Explanation for Recommender Systems: Satisfaction vs. Promotion

Mustafa Bilgic
mbilgic@cs.utexas.edu

May 19, 2004

Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
Supervising Professor

Elaine Rich
Department of Computer Sciences
University of Texas at Austin
Second Reader

Abstract

There is much work done on Recommender Systems, systems that automate the recommendation process; however there is little work done on explaining recommendations. The only study we know did an experiment measuring which explanation system increased user's acceptance of the item how much (*promotion*). We took a different approach and measured which explanation system estimated the true quality of the item the best so that the user can be satisfied with the selection in the end (*satisfaction*).

Contents

1 Introduction

With the exponential growth of information available on the web and with the increase in the number of books, CDs, and films to choose from, a new need in technology emerged: *recommender systems*. Recommender systems are systems that build a representation of a user's likes and dislikes and they suggest items to the user based on this representation. Different methods have been employed for computing recommendations. The most common two approaches are *collaborative filtering* and *content based* recommendations.

To date, however, most of the research was done on how to efficiently make good recommendations. Even though touched on by some papers, the issue of improving users' trust in recommendations is not much studied. How will the users know that the recommended item really talks to their tastes so that they can adopt it? One straightforward way to tackle this problem is to explain the reasoning behind the recommendation; a similar method that has been tried and proved to be successful for expert systems [4]. This lead the researchers to build "explanation systems", systems that explain why an item is recommended to the user. Different papers [13, 3] describe few explanation systems for their recommender systems. However, the only exclusive study on explanation systems is done by Herlocker et al. [8].

There are two possible approaches to test the effectiveness of explanation systems: the *promotion approach* and the *satisfaction approach*. According to the *promotion approach* the best explanation system is the one that is most successful at convincing the user to adopt the item. The *satisfaction approach*, on the other hand, says that the best explanation is the one that lets the user assess the quality of the item the best. We believe that the *satisfaction approach* is a better approach than *promotion approach* because what matters in the end is the satisfaction of the user. Moreover, if the users are satisfied with their selections, they will continue using the system with an increased trust in it.

Herlocker et al. implement several explanation systems for collaborative filtering. However, we argue that they test explanation systems' performance from a *promotion approach* view, because according to their study, the more positively the user responds to a recommendation with an explanation system e, the better the e is. Additionally, they have done research only on collaborative filtering. We implement three different explanation systems and we test their successes from a *satisfaction approach* view. Moreover, our research applies to both content based systems and collaborative filtering.

We conducted user surveys to find out which explanation system predicts the true rating of the recommended item the best. We used the Learning Intelligent Book Recommending Agent (LIBRA) for our experimental study. LIBRA is an online recommender system that employs both collaborative filtering and content based algorithms [11].

The rest of the document is organized as follows: We give an overview of recommender systems in section 2. Section 3 explains the underlying recommender system. Section 4 discusses the experimental methodology and the results. We discuss future work in section 5. We conclude in section 6.

2 Background

2.1 Recommender Systems

We face far more choices than we can try in the world: which book shall I read, which movie is worth watching, which candidate shall we hire for the job opening, where shall I have dinner tonight, etc. We often rely on recommendations made by word of mouth, recommendation letters, book reviews, or travel guides. Recommendation systems are systems that are meant to augment this process in cases where we need help [16]. Examples of such instances include: gathering many like-minded people in a database, gathering information about thousands of books, combining information in guides far more than we can research, etc.

For a typical recommender system, there are three steps:

1. The user provides some form of input to the system. These inputs can be both explicit and implicit [15]. Ratings submitted by users are among explicit inputs whereas the URLs visited by a user and time spent reading a web site are among possible implicit inputs.

2. These inputs are brought together to form a representation of the user's likes and dislikes. This representation could be as simple as a matrix of items-ratings, or as complex as a data structure combining both content and rating information.

3. The system computes recommendations using these "user profiles."

Even though the steps are essentially the same for most recommender systems, there have been different approaches to both step 2 and 3. Two of

the traditional approaches to building a user profile and computing recommendations are collaborative filtering (CF) and content based (CB) recommendation. Some researchers also tried hybrid approaches to improve the quality of the recommendations.

2.2 Collaborative Filtering

Collaborative filtering, as its name suggests, is an attempt to simulate collaboration among users for sharing recommendations and reviews. The system recommends items to a user by matching his/her personal tastes to that of other users in the system. Most of the CF systems apply the nearest neighbor model for computing recommendations. The nearest neighbor model consists of three basic steps [7]:

First, the users provide ratings for the items they have experienced before.

Second, the *active user* [1] is matched with other users in the system. To do so, correlation coefficients between the *active user* and other users are computed. The users whose ratings highly correlate with that of the *active user* are called the *neighbors*. A standard method for computing correlations is the Pearson Correlation method.

