Text Categorization Through Probabilistic Learning: Applications to Recommender Systems

Paul N. Bennett

CS 379H
Department of Computer Sciences
Plan II Honors Program

University of Texas
Austin, TX 78712

Email : pbennett@cs.utexas.edu

May 4, 1998

Raymond J. Mooney, Ph.D.
Department Of Computer Sciences
Supervising Professor

Risto Miikkulainen, Ph.D.
Department Of Computer Sciences
Second Reader

Abstract

Author: Paul N. Bennett
Title: Text Categorization Through Probabilistic Learning: Applications to Recommender Systems
Supervising Professor: Raymond J. Mooney, Ph.D.

With the growth of the World Wide Web, recommender systems have received an increasing amount of attention. Many recommender systems in use today are based on *collaborative filtering*. This project has focused on LIBRA, a *content-based* book recommending system. By utilizing text categorization methods and the information available for each book, the system determines a user profile which is used as the basis of recommendations made to the user. Instead of the *bag-of-words* approach used in many other statistical text categorization approaches, LIBRA parses each text sample into a semi-structured representation. We have used standard Machine Learning techniques to analyze the performance of several algorithms on this learning task. In addition, we analyze the utility of several methods of feature construction and selection (i.e. methods of choosing the representation of an item that the learning algorithm actually uses). After analyzing the system we conclude that good recommendations are produced after a relatively small number of training examples. We also conclude that the feature selection method tested does not improve the performance of these algorithms in any systematic way, though the results indicate other feature selection methods may prove useful. Feature construction, however, while not providing a large increase in performance with the particular construction methods used here, holds promise of providing performance improvements for the algorithms investigated. This text assumes only minor familiarity with concepts of artificial intelligence and should be readable by the upper division computer science undergraduate familiar with basic concepts of probability theory and set theory.

Contents

1 Introduction

The growth of the World Wide Web has brought to nearly everyone's attention a trend that has been steadily increasing in recent years; namely, a jump in the amount of digital information available, but a lack of effective access to this information, has prompted an increase in development of recommender systems. A recommender system is a system that suggests items (e.g. records, books, news articles, pictures, etc.) of interest to the user (Maes, 1994; Resnik & Varian, 1997). The majority of systems in existence are based on *collaborative filtering.* These collaborative filtering systems make recommendations by matching users with other "like-minded" users where "like-minded" is indicated by correlations among user ratings of items. This approach tends to break down when (a) the system does not know of any similar users for a given user, (b) "marginal" items fail to be rated by enough users, and (c) "new" items cannot be recommended until others have rated them.

In contrast, a *content-based* recommender builds a profile of a user based on the *content* of the items and the user's ratings (Balabanovic & Shoham, 1997). This allows the system to make recommendations to a user based solely on that user's interests. Our prototype system applies text categorization learning methods to items with semi-structured text descriptions in order to make recommendations to the user. Other content-based recommenders that use text categorization have been used to recommend web pages (Pazzani, Muramatsu, & Billsus, 1996) and Usenet messages (Lang, 1995). The system developed for this research is dubbed LIBRA (**L**earning **I**ntelligent **B**ook **R**ecommending **A**gent) and was conceived as an interdisciplinary project between Ray Mooney of the UT-Austin Department of Computer Sciences and Loriene Roy of UT-Austin Library and Information Sciences. Currently LIBRA operates by building a database of books retrieved from web pages at Amazon.com (the current recommender at Amazon.com appears to be based on collaborative filtering). These web pages contain semi-structured text descriptions of books. The system, however, can be applied to any semi-structured samples of text. A user provides an integer-rating in the range of 1 - 10. The system uses these ratings to generate a user profile.

Most other content-based systems that use text categorization do not exploit semi-structured text that may appear in the data. It is common for many text-categorization approaches to use a *bag-of-words* method which represents a text sample as an unordered set of all words appearing in the text (regardless of position). LIBRA, however, uses a simple pattern-based extraction method to identify values for various slots, e.g. Author, ISBN, Price, etc. The book is represented as a group of set-valued features (Cohen, 1996a, 1996b) where each slot is a feature. The value of a slot is the set of words that occur in the value portion of the pattern for the slot. Currently not all of the slots extracted are used for learning. By taking advantage of a semi-structured representation, the system is allowed the latitude to favor more informative types of information (e.g. An author's name occurring in one slot, e.g. "AUTHOR: J.R.R. Tolkien", could theoretically carry more weight than when it appears in a less meaningful slot, "REVIEW: Best books since J.R.R. Tolkien's ...").

Since most recommender systems output a ranked list of samples, it is more important to assess the effectiveness of the ranking produced rather than a simple category prediction (such as "like" or "dislike") or even strict scores. The performance metric used, Spearman's ranked correlation coefficient, actually evaluates the quality of the ranking. It accomplishes this by correlating the ranking of items by user scores to the ranking of items by system predictions.

The prototype system was developed previous to this research—that is, the software tools had already been constructed for: downloading and extracting text from web pages; learning with the Naive Bayes algorithm extended for set-valued features; and learning with the Weighted Binary Naive Bayes algorithm. The current research involved: constructing data sets appropriate for performance evaluation; integrating the system into an external testing system; development of the 10-Ratings learning algorithm (described below); addition of alternate feature construction methods (described below); development of the distinction between rating accuracy and ranking accuracy; investigation and software construction of a testing metric appropriate for evaluating ranking accuracy; and systematic analysis of the learning algorithms' performance, feature construction methods, and feature selection efficacy.

The remainder of the paper is organized as follows: Section 2 provides general background information needed to understand the methods applied in the research and the results reported here; Section 3 gives a complete description of the system; Section 4 details the experimental setting, the results obtained, and a discussion of the results; Section 5 discusses future work that could extend the research, and section 6 summarizes the paper.

2 Background

2.1 Machine Learning

In general, learning is the use of experience in such a way that performance at a certain task is greater with that experience than without it (Mitchell, 1997). *Machine Learning* seeks to construct *systems* that have the ability to automatically generalize from their experience in a manner that increases performance at a given task. However, in order to scientifically analyze a system's performance, we seek to capture certain notions such as what it means for a system's performance to increase, when it can be said that one system *experimentally* outperforms another, and how a difference in performance is judged to be significant. To this end, the field of *Machine Learning* has standard tools to use when assessing a system.

Usually a learning system's experience comes in the form of training examples. Sometimes these examples carry useful tags of information provided by a "teacher" (either the user or another software system) such as a category associated with the example, an underlying concept with which it corresponds, a score, or other information which the learner can possibly make use of during the process of learning (also called training). In other cases, the examples come without any additional information provided by an outside source. The former case is called supervised learning and the latter unsupervised (Russell & Norvig, 1995).

In order to judge the performance of the system, a learning curve is plotted with the number of training examples on the x-axis and some performance metric on the y-axis (where an increase in performance is shown by an increase in the y value). Several features of this curve are indicative of the system's properties. The first of these is the rate at which the curve climbs to a high value. A very steep curve means that the system requires only a small number of values before its ability to generalize from these examples leads to significant performance gains. This is particularly important when obtaining examples is expensive (whether the resource expense is computational, monetary, or user time), as it means the system will "cost" little to perform well. Another major feature of a learning curve is the asymptotic height of the curve on the y-axis; typically this point comes at the maximum number of training examples. This part of the curve is an indicator of the best performance generally obtainable from a system.

In order to compensate for the random possibility that a system performed well because of the particular kind or order of the training examples, the system is typically run n times and the average of the performance criteria over the n times is used in constructing the learning curves. To further ensure that this is not a skewed sampling, the choice of the training examples to use in these n trial runs are usually chosen by randomly partitioning the x number of examples into n subsets with $\frac{x}{n}$ examples in each partition; each subset is used as the "test" examples (the examples that will be used to judge system performance) for exactly one of the trials and the remaining $x - \frac{x}{n}$ are used as the training examples during that trial. Thus each example is used as a test example once and a training example $n - 1$ times. This process is termed n-fold cross-validation (Mitchell, 1997)—the standard choice of n in *Machine Learning* is 10.

As mentioned above, the average of all the trial runs on some metric(s) is used as an indicator of the systems performance. In order to compare two such averages, a standard statistical tool, *Student's t*, is used (Spatz & Johnston, 1984; Mitchell, 1997). *Student's t* (t-test) allows one to judge whether the difference in two means is significant at a certain confidence level, p. A smaller p corresponds to an increased confidence that the difference is significant; typically confidence scores of $p > 0.05$ (also referred to as below 95% confidence) are considered insignificant. When choosing the metric that will later be compared with the t-test, it is important that the metric measure an aspect of the system relevant to the task performed.

2.2 Recommender Systems

A recommender system uses items rated by the user to suggest new items that the user will like. While recommender systems have been used in AI for sometime, their widespread investigation and use has come only recently with the explosion of electronic data which has inundated nearly every computer user. Users seldom have enough time to review all of the information that is available to them, therefore a recommender system can prove extremely useful in prioritizing and filtering the information at which users *will* look.

In order for a system to be *usable* as a recommender system, however, it must not only perform well, but it should possess several other desirable properties. These are: (1) A user should be allowed to change a rating—either to a different value or completely withdrawing the rating. (2) The user should be able to extend the current set of rated examples with new rated examples. (3) A user's request for recommendations from the system would be processed within a reasonable (specific to the task) amount of time. (4) The benefits the system provides should outweigh the cost of training the system. (5) The user's role in training should be integrated seamlessly into the actions they would normally be performing to accomplish the task. Preferably, all of these functions would be performed in real-time; that is, the ideal system would allow the user to interact effortlessly with it.

There are two general categories of recommender systems, *content-based* and *collaborative filtering* systems. A content-based system uses a representation of the item and the profile of the user it has constructed to analyze whether it is likely that the user will like the item based on its content. This is similar to a friend that might recommend an item to you because he knows your preferences and has tried the item himself. A collaborative filtering system uses the ratings of other "similar" users to make a recommendation. These systems often judge similarity by overlap in ratings among other items. This approach is like the friend who will recommend an item to you because other friends, believed to be similar to you, liked the item. Some recommender systems use a hybridization of these two approaches (Balabanovic & Shoham, 1997).

