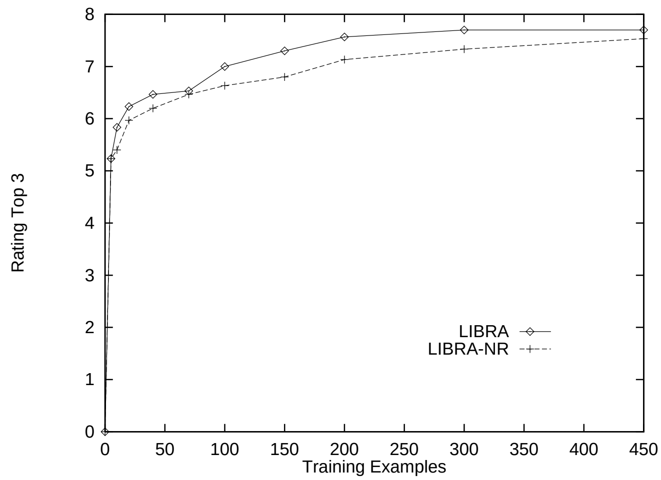

Figure 3: SF **Average Rating of Top 3**

system, select books from the final list of recommendations, and provide informed ratings after actually reading their selected items.

Another planned experiment involves comparing LIBRA's approach to a standard collaborative method. Given the constrained interfaces provided by existing on-line book recommenders, and the inaccessibility of the underlying proprietary user data, conducting a completely controlled experiment using the same training examples and book databases is difficult. However, users can be allowed to use both systems and evaluate and compare their final recommendations.

The examples collected during training in these evaluations can also be used to perform additional cross-validation experiments on predictive accuracy, like those presented in the previous section. In particular, although naive Bayes is known to perform quite well on text categorization tasks, it will be useful to compare it to other learning algorithms such as decision tree induction [38], rule induction [10], neural network learning [44, 32], vector-space methods from information retrieval [43, 45], and other learning methods for text categorization [48, 49].

Improving Learning from Small Training Sets

Since many users are reluctant to rate large number of training examples, various machine-learning techniques for maximizing the utility of small training sets should be utilized. One approach is to use unsupervised learning over unrated book descriptions to improve supervised learning from a smaller number of rated examples. One successful method for doing this in text categorization is based on a statistical training method called EM (Expectation Maximization) [33]. The basic approach is to initially estimate parameters from the supervised data, use this learned model to make predictions for the unsupervised data, label the unsupervised examples as if these predictions were accurate, add them to the training data, and re-estimate the parameters to obtain a new model. This process is iterated until it converges to a local maximum. This approach to using unsupervised data can improve performance by "spreading" the positive (or negative) influence of particular features to other related features which are correlated with it in the unsupervised data. For example, if the user has rated books by Gibson very highly, and in the unsupervised data there are several books that Gibson has co-authored with Sterling, then these books will likely be rated highly by the initial learned model. When these examples are then used to retrain the model, Sterling will also become a positive feature, and his sole-authored books may be recommended. Frequently, this use of correlational information in the unsupervised data can be effective in improving the accuracy of the model.

Another approach to improving learning from small samples is *active learning* or *sample selection*, in which training examples are acquired incrementally and the system attempts to use what it has already learned to select only the most informative new examples for the user to rate [11]. Specific techniques for applying this to text categorization have been developed and shown to significantly reduce the quantity of labeled examples required [25, 26]. One standard approach is *committee-based sampling*, in which a system is given a few initial training examples and learns a set of alternative classifiers called a *committee*. A range of diverse hypotheses for the committee can be constructed by employing different learning methods, different initializations of a randomized algorithm, different subsets of the training data, or different subsets of the features. Next, each of the resulting hypotheses is used to classify all of the unlabeled data and the example that generates the most disagreement amongst the committee members is presented to the user for labeling. The system then retrains on all the currently labeled data and the process repeats until the user is tired of rating examples. By using variance in predicted ranking to measure disagreement, we plan to apply such a technique to automatically select only the most promising training examples for LIBRA.