Lastly, predictions for the items that the *active user* has not yet rated, but the *neighbors* have rated are computed, and these items are presented to the user in descending order of the predictions. One way of computing predictions is to compute the weighted average of ratings of the *neighbors* using the correlation coefficients as weights.

Note that CF systems that use the nearest neighbor model rely upon the assumption that people who agreed in the past are likely to agree in the future as well [15].

2.3 Content Based Recommender Systems

CB systems recommend items based on items' content rather than other users' ratings. There are essentially four steps for CB recommendations:

The first step is to gather content data about the items. For example book title, author, description etc. for the books or the director, cast etc. for the movies are some of the common content information. Most systems use Information Extraction techniques to extract these data, and Information

[1] The user for whom the recommendations are computed.

Retrieval techniques to retrieve the relevant information [1, 13]. Web crawlers collecting data off the web are common tools in this step.

The second step is to ask the user to provide some ratings. In this step, the user might be asked to rate random items, or can search and find any books s/he likes.

The third step is to compile a profile of the user using the content information extracted in the first step and the rating information provided in the second step. Different information retrieval or machine learning algorithms can be used to learn a profile. Term-frequency/inverse-document frequency weighting [10] and the Bayesian learning algorithm [13, 14, 6] are some of the many techniques that have been tried.

The last step is to match unrated books' contents with the user profile compiled in the third step and assigning scores to the items depending on the quality of the match. The items are ranked according to their scores and presented to the user in order [13].

2.4 Hybrid Approaches

Both CF and CB systems have advantages and disadvantages over each other. The following lists of problems are discussed in [7, 11, 5, 17, 1, 2]. CF systems face three problems:

1. At the initial use of the system, there are not enough users to match with. This problem is called the *cold start* problem.

2. The users-items-ratings matrix is usually very sparse since recommender systems are used in domains where there are many items to choose from. Thus, finding highly correlated users might be difficult. This problem is called the *sparsity* problem.

3. When an item is added to the system, since nobody has yet rated it, it cannot be recommended to any user. This problem is called the *first rater* problem.

CB systems also suffer from three problems:

1. For some domains, either there is no content information available, or the content is hard to analyze.

2. Formulating taste and quality is not an easy task.

3. These systems can suggest only items whose content match with the user's profile. If the user has tastes that are not represented in his/her profile, items talking to the unrepresented taste will not be recommended.

The *cold start* problem is an issue for CB systems only when the *active user* is at his/her initial stage of use of the program. The *active user* does not have to wait for others to use the system in order to receive good recommendations. The second and third problems of the CF systems are not an issue for the CB systems, because CB systems do not try to match with other users.

The first problem of the CB systems is not an issue for CF systems because CF systems do not look for content. The second problem of CB is partially solved by CF because the taste and quality is entered by the users as ratings. CF also solves the third problem of CB because the *neighbors* might have tastes that the *active user's* profile do not represent.

Seeing that one disadvantage of a system is not an issue for another, much research has done on combining collaborative filtering with content based recommendation. Different techniques were employed to combine the two.

Basu, Hirsh, and Cohen view recommendation as a classification task and use an inductive learning system called *Ripper*. Both collaborative and content information is first transformed into set-valued attributes and then fed to *Ripper*. After trained on some examples, *Ripper* classifies each test item into one of the two categories: *liked, disliked* [2].

Sarwar et. al introduce the idea of *filterbots*, software robots that automatically rate a new item using the item's content information as soon as the item is entered to the database. *Filterbots* act like other users in the system except that they have more ratings than a typical user, and they do not ask for recommendations. Thus, *filterbots* are meant to overcome both the *sparsity* and the *first rater* problem. [17].

Melville, Mooney, and Nagarajan employ a technique where the system first complements the user data by using the content information and then uses collaborative filtering to compute recommendations. In this system, first the sparse users-items-ratings matrix is filled with *pseudo-ratings*, ratings provided by content based recommender and then collaborative filtering is performed on the full matrix [11].

As a last example, Balabanoić and Shoham introduced a system called *Fab*, a recommendation system for the Web. In this system, user profiles are built by using the content information first. Then, the users are matched against each other using these profiles rather than using merely the ratings. Once *neighbors* are found for the *active user*, predictions are made by collaborative filtering methods. In order to perform well, however, the construction of accurate profiles is crucial [1].

3 System Description

3.1 The underlying recommender system

We have a system called Learning Intelligent Book Recommending Agent (LIBRA), which employs collaborative filtering along with content based recommendation. The system is composed of two major components that were adapted from few different researches [13, 11, 7] and another component that we have newly built to bridge these two components. The three components are pictured in red in Figure 1. The first component is the Content Based Ranker that ranks books according the degree of the match between their content and the *active user's* profile. The second component is the Rating Translator that assigns ratings to the books based on their rankings. The third component is the Collaborative Filterer, which performs filtering on the users-items-ratings matrix.