2.3 Text Categorization

Text categorization in general involves assigning a category (or categories) to a sample of text. Sometimes the task is to determine the part-of-speech that a word has in some text; in other cases it may involve classifying a sample into relevant subject categories. Typically the text is represented using some set of features extracted from the sample. This ranges from using each single word as a feature to using logical sentences built to correspond to the meaning of a segment of text.

Given a certain text sample, we would like to be able to determine the most probable category for that sample given all of the information; this category is referred to as the *maximum a posteriori* category (Cat_{MAP}) (Mitchell, 1997). A classifier system that, for each sample, can determine the Cat_{MAP} and chooses the Cat_{MAP} as the predicted class, cannot *on average* be outperformed by another system which uses the *same* information and hypothesis space. A classifier that performs this way is called a *Bayes Optimal Classifier* (Mitchell, 1997).

2.4 Information Extraction

Information Extraction attempts to apply patterns (or templates) to text in order to extract information relevant to certain areas (Lehnert & Sundheim, 1991; Cardie, 1997; Califf & Mooney, 1998). In essence, it attempts to take advantage of certain shallow regularities in language as a means to extract information relevant to certain highly informative fields. For instance, for text describing a conference, we may want to automatically extract features for fields such as date, place, price, etc. Information Extraction can be helpful in breaking down free form text into meaningful segments. For the interested reader, appendices A and B show a sample Amazon page and the information extracted from it by LIBRA, respectively.

2.5 Naive Bayes Algorithm

The Naive Bayes algorithm is a simple algorithm but has performed quite well in most domains (Mitchell, 1997). It is based on Bayes' "inversion" theorem which can be stated as:

Let *Hypothesis* and *Evidence* be any random variables,

$$P(Hypothesis \mid Evidence) = \frac{P(Evidence \mid Hypothesis)P(Hypothesis)}{P(Evidence)} \quad (1)$$

Assuming that we can represent an example, ε, as a conjunction of n features $f_i \in F$ with no loss of information, then in order to construct a *Bayes Optimal Classifier* that chooses the Cat_{MAP} of a classification space, $Category$, of j categories given an example, we have (where $argmax$ returns the index that maximizes the value of its argument):

$$
\begin{aligned}
Cat_{MAP} &= \underset{Cat_k \,\in\, Category}{argmax} \; P(Cat_k|\varepsilon) \\
&= \underset{Cat_k \,\in\, Category}{argmax} \; P(Cat_k|f_1, f_2, \ldots, f_n) \\
&\text{From Bayes}' \text{ Theorem we have,} \\
&= \underset{Cat_k \,\in\, Category}{argmax} \; \frac{P(f_1, f_2, \ldots, f_n|Cat_k)P(Cat_k)}{P(f_1, f_2, \ldots, f_n)} \\
&\text{Since the denominator is independent of the } argmax \text{ index } Cat_k, \\
&\text{i.e. it is simply a normalization term} \\
&= \underset{Cat_k \,\in\, Category}{argmax} \; P(f_1, f_2, \ldots, f_n|Cat_k)P(Cat_k) \qquad (2)
\end{aligned}
$$

The Naive Bayes algorithm makes an approximation to this Bayes Optimal Classifier by assuming that each of the $f_i \in F$ are independent of any subset of $F - \{f_i\}$ given the category value. This allows for the following simplification:

$$
\begin{aligned}
Cat_{MAP} &= \underset{Cat_k \,\in\, Category}{argmax} \; P(f_1, f_2, \ldots, f_n|Cat_k)P(Cat_k) \\
&\text{From the Naive Bayes independence assumption} \\
&= \underset{Cat_k \,\in\, Category}{argmax} \; \prod_{f_i}^{F} [P(f_i|Cat_k)P(Cat_k)] \qquad (3)
\end{aligned}
$$

That the Naive Bayes algorithm is a Bayes Optimal Classifier when the independence assumption holds, is obvious, but what is less obvious is that the Naive Bayes algorithm sometimes performs optimally even when this assumption doesn't hold. This results from the fact that under many less restrictive conditions, even though the probability estimates are strictly incorrect, the actual Cat_{MAP} is still the category with the maximal probability estimate (Domingos & Pazzani, 1996).

2.6 Feature Selection and Feature Construction

It is important to note that we have used the term *features* to refer both to the slots (set-valued features) and when referring to the words that were extracted as the values of the slots. Unless it is explicitly made clear, when we use *word, feature, term,* etc., we are referring to an attribute that is an atom composed of Slot-name.token, e.g. *AUTHOR.Lewis*, *REVIEWS.Lewis*, etc. It is atomic in the sense that, as far as the learning algorithm is concerned there is no connection between *AUTHOR.Lewis* and *REVIEWS.Lewis*. Also, we would like to make it clear that a feature is in no way limited to being simply an actual English word or proper name, etc. concatenated to the slot name. There are various methods to construct features, and a system can apply these to the text occurring in a slot. For example, if a system is using multiple word phrases, then we might have attributes such as *TITLE.(Big Boy)*, *REVIEWS.(Big Boy)*, *TITLE.(Boy Big)*, etc. All of which are distinct attributes. Even using the single word method, the construction of the attributes depends on how the system represents a "single word". For instance, we might treat an author's entire name as a "single word" which might yield, *AUTHOR."C.S.Lewis"*.

There are several problems with our feature representation of examples. The main problem is that the number of features is quite large. There are about 30,000 in each of the two data sets, and this is only with using each word slot pair. If one were to use multiple word phrases or other construction, the number of features becomes much higher. As learning algorithms tend to have a computational complexity proportional to the size of the feature set (Koller & Sahami, 1996), large feature sets can be quite expensive. While Naive Bayes is one of the cheaper algorithms in this respect, in order to apply other more sophisticated algorithms to this task, it would most likely require feature selection in order to be computationally feasible. In addition, the probability distribution function for the probability of a category given an example is often extremely complex in problems with a large feature set (Koller & Sahami, 1996). When data is limited, it is very difficult to accurately estimate the numerous probabilistic parameters needed for this high dimensional space; thus overfitting, estimating parameters that are overly specific to the training set and thus don't generalize well to the test set, is likely (Koller & Sahami, 1996). In addition, the large number of irrelevant and redundant features that tend to be present in large feature spaces often end up misleading the learning system (Koller & Sahami, 1996).

Thus, the general goals of feature selection are more accurate results and reduced running time. Feature selection generally tries to achieve its goals by: (1) reducing the feature set to a smaller but highly informative subset; (2) and constructing higher order features whose higher relevance (will hopefully) more than compensates for adding an additional feature (Yang & Pedersen, 1997). The first we will always refer to as feature selection. The second of these we will refer to as feature construction. One method of constructing higher order features we have tried uses both multiple word phrases (pairs of adjacent words) and single words. Consider that using this method more than doubles the number of features in the domain. It more than doubles because of duplicates being identical as single words. That is, "Black Cat Black Dog" would be represented as [Black Cat Dog] in a single word approach, but as [Black (Black Cat) Cat (Cat Black) (Black Dog) Dog] in a multiple word phrase approach. Thus adding higher order features to the initial feature pool sometimes increases the need for using feature selection to select only a subset to actually use—since the much higher number of features may worsen those problems mentioned above. We experiment with methods of both feature selection and feature construction.

2.7 Information Gain

If we are to perform feature selection over the data sets, intuitively we would like to choose the optimal subset F_G of F such that F_G maintains the original *relevant* information available to us in F. One way to approximate this optimal subset is to choose the n most "informative" features of F according to some criterion for judging how informative a feature is. One standard way to determine the amount of information a feature has is by applying standard ideas of information theory, Entropy and Information Gain. Entropy is a measure of the homogeneity of a data sample. That is, given a data set contains items from j categories, Entropy measures the extent to which these items are dispersed among the j categories. Information Gain measures the reduction in Entropy when given the value of a certain feature. Thus, features that yield a large reduction in Entropy are often helpful in classification because their ability to divide the training set into "purer" subsets than the other features, is a good indicator they will continue to do so over the whole domain. The standard definitions for Entropy and Information Gain are (Mitchell, 1997): Let E be the set of all training examples, $\{c_i\}_{i=1}^{j}$ be the set of j categories in the target space, τ be the set of all attributes, T be an attribute $T \in \tau$, $Values(T)$ be the set of values which T can take on, and $E_{v(T)}$ denote $\{x \in E \mid the\ value\ of\ T\ in\ x\ is\ v, where\ v \in Values(T)\}$. Since it will be clear from context which $T \in \tau$ we mean we will abbreviate $E_{v(T)}$ as E_v. Then Entropy ($Entropy$) and Information Gain ($Gain$) can be defined as:

$$Entropy(E) \equiv \sum_{i=1}^{j} [-P(c_i \mid E) \log_2 P(c_i \mid E)] \tag{4}$$

$$Gain(E, T) \equiv Entropy(E) - \sum_{v \in Values(T)} \left[\frac{\mid E_v \mid}{\mid E \mid} Entropy(E_v) \right] \tag{5}$$

It can easily be seen from the definition how Information Gain measures the reduction in Entropy induced by partitioning a set according to the values of a given attribute.

3 System Description

3.1 Extracting Information and Building a Database

LIBRA currently has accumulated a database of 2,600 science fiction books and a database of 3,061 general literary fiction books. These databases were built by first performing a keyword/subject search and then downloading and parsing the pages corresponding to the resulting URL's from the search. Only the title, authors, synopsis, and subject slots are currently employed in learning; however, values for URL, type, length, price, ISBN, etc. are also extracted.

The values for the slots are extracted with a pattern-based matcher that uses handwritten rules, including pre-filler, filler, and post-filler patterns (Califf & Mooney, 1998), to extract information from the text. The Amazon pages are fairly structured making it easy to design information extraction rules. The values extracted for each slot are stored as an unordered set of words. The collection of these set-valued features make up the complete representation for a book.