A slightly different approach is to advise users on easy and productive strategies for selecting good training examples themselves. We have found that one effective approach is to first provide a small number of highly rated examples (which are presumably easy for users to generate), running the system to generate initial recommendations, reviewing the top recommendations for obviously bad items, providing low ratings for these examples, and retraining the system to obtain new recommendations. We intend to conduct experiments on the existing data sets evaluating such strategies for selecting training examples.

Providing a learning system with prior knowledge of the target concept is yet another effective way of improving learning from small samples. Biasing learning with existing knowledge or inductively revising an existing knowledge base has been shown to improve accuracy on a range of real-world problems [34, 47, 36]. LIBRA's current Bayesian learner can be biased with initial knowledge by providing priors for the conditional probabilities ($P(w_k|c_i, s_j)$'s), and using them to bias the corresponding estimates computed from the data. If the user is willing to provide initial information on authors, subjects, or general keywords that are known to be of interest (or disinterest), it could be mapped to priors for the relevant parameters. This approach has already been successfully applied to web-page recommendation [35].

Utilizing Richer Representations of Language Content

The use of unordered bags of words is a very weak (but frequently effective) representation of content. We also plan to explore richer representations and their ability to improve recommendation performance. Specifically, we will examine the use of phrases and syntactic parsing [31, 14] and the use of lexical semantic information such as WordNet [13] to enrich the set of features used to describe items [35, 15].

Integrating Content-Based and Collaborative Methods

Studying additional ways of combining content-based and collaborative recommending is particularly important. The use of collaborative content in LIBRA was found to be useful, and if significant data bases of both user ratings and item content are available, both of these sources of information could contribute to better recommendations [5, 17]. One approach, similar to that introduced in [6], is to include in the representation of each book two additional slots: one for the names of users who liked it and another for the names of users who did not. By also making this slot a *bag* in which the number of times a user's name appears depends on the strength (weakness) of his or her rating, the existing learning algorithm can be allowed to exploit both the sign and magnitude of other users' opinions.

Utilizing Additional Sources of Book Information

Expanding LIBRA to exploit book information beyond that available at Amazon is another area of future research. A first step in this direction is constructing routines that extract book information from other similar sources, such as other on-line booksellers (e.g. BarnesAndNoble, Borders) and on-line library catalogs. A somewhat more ambitious task is extracting book information from less structured sources. For example, the system could query general web search engines such as AltaVista, Yahoo, and DejaNews with the author and title of each known book and attempt to extract useful information about the item from the resulting web pages and newsgroup postings that are retrieved.

Currently, we have only considered recommendations using abstracts and other short descriptions of books. However, the eventual goal is to serve as a recommendation service for digital libraries where complete text is available in electronic form. Since the time complexity of the underlying algorithms is linear, it should be feasible to scale the system to complete text; however, feature selection [50, 32] will likely be required to reduce the size of book descriptions and maintain reasonable memory and CPU requirements.

CONCLUSIONS

The ability to recommend books and other information sources to users based on their general interests rather than specific inquiries will be an important service of digital libraries. Unlike collaborative filtering, content-based recommending holds the promise of being able to effectively recommend unrated items and to provide quality recommendations to users with unique, individual tastes. LIBRA is an initial content-based book recommender which uses a simple Bayesian learning algorithm and information about books extracted from the web to recommend titles based on training examples supplied by an individual user. Initial experiments indicate that this approach efficiently provides accurate recommendations in the absence of any information about other users.

In many ways, collaborative and content-based approaches provide complementary capabilities. Collaborative methods are best at recommending reasonably well-known items to users in a communities of similar tastes when sufficient user data is available but effective content information is not. Content-based methods are best at recommending unpopular items to users with unique tastes when sufficient other user data is unavailable but effective content information is easy to obtain. Consequently, methods for integrating these approaches will perhaps provide the best of both worlds.

Finally, we believe that methods and ideas developed in machine learning research [30] are particularly useful for content-based recommending, filtering, and categorization, as well as for integrating with collaborative approaches [7, 6]. Given the future potential importance of such services to digital libraries, we look forward to an increasing application of machine learning techniques to these challenging problems.