3.1.1 Content Based Ranker

LIBRA, originally built by Mooney [13] as a content based recommender system, has a database of books categorized by six genres: literature, mystery, romance, science fiction, science and technology, and children's books. Each genre has approximately 8000 books. The books are stored in a semistructured representation of ISBN, Author, Title, Description, Subject, Related Authors, and Related Titles. The text in each slot is a bag of words, i.e. unordered sequences of words, and a book is a vector of bags of words. These data were collected in 1999 by crawling Amazon's web page.

After choosing a genre, the *active user* rates a set of books by using the search facility of the system. The user is asked to rate items on a discrete 1-5 rating scale.

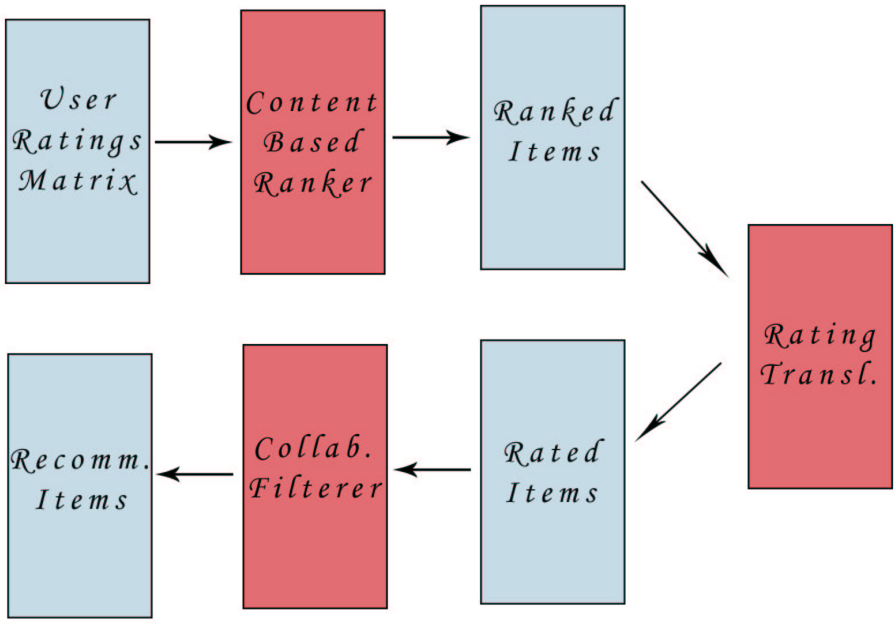

Figure 1: The Underlying Recommender System

The Content Based Ranker employs text categorization techniques using a bag-of-words naive Bayesian text classifier. Instead of classifying into five classes, there are only two classes: {*likes, dislikes*}. However, rather than just classifying into one of these classes, LIBRA assigns a score to each item and the items are ranked according to their scores.

LIBRA composes a "user profile table" for the *active user* after it has been trained on some examples. This table consists of three columns: a slot column, a column for the token in that slot, and the strength column. The strength for a token t in a slot s is: $\frac{P(t|c_l,s)}{P(t|c_d,s)}$ where c_l is the category of *likes*, and c_d is the category of *dislikes*. A score for a book is then computed by multiplying the strengths of each token t in slot s of the book. The last step is to rank the test examples based on their scores. This gives us the "Ranked Items" vector in Figure 1.

To keep brevity, most of the details of the content-based ranker are omitted. Please refer to [13] for a detailed description of the component.

3.1.2 Rating Translator

The Rating Translator acts as a bridge between the Content Based Ranker and the Collaborative Filterer. The Collaborative Filterer compares users based on their ratings. Thus, in order to make use of the output of the Content Based Ranker, the ranks assigned to the test examples must be converted into ratings. We implemented the Rating Translator to do the job.

A straightforward way to convert rankings into ratings is to assign the top rating to the top item and decrease the rating as we go down the sorted list. We do this by using rating-percentages arrays. A rating-percentages array for a user shows what percent of the user's training examples fall into which rating category. An example of such an array is given in Table 1.

Table 1: Rating-percentages

Rating	**Number of training examples**	**Rating percentages**	**Smoother Array**	**Smoothed Array**
5	10	20%	16%	18%
4	20	40%	24%	32%
3	10	20%	20%	20%
2	0	0%	12%	6%
1	10	20%	28%	24%

However, since users usually search for items that they like instead of random items, rating-percentages arrays tend to be skewed towards the top ratings. To minimize the effect of this skewing on ratings, we smooth the rating-percentages arrays with yet another array, called the smoother array. The smoother array is a rating-percentages array that is computed using the data in Table 5 of [13]. This table contains ratings provided by some volunteers on random items.

The smoothed array is computed as follows:

```
for i=1 to 5
  smoothed[i]=(ratingpercentages[i]+w*smoother[i])/(1+w)
```

where w is the smoothing constant. w is inversely proportional with the number of ratings provided by a user. In the example in Table 1, we chose $w = 1$.