3.2 Book Representation

Each slot is treated as a vector of binary features; thus this differs from approaches which consider only the words that occur in a given text sample (Mitchell, 1997; Joachims, 1997). Furthermore, we have applied methods of feature construction and feature selection.

For feature construction, we have tried one type of higher order feature—the multiword method. This makes a feature out of every word that occurs in a slot as in the normal method, and in addition, makes a feature out of every pair of sequential words. This acts as a higher order feature since it is essentially weighting the fact that order and co-occurrence matters.

While Naive Bayes can work efficiently with large dimensional spaces, it may be of help to perform feature selection over the features prior to learning. One reason why this may be helpful is that the sheer space required to store all of the probabilistic estimates can get expensive. We have used the information gain criterion to choose the N most informative features to be used during training and prediction. N was varied to be $500, 1000, 2000, 4000, 8000$, and 16000. However, it must be noted here that because Naive Bayes makes explicit independence assumptions about the terms, it is thought to be less sensitive to changes in context caused by feature selection. Thus, it may make a poor indicator as to the superiority of a given feature selection method (Yang & Pedersen, 1997).

3.3 Learning a Profile

A Naive Bayes (NB) Classifier which has been extended to efficiently deal with set-valued features is used for the text categorization task. The probability estimates are smoothed using Laplace estimates as described in Kohavi, Becker, and Sommerfield (1997). The smoothing includes near-zero estimates for novel words encountered in test samples but not encountered in training examples. The 1 - 10 user rating for an item is treated as the category of that item. So, in order to calculate the posterior probabilities of the categories, the probabilities of a feature given a category (rating) are computed. The probabilities are mapped into a logarithmic space to avoid underflow. In order to be able to efficiently compute a probability estimate for a sample at testing time, the posterior probability of each category given the empty set is precomputed (i.e. the probability of a category given no words occur in the example). These estimates can then be adjusted for the actual words that occur in a sample.

In fact, revising the posterior probabilities estimates is quite easy with the Naive Bayes approach. Naive Bayes (and Bayesian methods in general) estimates can be incrementally updated—both adding and retracting information (Pearl, 1988). Thus, Naive Bayes allows us to achieve several of the criteria for a useful recommender system enumerated in section 2.2. Namely, the user can retract ratings, extend the set of examples rated, and obtain recommendations, all fairly efficiently. Currently the system is not constructed in a way to fully exploit all of the incremental attributes of the algorithm.

3.4 Producing, Explaining and Revising Recommendations

In order to produce recommendations, LIBRA learns a profile, predicts scores for the non-rated samples, and finally ranks the samples by their scores. Currently LIBRA uses one of three methods to learn a user profile. The first method is simply a binary NB classifier. It treats items rated 1 - 5 as negative instances, and those rated 6 - 10 as positive instances. The scores are ranked based on the natural log of the posterior odds of positive, $\ln\left(\frac{P(Positive|Example)}{P(\neg Positive|Example)}\right)$ (Pearl, 1988). A second method treats the 10 ratings as 10 distinct categories. When predicting for a test sample, the system first computes the posterior probability of each category given the test sample. Then the expected value of the posterior probability distribution for the categories is computed and used as the predicted score, $\sum_{i=1}^{10} iP(i)$, 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 $\sum_{i=6}^{10} P(i) > \sum_{i=1}^{5} P(i)$. The final method used is a weighted binary model that 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.

Both the Binary and the Weighted Binary approach have a limited explanatory capability. The explanations consist of the top features that most contributed to the score, e.g.

> 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 given 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 examining the rankings produced by the system, the user can choose examples (the user would probably want to choose ones where there is disagreement with the system) to rate. Then allow the system to use these new ratings to revise its recommendations. As with the use of relevance feedback (Salton & Buckley, 1990), this can be repeated in order to further improve recommendations.

3.5 System Details

This section relates details of the algorithms used that would be necessary for reproducing the results reported here.

3.5.1 Estimating Probabilities

This section relates the estimations and smoothing factors used for estimating the probability parameters needed to make a prediction.

Let E_{Cat_k} denote $\{x \in E \mid \textit{the category of } x \textit{ is } Cat_k, \textit{where } Cat_k \in Category\}$ and assume the notation used earlier in the paper. Then the following are the probability estimates we would use prior to smoothing.

$$P(Cat_k) = \frac{\mid E_{Cat_k} \mid}{\mid E \mid} \tag{6}$$

$$
\begin{aligned}
P(f_i = t \mid Cat_k) &= \frac{P(f_i = t \bigcap Cat_k)}{P(Cat_k)} \\
&\text{By definition of conditional probability,} \\
&= \frac{\frac{|E_t \bigcap E_{Cat_k}|}{|E|}}{\frac{|E_{Cat_k}|}{|E|}} \\
&= \frac{\mid E_t \bigcap E_{Cat_k} \mid}{\mid E_{Cat_k} \mid} \qquad (7)
\end{aligned}
$$

Of course, $P(f_i = \bar{t} \mid Cat_k) = 1 - P(f_i = t \mid Cat_k)$. The weighted binary estimates according to the same estimations above. However, the matter of weighting makes how the form looks in the end slightly different; they can be estimated as follows (where w_r is the function of r mentioned above:

$$
\begin{aligned}
P(Positive) &= \frac{\mid E_{Positive} \mid}{\mid E \mid} \\
&= \frac{\sum_r [w_r \mid E_r \mid]}{\mid E \mid}
\end{aligned}
$$

$$
\begin{aligned}
P(f_i = t \mid Positive) &= \frac{\mid E_t \bigcap E_{Positive} \mid}{\mid E_{Positive} \mid} \\
&= \frac{\sum_r [w_r \mid E_t \bigcap E_r \mid]}{\sum_r [w_r \mid E_r \mid]}
\end{aligned}
$$

In order to estimate for $Negative$ for the weighted binary classifier, simply substitute $1 - w_r$ for w_r. When one of the probabilities in equation 3 is zero, it will dominate the whole computation by making the whole value from going to zero. It could easily happen that the true probability was not zero, but the rarity of the term or category compared to the relatively small number of input examples caused the item not to appear in the training examples. In order to account for these issues, we use smoothing methods reported in (Kohavi et al., 1997). Namely, we use a Laplace-m estimate where m is the number of training examples. In general the Laplace-m estimate is:

For N matches out of n instances for a $k-$valued problem,

$$
\frac{N + \frac{1}{m}}{n + \frac{k}{m}}
$$

So, for our problem we have

$$
P(Cat_k) = \frac{\mid E_{Cat_k} \mid + \frac{1}{|E|}}{\mid E \mid + \frac{|Category|}{|E|}} \qquad (8)
$$

$$P(f_i = t \mid Cat_k) \quad = \quad \frac{\mid E_t \bigcap E_{Cat_k} \mid + \frac{1}{|E|}}{\mid E_{Cat_k} \mid + \frac{2}{|E|}} \tag{9}$$

Notice that both of these estimates approach a uniform distribution as the number of examples goes to zero and they approach the original estimates as the number of examples goes to infinity. Thus the weighting reflects the fact that we are more confident in the original estimates according to how many training examples we have.

Each novel word is given a small estimate as well which is $\frac{1}{|E||E_{Cat_i}|+2}$. Note that this formula weights a novel word toward a less likely *a priori* category. The reason for this is that the more occurrences of a category in the training examples, the less likely it is that a word associated with that category has been left out because of a skewed sample.

3.5.2 Real Valued Scores

In order to produce rankings, we need to obtain values that can be partially ordered. In addition, it is important that these values are normalized probability estimates. That is, we dropped the normalization term during the derivation of equation 2 as it was irrelevant when comparing probabilities for the *same* example since they each were weighted by that same normalization term. However since the normalization term ($P(\varepsilon)$) may differ across several examples, in order to compare probability estimates across examples, we must have comparable values. As mentioned above, we use the natural log of the posterior odds of positive to rank the predictions for the Weighted Binary and Binary Classifier. We show in the following derivation that the normalization factor drops out of the estimation of the posterior odds as follows:

By definition (Pearl, 1988),

$$O(Positive \mid \varepsilon) \quad = \quad \frac{P(Positive \mid \varepsilon)}{P(\neg Positive \mid \varepsilon)}$$

Since $Positive$ and $Negative$ are mutually disjoint exhaustive subsets of the domain,

$$= \quad \frac{P(Positive \mid \varepsilon)}{P(Negative \mid \varepsilon)}$$

Substitution of Bayes Theorem yields,

$$= \quad \frac{\frac{P(\varepsilon|Positive)P(Positive)}{P(\varepsilon)}}{\frac{P(\varepsilon|Negative)P(Negative)}{P(\varepsilon)}}$$

$$= \quad \frac{P(\varepsilon \mid Positive)P(Positive)}{P(\varepsilon \mid Negative)P(Negative)} \tag{10}$$

Note that the numerator and the denominator of the final line in derivation 10 are exactly those needed to compute the Naive Bayes prediction derived in 3. Since the probabilities have already been mapped into a logarithmic space, continuing to work in the logarithmic space is easiest when possible. So, computing the log of the final line of 10 is equivalent to simply subtracting (since they are logs) the estimate used in prediction for negative from that of positive.

In order to rank the predictions for the 10-Category Classifier, we can simply use the expected value as mentioned above. However in order to produce the expected value, we must normalize the 10 different estimates we have at this point of $\ln[P(\varepsilon \mid Cat_i)P(Cat_i)]$. This presents a minor problem. We began working with logarithms in the first place in order to avoid underflow, but we would like to normalize the actual estimates. If we tried to map the logarithms directly back into the exponential space, we would have underflow in some cases (this was experimentally confirmed). However, if we were to add some term μ to each of the logarithmic estimates that enabled us to exponentiate them without underflow, the addition of this factor, since it is equivalent to multiplication in the regular space, would be naturally accounted for in the normalization process. In order to choose an appropriate μ, we simply use the maximum of the logarithmic estimates. Thus, it is guaranteed that we will not have underflow for the Cat_{MAP} estimate, and it is only if a category has magnitudes lower a posteriori probability, that it might underflow. If this were the case, the underflow term's correct value would be so low that the computation of the expected value would have remained relatively unchanged even with the correct value.

