Language Independent Recommender Agent

Osman Yucel
The University of Tulsa
800 S Tucker Dr
Tulsa, Oklahoma, USA
osman-yucel@utulsa.edu

Sandip Sen
The University of Tulsa
800 S Tucker Dr
Tulsa, Oklahoma, USA
sandip-sen@utulsa.edu

ABSTRACT

This paper presents a new "Language Independent Recommender Agent" (LIRA), which helps the target users to make use of the information distributed over other users' reviews, or any text-source pair on the web about the candidate items. While existing review-based recommendation systems try to learn the features of the candidate items and the users' preferences for those features, they do not handle the varying perspectives of different users on those features. The approach proposed in this paper constructs an agent for each user, which runs regression algorithms on review texts from different sources and builds trust relations between themselves and other users' agents. LIRA ignores the reviews of the target users which allows the system to work without requiring reviews from target users. Unlike existing systems, LIRA calculates trust values based on prediction accuracy of the sources: LIRA uses predictive capacity of the text sources instead of social connections or rating similarity. A key advantage of the proposed approach is that it does not require the reviews to come from the same community or peer user group. Because we ignore the rating values of the reviewers, and only consider ratings of target user, we can collect and use reviews from different web pages and even from professional critiques as long as we know the item that text was written for and can identify the source of that text. Since we are not clustering or finding similarities between text from different sources, LIRA does not even require the text to be in the same language. To recommend items to a given target user, we can utilize reviews written in multiple languages, as long as text sources are consistent in their usage of language constructs.

CCS Concepts

•Information systems → Recommender systems; • Computing methodologies → Intelligent agents;

Keywords

Recommendation Systems, Text Mining, Agents

1. INTRODUCTION

The demand for accurate recommendation systems has dramatically increased with the growth of user engagement in varying e-Commerce portals and is facilitated by the rapid proliferation of personal and usage data on the Internet. An average user does not have enough time or resources to systematically explore the content on e-Commerce sites to identify items of interest. Hence there is a pressing need for automated and personalized assistance for exploring content at online stores and service outlets.

While a large number of approaches have been developed for creating recommendation systems, there is still a fundamental challenge to developing these systems. The recommendation systems must be able to learn about both the preferences of the users as well as the properties of the contents to provide accurate personalized recommendations to the users. Both of these requirements present new challenges.

The first challenge is collecting information about content properties. The information about the items used by current recommendation systems is generally provided by the creator of the content or by a few privileged individuals. This commonality makes the available information on the contents very objective and limited for the purpose of recommendation. This is because effective recommendation systems need to leverage subjective information from the perspective of individual users of the system. To illustrate this problem, consider the following recommendation request by a user on the *GoodReads* website:

> "Id Like to read a page-turner book with a strong female lead ! i like YA , with humore and some supernatural aspects to it." (*sic*)

The second critical challenge is learning the preferences of the users: humans are not perfectly rational beings and will not necessarily be aware of or be able to effectively articulate their own preferences. There are a number of issues that prevent the recommendation systems from accurately modeling the users. Probably the most problematic of these is that the human users are not perfectly rational and may not be completely aware of all of their own preferences. Studies has exposed paradoxical transitive preferences from users (such as $A \succ B \succ C \succ A$) when asked to compare a set of items [7].

Existing systems interpret the reviews of users by relating them to their ratings. This introduces two problems: (i) since the rating system can differ among different web pages (different ranges, thumbs up/down, etc.), using reviews from different web pages is challenging and (ii) those systems can miss the fact that a feature of an item which causes one user to dislike an item, might cause another user to like the item. To illustrate this problem, we analyze a review by $User_A$ accompanied with a low rating:

> "Romantic chick-flick; total waste of time!"

While it is obvious that $User_A$ did not like the item, the same review can suggest that $User_B$, who likes romantic

movies, might like this movie. Another problem with existing review based systems is that they extract the features of the items and subsequently discard the information about the reviewers. For example, suppose $User_A$ reviewed a hotel $Hotel_X$ as:

"The view from the hotel was excellent!"