The ratings are assigned as follows: Assume that there are x items. The rater will assign $x * smoothed[i]/100$ of the items a rating in the interval $[i, i-1)$. Specifically, y^{th} item in the bin of $[i, i-1)$ will be assigned $i - \frac{y-1}{x*smoothed[i]/100}$.

3.1.3 Collaborative Filterer

We use the Collaborative Filterer that was originally implemented by Herlocker et. al [7] for a news recommender and adapted by Melville et. al [11] for a movie recommender. We slightly modified it to fit into our system. The collaborative filtering for the *active user* is composed of three steps:

1. Compute correlations between the *active user* and other users of the system.
2. Choose the best n users (i.e. *neighbors*) that have the highest correlation with the user.
3. Compute predictions for items using the *neighbors'* ratings.

We use Pearson Correlation to measure the similarity between users, which is:

$$P_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \overline{r}_a) \times (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i=1}^{m} (r_{a,i} - \overline{r}_a)^2 \times \sum_{i=1}^{m} (r_{u,i} - \overline{r}_u)^2}}$$

where m is the number of items, $r_{a,i}$ is the rating given by the *active user* to the item i, and $\overline{r}_a$ is the mean rating of the *active user*. $r_{u,i}$ and $\overline{r}_u$ are similar.

The predictions for items are computed using the formula:

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{n} (r_{u,i} - \overline{r}_u) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}}$$

where n is the number of *neighbors* and $P_{a,u}$ is the Pearson Correlation constant described above.

The items are sorted based on the predicted ratings and then presented to the user.

3.2 The Explanation Systems

There are a handful of good recommender systems available now. Some are available for research purposes such as GroupLens [15], Video Recommender [9], Ringo [18], MovieLens on a PDA [12], and some are for commercial uses such as Amazon, NetFlix, CDNow, and MovieFinder. Even though some of them provide some sort of explanation about their recommendations, most are black boxes as to why they recommend a specific item [8]. Thus, the users' only way to assess the quality of the recommendation is to try the item, i.e. read the book or watch the recommended movie. However, since users use recommender systems to reduce the time of choosing an item, it seems very unlikely that they would spend time trying an item without trusting that it is worth trying.

Explanation systems can prove to be useful for gaining the trust of the users and thus improve the acceptance of the system. Explanation facilities have been used for expert systems before and proved to be useful [4]. On the domain of recommendation, Herlocker et al. have shown that explanation systems increased the acceptance of automated collaborative filtering systems [8].

There are many purposes explanation systems can be used for, some of which are interleaved. Providing transparency into the working of the system, increasing users' trust in the system, letting users spot erroneous recommendations, convincing users to purchase an item, and informing users about the recommendations are some examples of the purposes of explanation systems.

The effectiveness of an explanation system can be measured from two different approaches: the *satisfaction* approach and the *promotion* approach. According to the *satisfaction* approach, the best explanation is the one that lets the users assess the quality of the item the best. For the *promotion* approach, the best explanation is the one that is most successful at convincing the user to adopt an item.

As described in previous sections, there is much research done in computing good recommendations. However, there is very little research done on explaining recommender systems, at least to our knowledge. The only exclusive study on explanation systems was done by Herlocker et al. They present the user a recommendation with twenty one different explanations. The title of the item is removed in order to prevent any bias it might cause. The user is asked to rate the recommendation by just looking at the expla-

nations. Herlocker et al. present the explanation system for which the mean rating is the highest as the best explanation. Thus, we believe that they took the *promotion* approach. For more information about their study on explanation systems, please see [8].

We believe that the *satisfaction* approach is a better approach than the *promotion* approach, because what matters in the end is the satisfaction of the user. If the user is satisfied with his/her selection in the end, the user's trust in and happiness with the system will increase, which in turn means that the user will continue using the system. Thus, we took the *satisfaction* approach.

We have used three explanation systems in our study: *keyword style explanation* (KSE), *neighbor style explanation* (NSE), and *influence style explanation* (ISE). Two factors played a role in choosing these styles of explanations. One factor that affected the decision is the availability of the information, i.e. content and collaborative information. We included KSE for systems that are purely content-based and NSE for purely collaborative filtering systems. ISE is not system dependent as will be described below. The second factor that affected our selection of these styles is that we wanted to test how KSE and ISE perform compared to NSE, which is the best performing explanation system in Herlocker et al.'s study [8].

3.2.1 Keyword Style Explanation

Once the users are provided a recommendation, they are usually eager to learn "what is in it that speaks to their interests." KSE analyzes both the content of the item and the profile of the user to find matches. If KSE is implemented on top of a content-based recommender system, like in our case, such a comparison is already made during the computation of the recommendations. Then, KSE can just re-format the information in a user readable form and present it to the user.