3.5.3 Feature Selection

The definition we use for Information Gain is similar to that used in general in text categorization problems for j-category problems (Yang & Pedersen, 1997) and can easily be derived from the standard definitions of Information gain and Entropy as follows:
Assume we have the definitions used for equations 4 and 5. Furthermore, let T be a binary attribute whose values are $\{t, \bar{t}\}$ denoting the presence or absence of a term, respectively. Let $P(c_i \mid D = \sigma)$ denote the probability of randomly choosing $x \in \sigma$ such that the category of x is c_i (D and σ can be thought of as a random variable over the domain of examples and some instantiation of the variable, respectively). We will abbreviate this as $P(c_i \mid \sigma)$. Let $P(t \mid D = \sigma)$ denote the probability of randomly choosing $x \in \sigma$ such that $v(T) = t$, and let $P(\bar{t} \mid D = \sigma)$ be defined similarly. We abbreviate these $P(t \mid \sigma)$ and $P(\bar{t} \mid \sigma)$ respectively.

$$
\begin{aligned}
Gain(E,T) &\equiv Entropy(E) - \sum_{v \in Values(T)} \left[\frac{\mid E_v \mid}{\mid E \mid} Entropy(E_v) \right] \\
&= \sum_{i=1}^{j} [-P(c_i \mid E) \log_2 P(c_i \mid E)] - \frac{\mid E_t \mid}{\mid E \mid} Entropy(E_t) - \frac{\mid E_{\bar{t}} \mid}{\mid E \mid} Entropy(E_{\bar{t}})
\end{aligned}
$$

Substitution of the definition of Entropy yields

$$
\begin{aligned}
&= \sum_{i=1}^{j} [-P(c_i \mid E) \log_2 P(c_i \mid E)] - \frac{\mid E_t \mid}{\mid E \mid} \sum_{i=1}^{j} [-P(c_i \mid E_t) \log_2 P(c_i \mid E_t)] \\
&\quad - \frac{\mid E_{\bar{t}} \mid}{\mid E \mid} \sum_{i=1}^{j} [-P(c_i \mid E_{\bar{t}}) \log_2 P(c_i \mid E_{\bar{t}})]
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{i=1}^{j}[-P(c_i \mid E)\log_2 P(c_i \mid E)] + \frac{\mid E_t \mid}{\mid E \mid}\sum_{i=1}^{j}[P(c_i \mid E_t)\log_2 P(c_i \mid E_t)] \\
&\qquad + \frac{\mid E_{\bar{t}} \mid}{\mid E \mid}\sum_{i=1}^{j}[P(c_i \mid E_{\bar{t}})\log_2 P(c_i \mid E_{\bar{t}})]
\end{aligned}
$$

Since $P(t \mid E) = \frac{\mid E_t \mid}{\mid E \mid}$ and $P(\bar{t} \mid E) = \frac{\mid E_{\bar{t}} \mid}{\mid E \mid}$,

$$
\begin{aligned}
&= \sum_{i=1}^{j}[-P(c_i \mid E)\log_2 P(c_i \mid E)] + P(t \mid E)\sum_{i=1}^{j}[P(c_i \mid E_t)\log_2 P(c_i \mid E_t)] \\
&\qquad + P(\bar{t} \mid E)\sum_{i=1}^{j}[P(c_i \mid E_{\bar{t}})\log_2 P(c_i \mid E_{\bar{t}})] \qquad (11)
\end{aligned}
$$

Though the computation of some of the terms here could be done at the same time as computation of estimates needed for the Naive Bayes algorithm, we have kept them separate in order to preserve abstraction. Furthermore, since the feature selection is completely done as a preprocessing phase, any features that are encountered during testing that have been feature selected out of the set are treated as novel features.

4 Experimental Results

4.1 Methodology

4.1.1 Data Collection

The first 5500 URL's returned from the keyword search "literature fiction" were downloaded from the Amazon web site and parsed into a book representation. During the processing of these 5500 URL's, there were 28 errors. These URL's were discarded. Of the remaining 5,472, 2,409 were classified as *inadequate information* pages and 3,063 were classified as *adequate information* pages. A page was deemed to have *inadequate information* if it did not contain an instance of at least one of the following slots: *comments*, *reviews*, or *synopses*. These pages were written to a separate file for possible use in the future as they do contain information such as title, author, subject, etc. that could be of use to the learning system. In many cases however, these pages lack so much information that it would be difficult for a user who was unfamiliar with the book to rate it and would be likewise pragmatically useless for a similar user of the recommending system. Of the 3,063 *adequate information* pages, two pages duplicated the ISBN's and exact features of other titles present in the data, and as a result, these two were discarded. The remaining 3,061 titles have unique ISBN's and form the corpus of our general fiction database. The database does contain duplicates in the sense that some titles are present as a "hardcover edition" and a "cassette edition"; while LIBRA does possess heuristics for associating related instances, they were left distinct at this time. This was done for two reasons: (1) some users may express regularities in interests such as audio vs. written, but more importantly (2) obtaining user ratings for very similar items will allow the testers to evaluate, to at least some degree, the consistency in user ratings.

Two sets of 1,000 titles were chosen randomly from the 3,061 titles, and one user rated each. The two data sets shared 589 titles in common. Each user used two windows when rating the examples. One window contained the LISP process which prompted for the user's rating; another window was a web browser in which the LISP process loaded the current example to be rated. Thus, both users had access to the full graphical presentation of the page as well as the textual information. Each user was allowed to enter an integer rating from 1 - 10, inclusive. The ranking correlation coefficient between the users' ratings for the overlapping titles was 0.7510047. *Data Set 1* contained 0.642 negative user ratings (i.e. ≤ 5), and *Data Set 2* contained 0.596 negatively rated examples. The distribution over the 1 - 10 ratings is given in Figure 1.

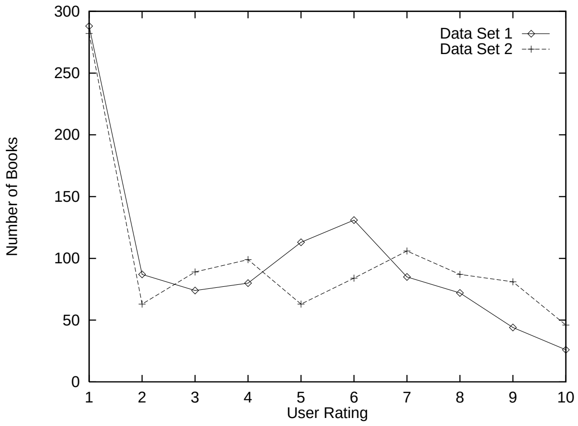

Figure 1: Distribution of User Ratings for Both Data Sets

The textual data obtained from Amazon has many real-world aspects. The users reported that while rating the books, some pages (retrieved directly from Amazon) contained synopses that were clearly intended for other books; there were also spelling errors which a simple approach such as ours would treat as separate instances. In addition, the amount and quality of description of the books tended to vary across a wide range.

4.1.2 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 was 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 be 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. In order to compute r_s when there are ties in the data, the median of the rankings each example in the tie would have been given had they been sequential, is assigned as the rank of each (Anderson & Finn, 1996). Also, the square of the correlation coefficient is interpreted as the percentage of performance change from using the system predictions as an estimate of user preference rather than guessing the mode; in more technical terms, the percentage reduction in the sum of squared deviations when using the system predictions instead of the mode of the data as the basis for prediction (Anderson & Finn, 1996). Since this metric is simply the square of the correlation coefficient, we have not reported it separately. When there are no ties, this reduces to the form given in most introductory statistics texts (Spatz & Johnston, 1984).

4.1.3 Systems and Hypotheses

Our current experiments compare a simple Binary Classifier and a 10-Ratings classifier which uses the expected value to predict ratings (hereafter referred to as Binary and 10-Ratings, respectively). We then compare the performance of the Weighted Binary Classifier (hereafter Weighted Binary) to the first two. We expected that with sufficient training data the 10-Ratings method would outperform the Binary classifier on the rank correlation measure since it exploits a user's 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 method should outperform the Binary method since the Weighted Binary method has at least some representation of the user's real-valued ratings, but it seems reasonable to hypothesize that the 10-Ratings method will outperform the Weighted Binary method since the heuristic employed by the Weighted Binary is probably losing information in its compression of 1–10 ratings into a binary space.

Currently, we have run experiments for two types of feature construction. The first of these, the standard treatment, just treats each word in a slot as a distinct feature, (Singleword). The second method is the method of extracting multiple words as mentioned above, (Multiword). Since the Multiword feature construction method more than slightly doubles the number of features, we expect that the increased noise will negatively effect the performance of the Multiword approach.

We also test feature selection using information gain on all three classifiers as well as both methods of feature construction. In general, we expect that feature selection will lead to an increase in performance because of the reduction in noise terms present in the data and the elimination of spurious correlations. Furthermore, since the Multiword approach contains many more features than the Singleword approach, we expect its performance to increase the most. The expectation is that once the extra noise introduced by the Multiword approach has been disposed of via feature selection, its more expressive model will lead to better performance than the Singleword.