Let another user, $User_B$, review a hotel $Hotel_Y$ using the exact same words. Existing review based recommendation systems generally interpret these reviews as "$User_A$ and $User_B$ both care about the views from the hotels" and "$Hotel_X$ and $Hotel_Y$ both has nice views." Therefore these systems will recommend $Hotel_X$ to $User_B$ and $Hotel_Y$ to $User_A$. But if $Hotel_X$ has a nice city view, which $User_A$ likes, and $Hotel_Y$ has a nice nature view, which $User_B$ likes, these recommendations are misleading. LIRA avoids this problem by learning that $User_B$'s reviews are not predictive of $User_A$'s ratings.

Another problem that existing review based recommenders face is that not every user is willing to write reviews for the items they evaluate. Rating/Review ratios in *IMDB* and *GoodReads* are only 1/400 and 1/20 respectively. Because of this sparsity, systems which require reviews from the target users will fail to produce recommendations for a large majority of the users. LIRA does not require them to write reviews and can recommend items to users as long as they rate items.

Finally, most of the existing review based systems depend on topic modelling approaches or finding similarities between different users' reviews. This limits their applicability to situations when all the reviews are in the same language. They also fail to work when the writing style of the users change, such as using "gr8" instead of "great". LIRA can use reviews from different languages for recommendation, and even different writing styles, as long as users are consistent about their language usage or writing style.

In this paper we present an agent based recommender system, LIRA, which creates agents for target users (A_T) and reviewers to create accurate recommendations. LIRA consists of targets user's agent, which processes messages from the other agents, which sends the words or sequence of words (N-Grams) in their reviews to target user's agents. LIRA agents, A_Ts, learn from those messages and then make predictions for unknown items based on the learnt knowledge.

We summarize the problems which LIRA can handle, where the existing systems fall short:

1. LIRA can make use of different perspectives of the users on different features of items, such as liking the different types of views in different hotels.

2. LIRA does not depend on other users' or sources' ratings, it can work on the ratings of a target user, given that reviews are available for items rated by the target user. This property also enables LIRA to utilize reviews from different sources, such as blogs, reviews from different websites, etc.

3. LIRA does not depend on social connections, explicit statement, or rating similarity for calculating trust for a review source. It can calculate these trust values by measuring the capability of those sources to predict target user's rating in training set.

4. LIRA only needs Ratings of target users and does not require them to write reviews to get recommendations.

5. LIRA is language independent: it can leverage reviews written in any language.

2. RELATED WORK

2.1 TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) is a measure of how important a word is to a document [19]. It is widely used in the text mining approaches. TF-IDF value is calculated by multiplying two measures: Term Frequency(TF) and Inverse Document Frequency(IDF).

TF value is calculated using the following equation:

$$TF_{ij} = \frac{f_{ij}}{max_k f_{kj}} \tag{1}$$

In the equation above, TF_{ij} is the Term Frequency value of word i in document j, f_{ij} is the frequency of i in j (number of occurences), $max_k f_{kj}$ is the frequency of the most frequent word k in document j. Thus, the most frequent term in document j gets a TF of 1, and other terms get fractions as their term frequency for this document.

IDF value is calculated using the following equation:

$$IDF_i = \log_2 \frac{N}{n_i} \tag{2}$$

In the equation above IDF_i is the Inverse Document Frequency of word i, n_i is the number of documents that word i appears in, and N is the total number of documents that we have in our collection.

Then TF-IDF value becomes:

$$TF - IDF_{ij} = TF_{ij} \times IDF_i \tag{3}$$

The terms with the highest TF-IDF score are often the terms that best characterize the topic of the document. We use TF-IDF because it can eliminate too frequent words, as we don't want the system to be affected by words which appears too frequently.

2.2 Recommendation Systems

Recommendation techniques can be broadly divided into two categories: *non-personalized* and *personalized* recommendation techniques. *Non-personalized Recommendation* techniques compute the average statistics of the recommended items and recommend the same item set to all customers, i.e., they do not consider individual user preferences. For instance, a bookstore might recommend a bestseller book to all of its customers. *Non-personalized* systems have the advantage of being fast. Moreover, they do not suffer from the *cold start* problem[1], since they do not need an initial data (ratings, etc.) from each customer. Nevertheless, the quality of the results is low because of the lack of personalization.

Personalized Recommendation techniques provide better results by considering individual user preferences. Two well-known approaches are *content-based* and *collaborative filtering*. Moreover, some hybrid approaches have emerged to overcome the problems of these approaches. Adomavicius and Tuzhilin [1] provide a detailed survey about recommender systems.

[1]The problem of not being able to provide recommendations because of the lack of preference information about the user.