We used the KSE implementation that has been described in [13]. We built it on top of Content Based Ranker explained in section 3.1.1. This KSE presents a table to the user explaining what words had the most influence on the rank of the recommended item. An example is presented in Figure 2.

As described in section 3.1.1, the system builds a profile table for each user containing strengths of each token in each slot. An item's content is matched against this profile. For each token t occurring c times in slot s, a strength of $c * strength(t)$ is assigned, where $strength(t)$ is retrieved from the

Slot	Word	Count	Strength	Explain
DESCRIPTION	HEART	2	94.14	Explain
DESCRIPTION	BEAUTIFUL	1	17.07	Explain
DESCRIPTION	MOTHER	3	11.55	Explain
DESCRIPTION	READ	14	10.63	Explain
DESCRIPTION	STORY	16	9.12	Explain

Figure 2: The Keyword Style Explanation

profile table. Then, the table is sorted according to the strength and only the first twenty entries are displayed to the user. If a user wonders where a word comes from, s/he can click on the *explain* column, which will take him/her to yet another table explaining in which training examples that word occurred and how many times. Only positively rated training examples are included in the table. An example of such a table is presented in Figure 3.

Title	Author	Rating	Count
Hunchback of Notre Dame	Victor Hugo, Walter J. Cobb,	10	11
Till We Have Faces : A Myth Retold	C. S. Lewis, Fritz Eichenberg,	10	10
The Picture of Dorian Gray	Oscar Wilde, Isobel Murray,	8	5

Figure 3: Explanation of where a word appears

Billsus and Pazzani provided similar explanations for their news recommender [3].

3.2.2 Neighbor Style Explanation

If the recommender system is collaborative filtering, then the users wonder what their recommending neighbors think about the recommended items. The *neighbor style explanation* (NSE) is designed to answer this question.

NSE was implemented along with twenty other explanation systems by Herlocker et al [8]. NSE performed best from the *promotion* view. NSE makes use of only the collaborative information to compile a chart that shows how the *active user's neighbors* think about the recommended item. An example is shown in Figure 4. To compute the chart, the *active user's neighbors'* ratings for the recommended item are categorized into three categories: Bad, Neutral, and Good. A bar chart is plotted and presented.

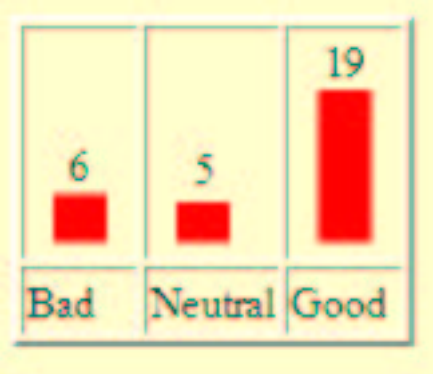

Figure 4: Table showing ratings of a user's neighbors

3.2.3 Influence Style Explanation

An *influence style explanation* (ISE) neither presents content nor collaborative information. Rather, it tells the users how their interactions with the recommender system influenced the recommendation. Thus, ISE is independent of the underlying recommender system.

ISE presents to the users a table of what had the most impact on the recommended item. If the data collected by the underlying system are ratings, this table might contain rated items that had the most influence; if the data are URLs visited by the users, the table can show which URLs caused the recommendation. Amazon has a similar style of explanation, however we do not know how they compute the explanations.

Since LIBRA collects ratings for books, we present a table of books that had the most impact on the recommendation. Each row of the table consists of two columns in which one is a book that the *active user* has rated and

the other column is the influence of that book on this recommendation. An example of such an explanation is shown in Figure 5.

BOOK	YOUR RATING Out of 5	INFLUENCE Out of 100
Of Mice and Men	4	54
1984	4	50
Till We Have Faces : A Myth Retold	5	50
Crime and Punishment	4	46
The Gambler	5	11

Figure 5: Influence Style Explanation

The ideal way to compute the influences is to remove the book whose influence is being computed from the training set, recompute recommendations, and measure the difference in ranking/score of the recommended book. However, this process could be very time consuming and thus could be uncomfortable for the *active user* especially if the user has rated a large number of items (which of course is good thing to get good recommendations). Thereby, different methodologies need to be implemented to reduce the response time.

To compute influences efficiently, we modified the "recompute recommendations" step. Instead of recomputing recommendations, we measure two numbers, content influence and collaborative influence, separately and take the weighted average of them. To compute the content influence of an item, we remove the item from the training set, we retrain the Bayesian Classifier with the remaining training set, and we recompute a score for the recommended item with the new user profile. Please refer to section 3.1.1 for more information on computing scores. The difference in the recommended item's score measured before and after the removal of the item is the content influence. To compute the collaborative influence we remove the item from the "Rated Items" vector and recompute a prediction for the recommended item. The difference in the predictions before and after the removal of the

item is the collaborative influence. Please refer to section 3.1.3 for more information on how predictions are computed. Finally, we combine these two numbers by taking a weighted average of them, giving us the final influence. The equation we used is: $finalInfluence = \frac{contentInfluence + w * collabInfluence}{1+w}$. We chose $w = 1$, however, a different value can be chosen depending on the system. The table is sorted according to $finalInfluence$.