4.2 Results

The graphs below are presented as they are for one of several reasons: (1) the presentation of the graphs follow the development of research, thus making the line of experimentation clearer to the reader; (2) Some of the graphs that could be collapsed into one graph were separated in order to lessen the clutter of the graphs. Figures 2 and 3 show the results for running the Binary and 10-Ratings systems on *Data Set 1*. The results for running the Binary and 10-Ratings systems on

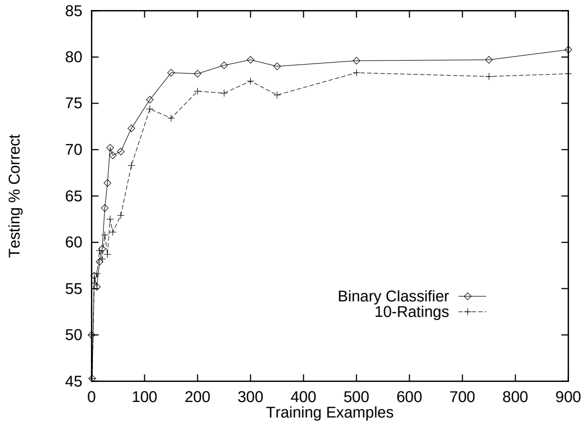

Figure 2: Binary Prediction Accuracy for Data Set 1

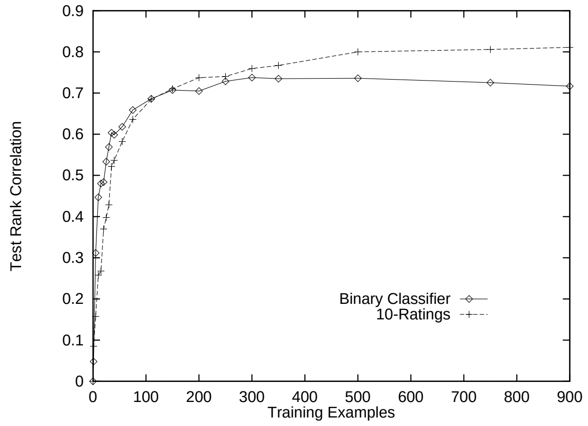

Figure 3: Rank Correlation Coefficient for Data Set 1

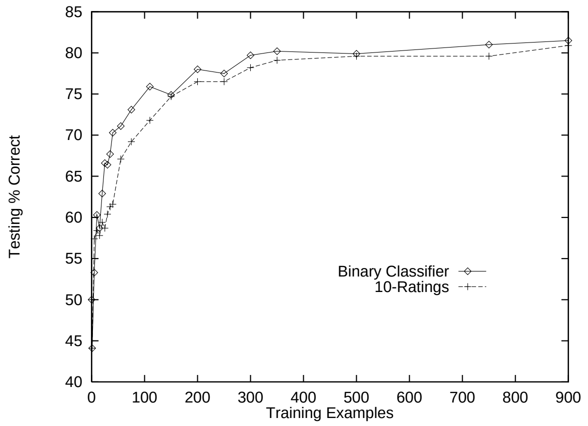

Figure 4: Binary Prediction Accuracy for Data Set 2

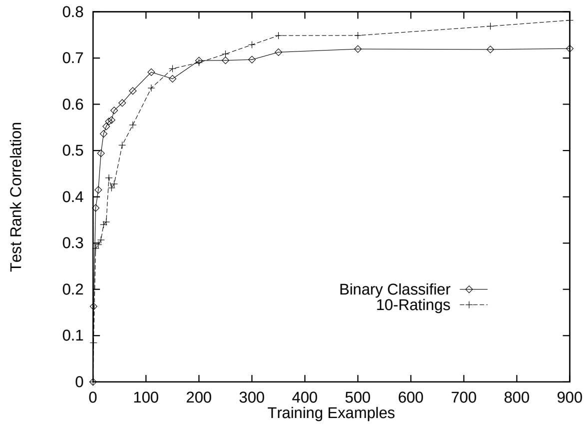

Figure 5: Rank Correlation Coefficient for Data Set 2

Data Set 2 are given in figures 4 and 5. Figures 6 and 7 show the results for running all the systems on *Data Set 1* while those for *Data Set 2* are displayed in figures 8 and 9. Overall, the predictions are reasonably accurate even given relatively small training sets. A correlation coefficient of 0.3 to 0.6 is generally considered “moderate” and above 0.6 is considered “strong.” Therefore, moderate correlations are produced after about 20 examples and strong correlations 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 sta-

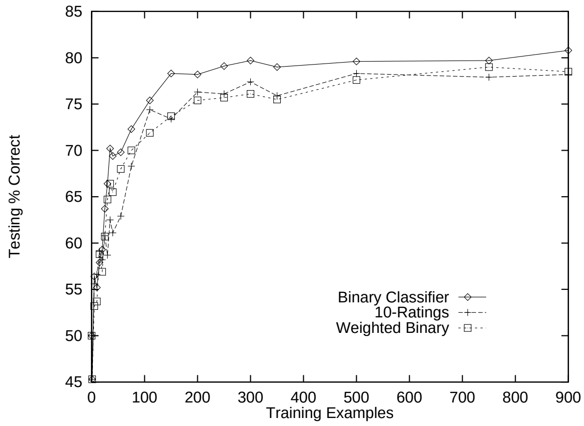

Figure 6: Binary Prediction Accuracy for Data Set 1 - All Systems

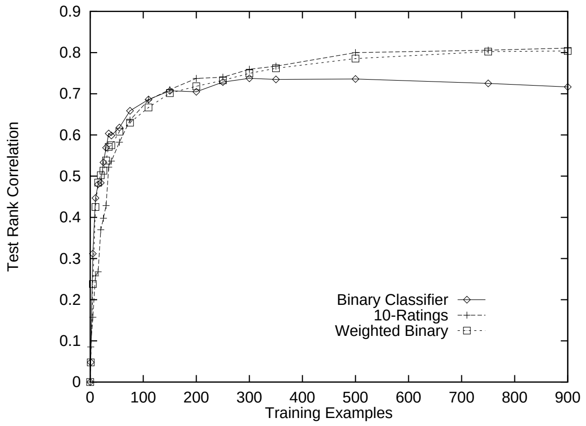

Figure 7: Rank Correlation Coefficient for Data Set 1 - All Systems

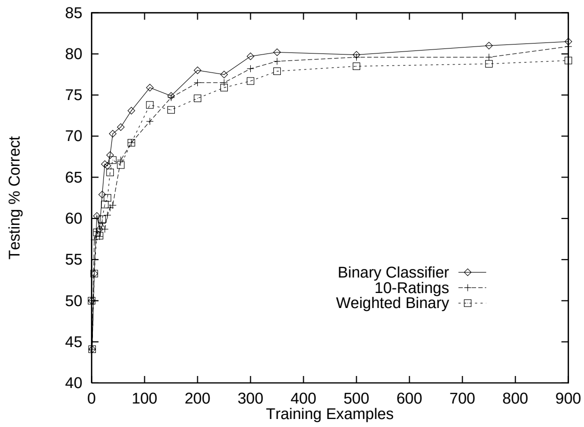

Figure 8: Binary Prediction Accuracy for Data Set 2 - All Systems

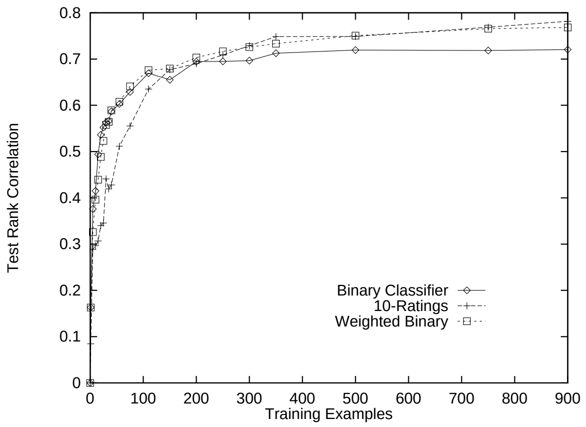

Figure 9: Rank Correlation Coefficient for Data Set 2 - All Systems

tistically 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 (this can be clearly seen on figures 3 and 5), 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 Binary and the 10-Ratings model is not significant.

The results for running each system with information gain based feature selection are given as follows: results for both data sets for the Binary Classifier are given in figures 10, 11, 12, and 13; the results for both data sets using the 10-Ratings Classifier and for the Weighted Binary classifier are not presented here as the results obtained do not differ significantly from those obtained for the Binary Classifier. The numbers in the legends represent how many features were used. Note that

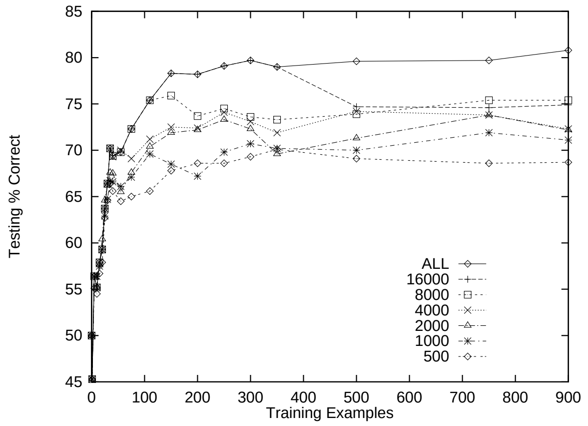

Figure 10: Binary Prediction Accuracy for Data Set 1 - Feature Selection using Information Gain with Binary Classifier

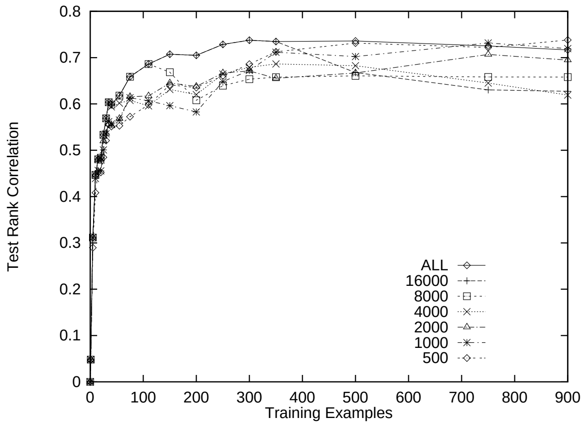

Figure 11: Rank Correlation Coefficient for Data Set 1 - Feature Selection using Information Gain with Binary Classifier