Content-based filtering algorithms recommend items that are similar to the ones that the user has liked. *Content-based* systems usually depend on textual content, e.g., keywords. Information Retrieval (IR) techniques, such as *TF-IDF (term frequency / inverse document frequency)* [26], can be applied on the keywords. The similarity among items can be determined by some scoring heuristics, such as *cosine similarity*, after the *TF-IDF* vectors are calculated. Apart from *IR* techniques, *Bayesian classifiers* and different machine learning techniques like *clustering decision trees* and *artificial neural networks* can be used for the similarity calculation [1]. Content-based recommendation systems have been used by the researchers on various topics [24], such as music [40], books [34], movies [8], etc. However these systems need the items to be marked with features, which is usually objective, and cannot make use of user similarities.

Collaborative filtering algorithms identify people with overlapping interests. These algorithms rely on the assumption that people who have exhibited similar interests in the past will continue to have similar interests in the future. Instead of considering correlation of items as in *content-based* algorithms, correlation between users' preferences is examined. This is achieved by analyzing user ratings of items: if two users have provided similar ratings for many items, then it is concluded that these users have similar preferences. Another way of finding users with similar interests is to match demographic characteristics of users, e.g., age, education, geographic locations, gender, etc. After determining like-minded users, the transitive relationship between users and items is utilized. For instance, if user c_1's preferences are similar to that of user c_2's and user c_2 likes item s_1, s_1 is recommended to c_1 if c_1 has not viewed item s_1 yet.

As in the case of *content-based* approaches, *IR* techniques, like *cosine measure*, can be used to calculate similarities between users. However, in *collaborative filtering*, similarity between vectors of the actual user-specified ratings is measured [1]. Moreover, different machine learning techniques are also applicable for identifying similarities between users [1, 3, 33].

Collaborative filtering techniques have their own limitations. They work reliably only when there are sufficient number of users in the system with overlapping characteristics. Another concern is that when a new item is added to the system, it cannot be recommended to others until a number of people have rated it. Furthermore, these techniques are computationally expensive, and the recommendation process becomes cumbersome for millions of users and items. Collaborative filtering has also been tried on various domains such as e-commerce [22], movies [42], etc. Another shortcoming is that these systems cannot identify when two users rate an item with the same rating with different reasons.

2.2.1 Recommendation with Topic Models

A topic model is a statistical model that generates documents (strings of words) from some set of topic clusters. Parametric estimation techniques and unsupervised clustering algorithms such as the Expectation-Maximization algorithm [10] can fit models to observed documents. With the model, it is possible to estimate a likely distribution over topics on a document level and a distribution over words on the topic level. Topic models are an effective tool for characterizing documents since they are not necessary dependent on a language's underlying grammar, and yet can identify significant topic trends using the rate of each word's appearance. Some of the more popular existing topic model frameworks are Latent Semantic Analysis (LSA) [9] and Latent Dirichlet Allocation (LDA) [5].

Topic models are often used in recommender systems in a variety of creative ways [4, 6, 12, 15, 23]. Generally, such recommender systems use topic models to generate latent features in their documents which can then be supplied to some supervised learning algorithm [6, 15].

The fLDA topic model from Agarwal *et al.* [2] extended the LDA model to include ratings as well. The paper approaches the problem where a system has some users and items, and the goal of the recommender system is to make recommendations based on the user's rating history and a textual description of each item. The fLDA model simultaneously learns: (a) the correlation between items' documents and features using LDA, and (b) the users' affinity towards each topic, to produce (c) a rating distribution for each user i and item j. The rating distribution inside the model is defined as a Normal (or binomial) distribution: $y_{ij} \sim \text{Normal}(\alpha_i + \beta_j + s_i^T \bar{z}_j, \sigma^2)$ where y_{ij} is the rating, α_i is the user's rating bias, β_j is the item's rating bias, s_i is the user's affinity towards each topic as a vector, and $\bar{z}$ is a latent variable measuring how each word in the description of item j correlates with a topic.