4 Experimental Methodology and Results

4.1 Methodology

As described in section 3.2, we took the *satisfaction* approach. We designed a user study where we asked the users to fill out an online survey. The ideal way to implement the survey of a *satisfaction* approach would be:

```
1      Ask for ratings and compute a recommendation r
2      for each explanation system e
2.1       Present r to the user with e
2.2       Ask the user to rate r
3      Ask the user to try r and then rate it again
```

Because the *satisfaction* approach says that the best explanation is the one that lets the user assess the quality of the item the best, the explanation system for which the difference between the rating provided at step 2.2 and 3 is minimum is the best one (Note that the user rates the same item in step 2.2 and 3.).

In the first step, we ask the *active user* to provide LIBRA at least three ratings, so that LIBRA can provide him/her a decent recommendation along with some meaningful explanation.

We remove the title and author of the book in the second step because we do not want the user to be influenced by it. Moreover, we randomize the order of explanation system seen for each run of the system to minimize the effect of seeing one explanation before another.

Since running this experiment would be very time consuming primarily due to the third step, we slightly modified it. Instead of reading the book, the *active user* can read the Amazon page describing the book and then make a guess about what his satisfaction would be should he read the book.

Comparing the ratings provided at step 2.2 and 3, we hypothesized that:

1. NSE will overestimate the rating of an item.

2. KSE and ISE will estimate the true rating well.

3. All three should have positive correlations with the actual ratings.

4.2 Results

The data were collected primarily from three different sources. Students at the Artificial Intelligence class at the University of Texas at Austin Computer Science department were asked to fill the online survey. The author also asked students at various departments at UT-Austin. Finally, the author asked his friends to fill in the survey. 34 people completed the online survey. Since the system allows the users to repeat steps 2 and 3 with different recommendations, we were able to collect data on 53 recommendations.

We use the following definitions in the rest of the paper. *Explanation-rating* is the rating given to an item by the users by just looking at the explanation of the recommendation. For example, *influence-rating* is the rating given by a user to an item that has been explained by the influence style explanation. These ratings are the ratings provided at step 2 of the experiment. *Actual-rating* on the other hand, is the rating that users give to an item after experiencing the item. This rating is the rating provided at step 3 of the experiment.

Since LIBRA tries to provide good recommendations to the user, we expect both *explanation-ratings* and *actual-ratings* to be high. As can be seen from histogram diagrams in Figure 6, the data are concentrated at categories 3 and especially 4 and 5. When we look at the means in Table 2, we see that means for the ratings are pretty high, being 3.75 at the least.

Table 2: Means and Std Deviations of the ratings

Type	**Mean**	**Std. Deviation**
Actual	3.75	1.02
Influence	3.75	1.07
Keyword	3.75	0.98
Neighbor	4.49	0.64

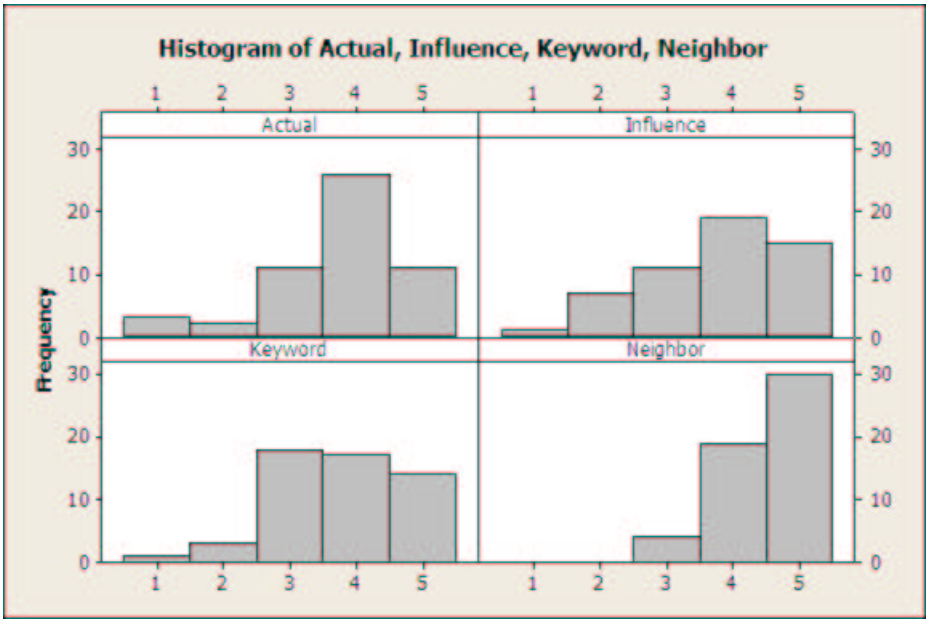

Figure 6: Histogram of Actual, Influence, Keyword, and Neighbor Ratings

We expect to have approximately normal distributions for the differences between the *actual-ratings* and *explanation-ratings*. The histograms of the differences are displayed in Figure 7. The means of the differences can be seen in Table 3.