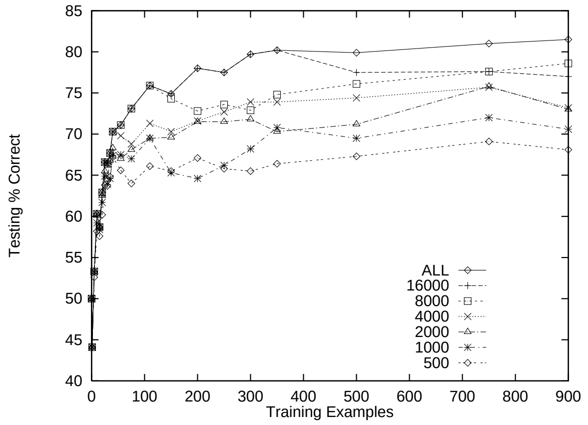

Figure 12: Binary Prediction Accuracy for Data Set 2 - Feature Selection using Information Gain with Binary Classifier

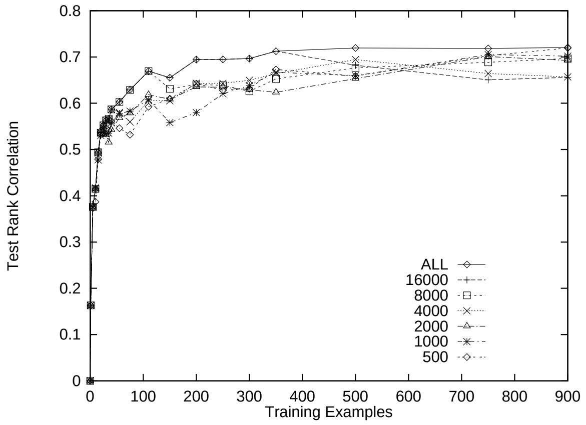

Figure 13: Rank Correlation Coefficient for Data Set 2 - Feature Selection using Information Gain with Binary Classifier

in all of the feature selection results, the less features that were used the poorer the performance that resulted. Although all of the graphs do not strictly show this type of correlation, they are all consistent in one feature. Namely, each feature selection curve follows the curve with all features until, it diverges. In addition, in all of the graphs, the fewer features that are being used the earlier in the learning curve the divergence occurs.

Figures 14, 15, 16, 17, 18, and 19, compare the results of running the three systems on *Data Set 1* with Singleword and Multiword feature construction. The results for *Data Set 2* were similar [1] and so are not presented. While most of the same simulations were performed for feature selection using Multiword feature construction as for Singleword, those results are not presented here since the results were fairly similar.

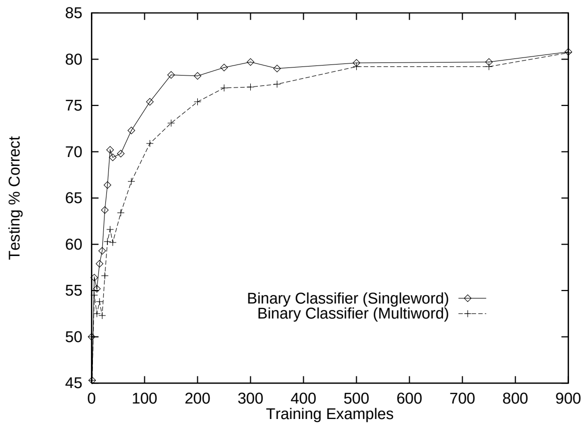

Figure 14: Binary Prediction Accuracy for Data Set 1 - Feature Construction Comparison for Binary Classifier

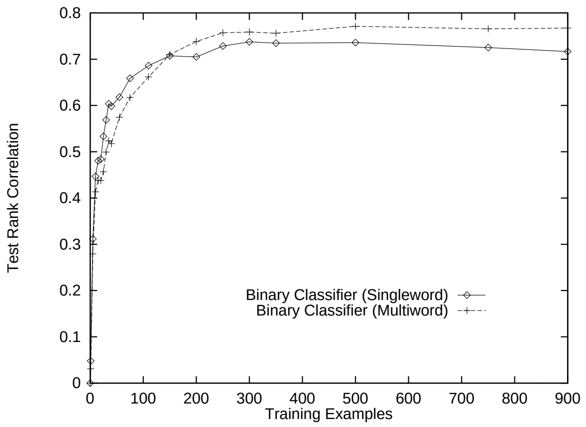

Figure 15: Rank Correlation Coefficient for Data Set 1 - Feature Construction Comparison for Binary Classifier

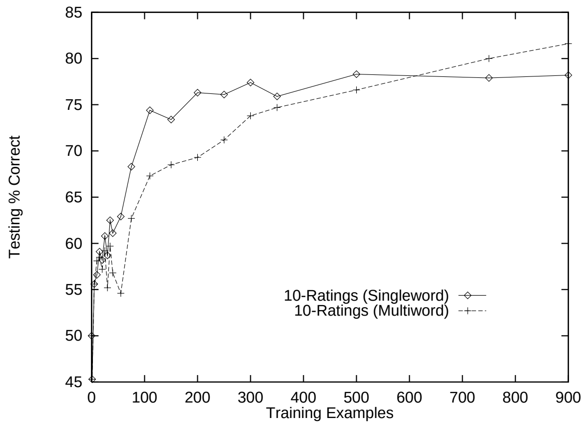

Figure 16: Binary Prediction Accuracy for Data Set 1 - Feature Construction Comparison for 10-Ratings Classifier

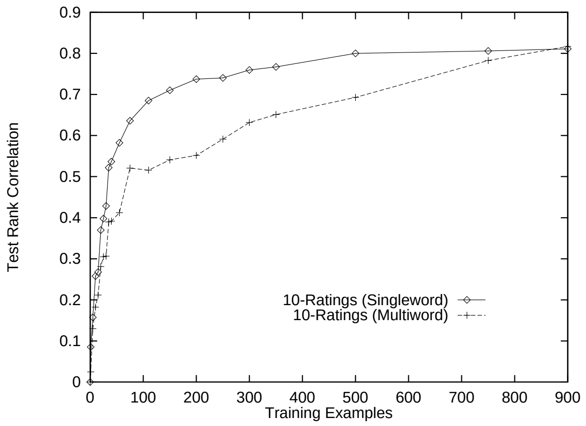

Figure 17: Rank Correlation Coefficient for Data Set 1 - Feature Construction Comparison for 10-Ratings Classifier

[1] Results for feature construction on *Data Set 2* using the 10-Ratings classifier were not available at the time, so we have presented *Data Set 1*.

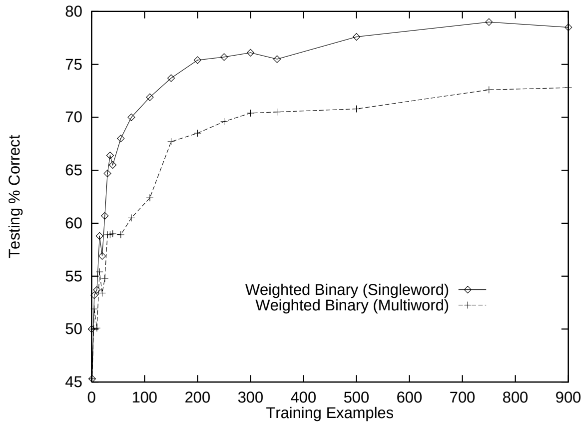

Figure 18: Binary Prediction Accuracy for Data Set 1 - Feature Construction Comparison for Weighted Binary Classifier

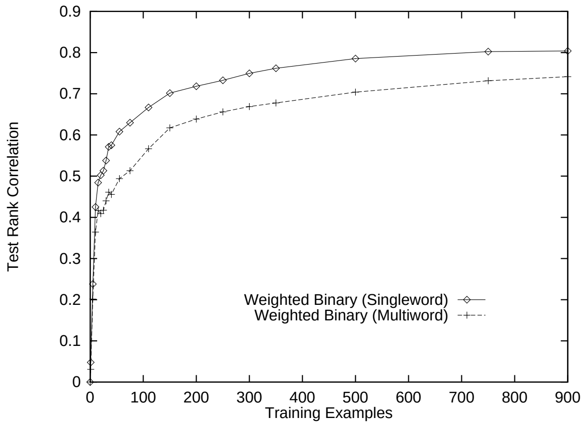

Figure 19: Rank Correlation Coefficient for Data Set 1 - Feature Construction Comparison for Weighted Binary Classifier

For the Binary Classifier the difference in Binary prediction accuracy using Singleword and Multiword feature construction was not significant, but according to the rank correlation coefficient, the Multiword feature construction did outperform the Singleword method (significant at the 0.001 level) using the Binary Classifier. For the Weighted Binary Classifier, using Singleword feature construction outperformed the Multiword feature construction for both performance metrics (significant at the 0.001 level). While the Multiword method outperformed the Singleword method using the 10-Ratings Classifier according to both metrics, the difference is only significant for binary prediction (0.05 level). Though the Multiword feature construction method did about as good (no significant difference) or better than Singleword for both the Binary and 10-Ratings Classifier in the limit, one should note that the Multiword method learns more slowly initially.

4.3 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

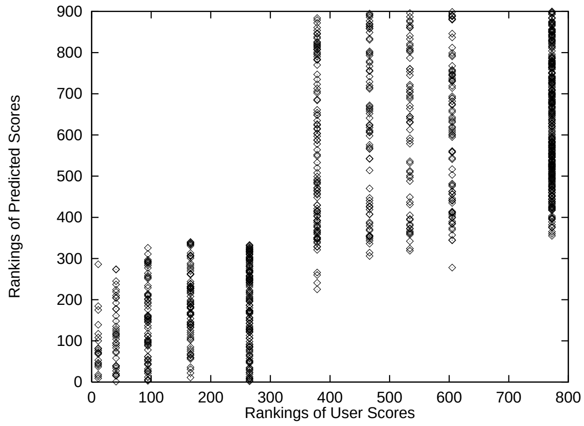

Figure 20: Scatter plot of Ranking for Prediction on the Training Data for One of the Ten Trial Runs Over Data Set 1 Using Binary Classifier (900 training examples)

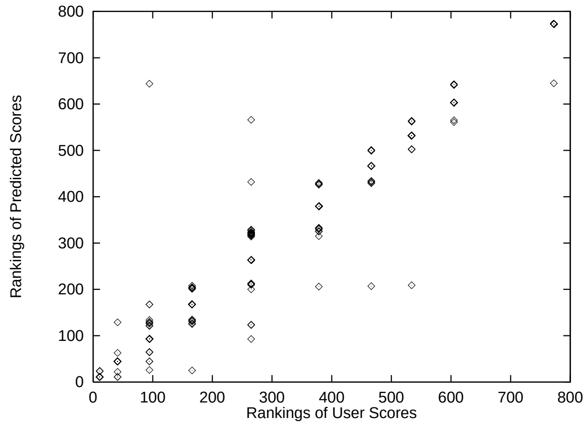

Figure 21: Scatter plot of Ranking for Prediction on The Training Data for One of the Ten Trial Runs Over Data Set 1 Using 10-Ratings Classifier (900 training examples)

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 20, 21, and 22), 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 difference in the rank correlation coefficients over the test examples.