ACKNOWLEDGEMENTS

Thanks to Paul Bennett for contributing ideas, software, and data, and to Tina Bennett for contributing data. This research was partially supported by the National Science Foundation through grant IRI-9704943.

REFERENCES

1. R. B. Allen. User models: Method, theory, and practice. *International Journal of Man-Machine Studies*, 32:511–543, 1990.

2. J. Alspector, A. Kolcz, and N. Karunanithi. Comparing feature-based and clique-based user models for movie selection. In *Proceedings of the Third ACM Conference on Digital Libraries*, pages 11–18, Pittsburgh, PA, June 1998.

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

4. S. Baker. Laying a firm foundation: Administrative support for readers' advisory services. *Collection Building*, 12(3-4):13–18, 1993.

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

6. C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720, Madison, WI, July 1998.

7. D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, Madison, WI, 1998. Morgan Kaufman.

8. M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, Orlando, FL, July 1999.

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

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

11. D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

12. DARPA, editor. *Proceedings of the 6th Message Understanding Conference*, San Mateo, CA, 1995. Morgan Kaufman.

13. C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

14. J. Fürnkranz, T. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization on the WWW. In *Papers from the AAAI 1998 Workshop on Text Categorization*, pages 5–12, Madison, WI, 1998.

15. B. Gelfand, M. Wulfekuler, and W. F. Punch. Automated concept extraction from plain text. In *Papers from the AAAI 1998 Workshop on Text Categorization*, pages 13–17, Madison, WI, 1998.

16. D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the Association for Computing Machinery*, 35(12):61–70, 1992.

17. N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 439–446, Orlando, FL, July 1999.

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

19. H. Kautz, editor. *Papers from the AAAI 1998 Workshop on Recommender Systems*, Madison, WI, 1998. AAAI Press.

20. A. Kent and et al. *Use of Library Materials: The University of Pittsburgh Study*. Dekker, New York, 1979.

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

22. N. Kushmerick, K. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 729–735, Nagoya, Japan, 1997.

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

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

25. D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156, San Francisco, CA, July 1994. Morgan Kaufman.

26. R. Liere and P. Tadepalli. Active learning with committees for text categorization. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 591–596, Providence, RI, July 1997.

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

28. A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *Papers from the AAAI 1998 Workshop on Text Categorization*, pages 41–48, Madison, WI, 1998.

29. K. McCook and G. O. Rolstad, editors. *Developing Readers' Advisory Services: Concepts and Committments*. Neal-Schuman, New York, 1993.

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

31. M. Mitra, C. Buckley, C. Cardie, and A. Singhal. An analysis of statistical and syntactic phrases. In *Proceedings of the 5th RIAO Conference, Computer-Assisted Information Searching on the Internet*, pages 200–214, 1997.

32. H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of 20th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 67–73, Philadelphia, PA, 1997.

33. K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 792–799, Madison, WI, July 1998.

34. D. Ourston and R. J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:311–344, 1994.

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

36. M. Pazzani and D. Kibler. The utility of background knowledge in inductive learning. *Machine Learning*, 9:57–94, 1992.

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

38. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo,CA, 1993.

39. P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Reidl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Computer Supported Cooperative Work Conference*, New York. ACM.

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

41. E. Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

42. E. Rich. Users are individuals: Individualizing user models. *International Journal of Man-Machine Studies*, 18:199–214, 1983.

43. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

44. D. E. Rumelhart, G. E. Hinton, and J. R. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing, Vol. I*, pages 318–362. MIT Press, Cambridge, MA, 1986.

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

46. I. Soboroff, C. Nicholas, and M. Pazzani, editors. *Papers from the SIGIR-99 Recommender Systems Workshop*, Berkeley, CA, 1999. ACM SIGIR.

47. G. G. Towell and J. W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70:119–165, 1994.

48. Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*, May 1999.

49. Y. Yang and X. Liu. A re-examination of text cateogrization methods. In *Proceedings of 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999.

50. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, 1997. Morgan Kaufman.