Table 3: Means, Std Deviations, and Confidence Intervals of the differences

Type	**Mean**	**Std. Deviation**	**95% Conf. Int.**
Influence	0.00	1.30	(-0.36, 0.36)
Keyword	0.00	1.14	(-0.32, 0.32)
Neighbor	0.74	1.21	(0.40, 1.07)

According to the *satisfaction* approach, the best explanation is the one that allowed users to best approximate the *actual-rating*. That is, the distribution of *(explanation-ratings − actual-ratings)* for a good explanation should be centered around 0. Thus, the explanation whose μ_d (the mean of the difference between *explanation-rating* and *actual-rating*) is closest to 0

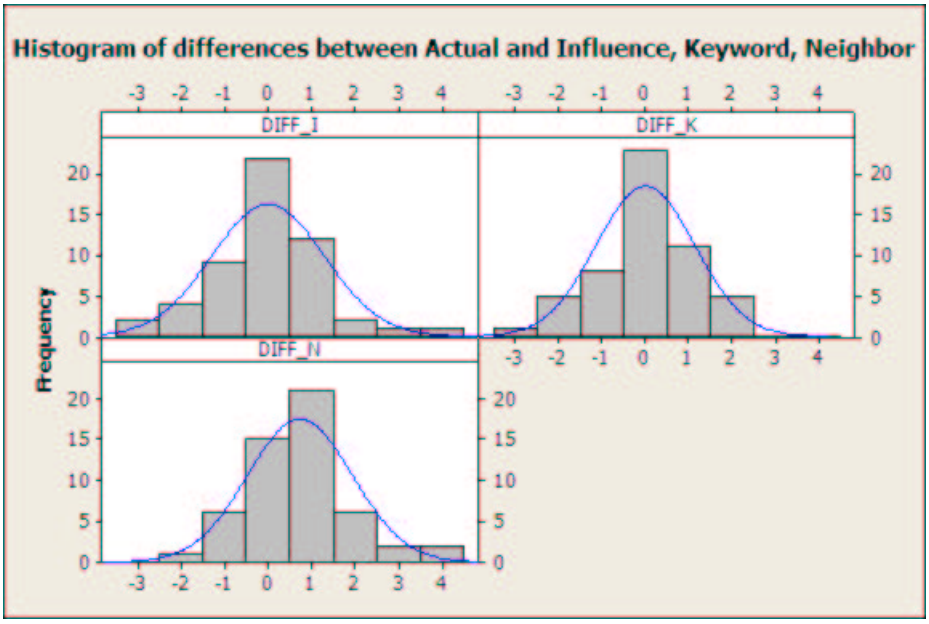

Figure 7: Histogram of Difference Between Actual, Influence, Keyword, and Neighbor Ratings

and that has the smallest standard deviation σ_d in Table 3 is a candidate for being the best explanation. The *keyword style explanation* (KSE) wins with $\mu_d = 0.00$ and $\sigma_d = 1.14$. When we look at the confidence intervals, we see that both KSE and *influence style explanation* (ISE) are very close. This table also shows that, with high probability, the *neighbor style explanation* (NSE) overestimates the *actual-rating* by at least 0.40. Considering that the mean for *actual-ratings* is 3.75, and also considering that the highest rating possible is 5.00, a mean of 0.74 overestimate is a significant overestimation. This table justifies both Hypotheses 1 and 2.

We have also run paired t-tests to find out whether these differences were due to chance only. The null hypothesis we used for all three types of explanations is $H_0(\mu_d = 0)$. Since we did not have prior estimates on whether KSE and ISE would overestimate or underestimate should they estimate wrong, the alternative hypothesis for these explanation systems is $H_a(\mu_d \neq 0)$. However, since we postulated that the NSE would overestimate the *actual-ratings*, the alternative hypothesis is $H_a(\mu_d > 0)$.

The results in Table 4 clearly show that we can reject the null hypothesis for NSE, because the probability of having $\mu_d = 0$ is 0.00. (i.e. $P = 0.00$). So, we accept the alternative hypothesis for NSE. For ISE and KSE on the other hand, we cannot reject the null hypothesis, because $P = 1.00$. So, results for these two styles are inconclusive. Thereby, the t-tests justifies our Hypothesis 1.