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. This does not support our earlier hypothesis that the Weighted Binary model will lose information by compressing the 1 - 10 ratings into a binary model.

The scatter plots also raise another point of interest (specifically figures 21 and 22). Note that there appear to be far fewer datapoints in figure 21. 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 21. Note that most of the users' ratings were ties since they had to choose an integer from $1 - 10$. Thus, the 10-Ratings model actually

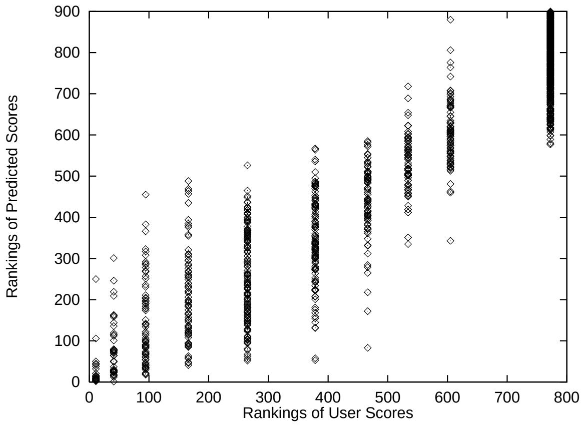

Figure 22: Scatter plot of Ranking for Prediction on the Training Data for One of the Ten Trial Runs Over Data Set 1 Using Weighted Binary Classifier (900 training examples)

has enough parameters to predict ties for scores that the user had as tied. There is a tendency for systems with a larger number of parameters to overfit the training data, and this is strong evidence that the 10-Ratings model is severely overfitting the training data. Note that the points in figure 22 are more spread out. By compressing the information, the Weighted Binary model has actually avoided overfitting the training data.

Feature selection has not led to improvements whatsoever. In all of the figures for feature selection, feature selection only led to degraded results. However, as pointed out above, all of the graphs are consistent in one feature; the fewer features that are used, the earlier in the learning curve that the curve diverges from the curve of the system using all features. This behavior is easily explainable. Consider that we were going to use feature selection to try to select some subset of features that approximate the relevant features in the entire set of features. If our approximation is slightly incorrect, the more training examples (or information) that become available, the more our approximation diverges from the actual distribution. Thus, the point at which the divergence occurs indicates the number of training examples up to which it may be useful to use feature selection—without our approximation diverging from the actual distribution by too great of a margin. As we decrease the number of features, we increase the degree to which our approximation is incorrect.

Finally, the results for feature construction are somewhat mixed. The Multiword feature construction method does not perform well at all using the Weighted Binary classifier, but using both the 10-Ratings and Binary Classifier, better (or not significantly different) results are obtained in the end by using it. Similar to the results above which compare the 10-Ratings Classifier and the Binary Classifier, the much slower rate of improvement in the initial part of the learning curve for the Multiword feature construction is attributable to the higher number of parameters that the Multiword method has. For smaller training sets, there is not enough information to accurately estimate the value of these parameters correctly, and thus, performance is low initially. This raises a point of some interest. *Document Frequency* (DF) is a score assigned to a word based on the number of documents that it occurs in (Yang & Pedersen, 1997). This score, though often viewed as *ad hoc*, has performed well when used for feature selection (Yang & Pedersen, 1997). In this light, we can view DF as a somewhat principled method since it is partially a confidence estimate in the accuracy of a parameter's predicted value. That is, (similar to the discussion of smoothing above) since the number of documents is at least as large as the number of documents that contain any given term, then as the number of documents a term occurs in increases, we know two things: (a) we have more information (examples) in general and can thus be more confident in our estimate; (b) learning the value of this parameter in 1 more example will effect our estimate very little. This suggests that a feature selection criterion that measures the proportion change in a parameter given X more examples with a term versus X more examples without a term may perform well. X could be chosen based on some measure of the average information supplied by an example. This method could easily be designed to be task-free (independent of an example's category), like DF (Yang & Pedersen, 1997), thus allowing feature selection to be statically done, independent of any single user's ratings.

5 Future Work

Future improvements to the system include improving the user interface. Currently most interaction is done via a LISP listener. Making the system completely browser based would greatly improve usability for the common user. It would also be interesting to compare the performance of other algorithms over this data. In order to do so, aggressive feature selection will most likely play a large role since many other learning systems cannot handle the high dimensionality of this domain. It would be interesting to investigate the feature selection method suggested above that selects features based on a confidence rating of the parameter's estimate. Extensions could be made for the system to combine information from other sources; in general or upon detecting that a certain example has low information content, an agent could be sent in search of information over the web. Furthermore, the use of initial user profiles (Pazzani & Billsus, 1997) could provide a boost in the early part of the learning curve. A comparison of the performance of the bag-of-words approach to the set-valued feature approach would be useful in evaluating the gain from using the information extraction methods. It might also be of interest to evaluate a 10 category method that simply chooses the most likely category and completely ignores the continuity of ratings. The system can be extended to use efficient methods of incremental updating as well. Finally, combinations of collaborative filtering with the current system could be investigated through such methods as extending one user's set of training examples with a similar user's set.

6 Conclusions

In the above we have shown the applicability of a more meaningful metric for recommender systems. Using that metric, we have demonstrated that we can obtain good rankings using only binary ratings. When we have access to enough 1 - 10 ratings, we have shown that we can obtain better results in the limit by exploiting the continuity of user scores with a classifier based on the expected value of the probability distribution generated by a ten-category Bayesian classifier. By combining the features of the two different systems into a weighted binary system (i.e. less parameters but retaining the meaningfulness of 1 - 10 ratings), we have shown that we can produce fairly accurate recommendations with only a small number of 1 - 10 ratings. In addition, we have shown that feature selection using Information Gain does not lead to a performance increase in systems using Naive Bayes algorithms. The feature construction methods tested here do show that a gain in performance can be obtained by using higher order features. We feel that feature construction methods hold the most potential for future improvement since they fundamentally change the domain in which the learner is operating.

Finally, we feel the system, in general, has demonstrated a level of performance and possesses enough other characters to be deemed a *usable* recommender system (as discussed in section 2.2). Its potential for assisting a user seems fairly powerful. Furthermore, all of the methods discussed here can be applied to other text based recommender systems.

7 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. The prototype system referred to as constructed prior to this research was written by Ray Mooney.

A Sample Amazon Page

Page Layout: First page is in top left corner. Second page is in top right corner. Third page is in bottom left corner.

Search | Browse Subjects | Bestsellers | Recommendation Center | Gift Center | Kids | Reviewed in the Media | Award Winners | Shopping Cart

amazon.com

Text-Only

The Invisible Man (World's Classics)
by H. G. Wells, David J. Lake, John Sutherland, Patrick Parrinder

List: $6.95
Our Price: $5.56
You Save: $1.39 (20%)

Add it to your Shopping Cart
(You can always remove it later...)

Availability: On Order; usually ships within 1-2 weeks.

Learn more about 1-Click[SM] ordering

Paperback
Published by Oxford Univ Pr (Trade)
Publication date: April 1996
Dimensions (in inches): 0.39 x 7.36 x 4.65
ISBN: 019283195X

Check out these titles! Readers who bought *The Invisible Man (World's Classics)* also bought:

- The Island of Dr. Moreau (Dover Thrift Editions); H. G. Wells, et al
- The War of the Worlds; H. G. Wells
- The Time Machine; H. G. Wells

Browse other Science Fiction & Fantasy titles.

Reviews and Commentary for *The Invisible Man (World's Classics)*

Have you read this book? Write an online review and share your thoughts with other readers.