Table 4: t-tests

	Actual	
	Hypotheses	**P**
ISE	$H_0(\mu_d = 0), H_a(\mu_d \neq 0)$	1.00
KSE	$H_0(\mu_d = 0), H_a(\mu_d \neq 0)$	1.00
NSE	$H_0(\mu_d = 0), H_a(\mu_d > 0)$	0.00

One other thing that needs to be noted is that the means themselves might be misleading. Consider the following scenario. Assume that we have a new style of explanation called, the *fixed style explanation*, where all *fixed-ratings* are 3. If the *actual-ratings* are equally distributed in the interval $[1, 5]$, then the mean difference between the *fixed-ratings* and the *actual-ratings* will be 0. However, this does not necessarily mean that *fixed style explanation* is a good explanation. A good explanation should have $\mu_d = 0$, a low σ_d, and a strong correlation constant r. That is, *explanation-ratings* should strongly correlate with the *actual-ratings*.

We have calculated the correlation between *actual-ratings* and *explanation-ratings* along with their respective probabilities of being due to chance and they are presented in Table 5.

Table 5: Correlations and P-Values

	Actual	
	r	**P**
ISE	0.23	0.10
KSE	0.34	0.01
NSE	-0.02	0.90

The most strongly correlating explanation is KSE as can be seen in Table

5. The probability of having this high of a correlation due to chance is only 0.01. Next, ISE has a correlation of 0.23, and the probability of having this correlation due to chance is 0.1. Even though it does not meet the standard value of 0.05, 0.1 is still not a big chance. The correlation constant for NSE is negative, however, the chance of having this small of a negative correlation is 90%. The correlation table justifies our Hypothesis 3 fully for KSE and partially for ISE. However, our Hypothesis 3 fails to be true for NSE.

5 Future Work

Another measure that can be calculated is the *disappointment level*, which is, once the user chooses an item by looking at an explanation, how much difference will there be between his expected and true final satisfaction? This requires analyzing the difference between the *explanation-rating* that is above some predefined threshold and the *actual-rating*, because if the *explanation-rating* is below some threshold, the users will not even try the item, thus they will not be disappointed at trying it.

Secondly, the users who participated in the experiment mostly had three ratings in their profile. If they had more, the results could be different.

And lastly, there are twenty other explanation styles described in Herlocker et al.'s paper [8]. The experiment can be repeated with other explanation styles as well. Note that NSE was the best explanation from a *promotion* view. Another style in that study can perform better from a *satisfaction* view.

6 Conclusions

Our Hypothesis 1 and 2 are justified while Hypothesis 3 is partially justified.

The mean of the differences (μ_d), the standard deviations (σ_d), and 95% confidence intervals in Table 3, the t-tests in Table 4, and correlation constants (r) in Table 5 all tell us to use the *keyword style explanation* (KSE), and not to use the *neighbor style explanation* (NSE). However, to produce KSE, the system needs to have content information and needs to make use of this content in the recommendation process.

The *influence style explanation* (ISE) also performed quite well. If the content information is not available, or if it is difficult to extract meaningful

information from the content, then the ISE can be used safely. Note that the differences in results for KSE and ISE were not dramatic.

NSE clearly overestimated the *actual-ratings* and should not be used from the *satisfaction* view. If an explanation system causes the users to overestimate the quality of the item, then the users will expect a high return on trying the item. However, once they experience the item, they will be disappointed and their trust in the system will decrease. They might even decease using the system. If an explanation system underestimates the quality of an item, then the users might not even bother examining the item. So, a good explanation system should do its best to estimate the real quality of the recommended item. KSE and ISE seem to do well from this view.

Lastly, ISE is not system dependent. i.e. The underlying system does not have to be based on neither content nor collaborative information. On the other hand, KSE is based on content information, where as NSE is based on collaborative information. Thereby, ISE can be implemented for any recommender system that gathers user input. Moreover, ISE might help the user understand how her/his interactions with the system affect the recommendation. This information is crucial if the system collects implicit data because the user can then adjust his/her interactions with the system accordingly [8].

References

[1] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[2] Chumki Basu, Haym Hirsh, and William 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. American Association for Artificial Intelligence, 1998.

[3] Daniel Billsus and Michael J. Pazzani. A personal news agent that talks, learns and explains. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 268–275. ACM Press, 1999.

[4] Bruce G. Buchanan and Edward H. Shortliffe, editors. *Rule-Based Expert Systems:The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984.

[5] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, August 1999.

[6] Michelle K. Condliff, David D. Lewis, David Madigan, and Christian Posse. Bayesian mixed-effects models for recommender systems. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[7] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237. ACM Press, 1999.

[8] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pages 241–250. ACM Press, 2000.

[9] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[10] Ken Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

[11] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth National Conference on Artificial Intelligence*, pages 187–192. American Association for Artificial Intelligence, 2002.

[12] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 263–266. ACM Press, 2003.

[13] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 195–204. ACM Press, 2000.

[14] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 54–61, Portland, OR, August 1996.

[15] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186. ACM Press, 1994.

[16] Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[17] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In

Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, pages 345–354. ACM Press, 1998.

[18] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.