Book Description :
"At last only the dead tips of the fingernails remained, pallid and white, and the brown stain of some acid upon my fingers. I was almost invisible..." In this horrific tale of man's toying with science and nature, an obscure scientist invents a formula that renders his flesh invisible. Now he can go anywhere, and do anything-except that can no longer render himself visible again-and he has gone murderously insane. When he enters the village pub on a wintery day, wrapped from head to foot, the invisible man at first presents a comic image. But as the villagers discover the truth, they turn on him in horror, as his malice and invisibilty prove a lethal combination. Only in death can he become visible-and harmless-once again. First published by in 1897, H. G. Wells's imaginative insights into human psychology enabled him to evoke in this novel perhaps the ultimate alienation that can befall a human being. --*This text refers to the cassette edition of this title.*

The Merriam-Webster Encyclopedia of Literature , 04/01/95:
Science-fiction novel by H.G. Wells, published in 1897. The story concerns the life and death of a scientist named Griffin who has gone mad. Having learned how to make himself invisible, Griffin begins to use his invisibility for nefarious purposes, including murder. When he is finally killed, his body becomes visible again. --*This text refers to the cassette edition of this title.*

Synopsis:
A strange man, wrapped up from head to foot, arrives at a village on a wintry day. His readiness to pay wins him the quick attention of the hostess, but from that moment on nothing about him bespeaks business as usual. Perhaps the ultimate metaphor for alienation, the invisible man presents at first a comic image. As the villagers discover the truth behind the clownish trappings, however, their jocularity turns to horror in Wells' brilliantly imaginative insight into human psychology. 4 cassettes. --*This text refers to the audio cassettes edition of this title.*

Synopsis:
"At last only the dead tips of the fingernails remained, palid and white, and the brown stain of some acid upon my fingers. I was almost invisible. . . ." Wells' horrific tale of one man's toying with science and nature. 4 cassettes. --*This text refers to the audio cassettes edition of this title.*

Synopsis:
When the dark stranger enters the village pub, muffled up to the chin, with hat, dark glasses and bandaged head, the narrow-minded locals are intimidated and suspicious. The mysterious man is a brash young scientist named Griffin who, in the course of his own experiments on himself, has become invisible--and criminally insane. Hounded out of town by villagers who discover his dark secret, Griffin's malice and invisibility prove a lethal combination. 2 cassettes. --*This text refers to the cassette edition of this title.*

Customer Comments
A reader from Colorado Springs, CO , 10/20/97, rating=8:
A book with an excellant plot that captivates throughout.
The Invisible Man is a classic science fiction novel, having an incredibly strong plot and being written in a way that utilizes it to the utmost. Through suspense and conclusion it keeps you thinking and, of course, wanting to complete it. It was hard to put down to go to bed. --*This text refers to the cassette edition of this title.*

A reader, 09/28/97, rating=10:
Very detailed and frighteningly realistic
Griffin was one of the most graphic book characters I have ever read about. H. G. Wells makes this novel put you in suspense by just turning the first page. --*This text refers to the paperback edition of this title.*

A reader, 04/12/97, rating=5:
This book was intersting!
This story was about an interesting topic. It was cool that the man turned himself invisible. I thought that if you were invisible, you could go through walls, doors, etc. but when I read this story, I learned that you can't. But the story of this book was kinda boring. I think the author should have written this story more vividly so I could of liked this book better. Steven Yum, ASIJ --*This text refers to the paperback edition of this title.*

Look for similar books by subject:

Science fiction
Wells, H. G. (Herbert George), 1866-1946
Literature - Classics / Criticism
Fiction
Find books matching ALL checked subjects
i.e., each book must be in subject 1 AND subject 2 AND ...

- I have read this book, and I want to review it.
- I am the Author and I want to comment on my book.
- I am the Publisher and I want to comment on this book.

Search Books

Search Our 2.5-Million-Title Catalog
Enter Keywords:
Search
Other ways to search our catalog

Or Browse by Subject
Pick a subject. Press go.
Business & Investing
Go
View all subjects

Search Music

You can search for--and buy--CDs right now, but this is only the beginning. While you're exploring, be sure to tell us what you'd like to see in *your* dream music store. Then stay tuned for our grand opening...

Enter Artist or CD Title:
Search
Other ways to search for music

Looking for classical? We're not selling classical music just yet, but we can notify you as soon as we start.

Top of Page | Home | About Amazon.com | Need Help? | Your Account | 1-Click Settings | Shopping Cart

B LIBRA Extraction from Sample Amazon Page

Page Layout: First page is in top left corner. Second page is in top right corner. Third page is in bottom left corner.

Sample LIBRA Book Representation

- TITLE: The Invisible Man (World's Classics)
- URL: /exec/obidos/ISBN%3D019283195X/002-3998572-8395206
- AUTHOR:
  - H. G. Wells
  - David J. Lake
  - John Sutherland
  - Patrick Parrinder
- AUTHOR-URL:
  - /exec/obidos/Author=Wells%2C%20H.%20G./002-3998572-8395206
  - /exec/obidos/Author=Lake%2C%20David%20J./002-3998572-8395206
  - /exec/obidos/Author=Sutherland%2C%20John/002-3998572-8395206
  - /exec/obidos/Author=Parrinder%2C%20Patrick/002-3998572-8395206
- TYPE: Paperback
- LENGTH: NIL
- LIST-PRICE: NIL
- PRICE: NIL
- PUBLISER: Oxford Univ Pr (Trade)
- PUBLICATION-DATE:
- DIMENSIONS: 0.39 x 7.36 x 4.65
- ISBN: 019283195X
- RELATED-BOOKS
  - RELATED-BOOK:
    - TITLE: The Time Machine
    - AUTHOR: H. G. Wells
    - URL: /exec/obidos/ISBN=0812505042/r/002-3998572-8395206
  - RELATED-BOOK:
    - TITLE: The War of the Worlds
    - AUTHOR: H. G. Wells
    - URL: /exec/obidos/ISBN=0553213385/r/002-3998572-8395206
  - RELATED-BOOK:
    - TITLE: The Island of Dr. Moreau (Dover Thrift Editions
    - AUTHOR: H. G. Wells, et al
    - URL: /exec/obidos/ISBN=0486290271/r/002-3998572-8395206
- COMMENTS:
  - The Merriam-Webster Encyclopedia of Literature , 04/01/95:
    - Science-fiction novel by H.G. Wells, published in 1897. The story concerns the life and death of a scientist named Griffin who has gone mad. Having learned how to make himself invisible, Griffin begins to use his invisibility for nefarious purposes, including murder. When he is finally killed, his body becomes visible again.
  - Book Description :
    - "At last only the dead tips of the fingernails remained, pallid and white, and the brown stain of some acid upon my fingers. I was almost invisible..." In this horrific tale of man's toying with science and nature, an obscure scientist invents a formula that renders his flesh invisible. Now he can go anywhere, and do anything-except that can no longer render himself visible again-and he has gone murderously insane. When he enters the village pub on a wintery day, wrapped from head to foot, the invisible man at first presents a comic image. But as the villagers discover the truth, they turn on him in horror, as his malice and invisibilty prove a lethal combination. Only in death can he become visible-and harmless-once again. First published by in 1897, H. G. Wells's imaginative insights into human psychology enabled him to evoke in this novel perhaps the ultimate alienation that can befall a human being.
- SYNOPSES:
  - "When the dark stranger enters the village pub, muffled up to the chin, with hat, dark glasses and bandaged head, the narrow-minded locals are intimidated and suspicious. The mysterious man is a brash young scientist named Griffin who, in the course of his own experiments on himself, has become invisible--and criminally insane. Hounded out of town by villagers who discover his dark secret, Griffin's malice and invisibility prove a lethal combination. 2 cassettes."
  - "At last only the dead tips of the fingernails remained, palid and white, and the brown stain of some acid upon my fingers. I was almost invisible. . . ." Wells' horrific tale of one man's toying with science and nature. 4 cassettes."
  - "A strange man, wrapped up from head to foot, arrives at a village on a wintry day. His readiness to pay wins him the quick attention of the hostess, but from that moment on nothing about him bespeaks business as usual. Perhaps the ultimate metaphor for alienation, the invisible man presents at first a comic image. As the villagers discover the truth behind the clownish trappings, however, their jocularity turns to horror in Wells' brilliantly imaginative insight into human psychology. 4 cassettes."
- REVIEWS:
  - REVIEW:
    - AUTHOR: A reader
    - DATE: 04/12/97
    - RATING: 5
    - TEXT: This book was intersting! This story was about an interesting topic. It was cool that the man turned himself invisible. I thought that if you were invisible, you could go through walls, doors, etc. but when I read this story, I learned that you can't. But the story of this book was kinda boring. I think the author should have written this story more vividly so I could of liked this book better. Steven Yum, ASIJ
  - REVIEW:
    - AUTHOR: A reader
    - DATE: 09/28/97
    - RATING: 10
    - TEXT: Very detailed and frighteningly realistic Griffin was one of the most graphic book characters I have ever read about. H. G. Wells makes this novel put you in suspense by just turning the first page.
  - REVIEW:
    - AUTHOR: A reader from Colorado Springs
    - DATE: CO
    - RATING: 8
    - TEXT: A book with an excellant plot that captivates throughout. The Invisible Man is a classic science fiction novel, having an incredibly strong plot and being written in a way that utilizes it to the utmost. Through suspense and conclusion it keeps you thinking and, of course, wanting to complete it. It was hard to put down to go to bed.
- AVERAGE-RATING: 7.6666665
- SUBJECTS:
  - Fiction
  - Literature - Classics / Criticism
  - Wells, H. G. (Herbert George), 1866-1946
  - Science fiction
- VERSIONS: NIL
- PROPERTIES: NIL

References

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

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

Califf, M. E., & 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* 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*, pp. 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*, pp. 709–716 Portland, OR.

Domingos, P., & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *The Thirteenth International Conference on Machine Learning*.

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*, pp. 143–151 San Francisco, CA. Morgan Kaufman.

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

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. In *The Thirteenth International Conference on Machine Learning*.

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

Lehnert, W., & 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*. McGraw-Hill, New York, NY.

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

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

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Revised Second Printing edition). Morgan Kaufmann Publishers, Inc., San Francisco, CA.

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

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach* (First edition). Prentice Hall, Upper Saddle River, NJ.

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

Spatz, C., & Johnston, J. O. (1984). *Basic Statistics, Tales of Distributions* (Third edition). Wadsworth, Inc., Belmont, CA.

Yang, Y., & Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. In *The Fourteenth International Conference on Machine Learning.*

Author Biography

Paul N. Bennett is the son of Janet E. Bennett and John K. Bennett. He was born in Michigan City, Indiana, on August 11, 1975. In 1990, he moved with his mother to Wimberley, Texas, where he completed high school. He entered the University of Texas at Austin in the Fall of 1993 as a member of the Plan II Honors Program. Paul was awarded a B.A. in Philosophy *cum laude* on December 20, 1997. In May 1998, he will be awarded a B.S. in Computer Sciences with Special Honors and receive a second major certification in Plan II Honors toward his B.A. Paul currently works for Cycorp, an Artificial Intelligence company based in Austin. During the summer of 1998, he will assist in presenting a co-authored paper (related to the work reported here) at the Text Categorization workshop for the AAAI 1998 conference. Paul will continue his education at Carnegie Mellon University where he is to enter the Doctoral Program in the School of Computer Science in the Fall of 1998.