Similarly, the TopicMF model by Bao *et al.* [4] incorporates a topic model based on matrix factorization; this model considers both ratings and reviews of items in its recommendation. Their model combines this topic model using matrix factorization with another recommender system using matrix factorization to produce a combined model. Given I users, J items, $D = I \times J$ reviews available, W total words and an observed user-to-item rating matrix $\mathbf{r}$ and a review-to-word frequency matrix $\mathbf{F}$, the algorithm attempts to find the best factorization for K hidden topics: $\mathbf{r} = \mathbf{u}_{I\times K}\mathbf{v}_{J\times K}^T, \mathbf{F} = \mathbf{\Theta}_{D\times K}\mathbf{\Phi}_{W\times K}^T$. The hidden matrices are learned by minimizing the least squares equation: $\min_{\mathbf{u},\mathbf{v},\mathbf{\Theta},\mathbf{\Phi}} ||\mathbf{r} - \mathbf{u}\mathbf{v}^T|| + ||\mathbf{F} - \mathbf{\Theta}\mathbf{\Phi}^T|| + \gamma$ where γ is an additional term that includes bias values and measures to prevent learned parameters from growing too large.

McAuley and Leskovec, have proposed the HFT algorithm, which creates topics to identify the hidden topics in the reviews to describe the items of interest, and also the users' interest in those items. They are also using their algorithm to identify the categories of those items automatically [29].

These systems work on reviews provided by the users, but in the process of combining them into topics they lose the information about the reviewers. For example if two people both mention that they like the 'view' of two different hotels, those items and users will be considered similar by these systems. However the types of the views can be different and the users opinion of a 'good view' might also differ between users.

2.2.2 Hybrid Recommendation Systems

Sparsity of available data makes the personalization of rating prediction difficult, as it makes it hard to link users and products to each other. To overcome this problem some researchers focused on generating hybrid recommendation systems to combine the strengths of different approaches [16]. Mostly the base of this hybrids are combination of content-

based [16, 39, 21] and collaborative recommendation systems [38]. Some researchers tried to incorporate the information from social networks to this hybrid recommendation systems either as social tags [32], or social trust [41]. Some of the more recent studies focused on creating hybrid recommendation systems which also make use of sentiment analysis approaches [14, 45]. Leung et. al. proposed a method to extract association rules from data and use it as a supplement to enhance the recommendation system. [20]

2.2.3 Recommendation Using Trust

The area of trust-based recommender systems has been the object of extensive study for the past years. Indeed, trust has been shown to provide significant improvements to classical collaborative filtering techniques. The main difference between most of these trust-enhanced methods is the acquisition of trust values between actor pairs.

Golbeck introduced *TidalTrust* algorithm, which uses a modified breadth-first search, to estimate the trust by using transitive rules[11]. In this study the trust of user u for user v is calculated using u's trusted users' trust on v. Massa et. al. [27, 28] proposed a similar method where instead of only looking at one level trust transitivity, they remove cycles from the trust network. Using a *trust propagation horizon* phase, they allow the trust to diffuse along the network.

While the studies above require an explicit statement about direct trust between users, it is very difficult to get that information from user explicitly. Therefore some other studies focused on defining their own implicit trust values based on implicit information. O'Donovan and Smith proposed a trust value based on the similarity of two user's ratings [35], while Wang et. al. generated a trust metric based on the similarity among users' tastes [43].

2.3 Gaussian Process

A Gaussian Process is a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions [37]. Every Gaussian Process can be fully specified with its mean function $m(x)$ and co-variance function $k(x, x')$. Given N data points, X_n, t_n where the inputs x are vectors of some input dimension, and targets t are the output values, the Gaussian process tries to infer the underlying function, $f(x)$ from the given data [25]. When that function is inferred a Gaussian process can use that function to predict t_{n+1} for a new data point x_{n+1}.

3. METHODOLOGY

In this section we define how LIRA works. We first define how the data is prepared and then explain how LIRA uses that data.

3.1 Data Preparation

LIRA requires target user's ratings and just the text from sources' reviews. Therefore we split the data into two set of triplets. For the target user side we get only $(User, Item, Rating)$ triplets. For sources' side we create triplets of the form $(Source, Item, Review)$. The triplets show that these two sets are only related by items. Which means that ratings of the review sources are not important, which allows us to use reviews from different websites or blogs, as long as we can identify reviews from a common source.

We want to calculate trust for every review source by their ability to predict a target user's rating. Therefore we split the given ratings and reviews into a training set which has 75% of the given information and the remaining 25% as the weight calculation set. We will explain how we use these sets in the trust calculation step in Section 3.2.2.

3.2 Definition of LIRA

We train the LIRA model separately for every single user. In this subsection we will explain how the system is built for one user. We will use the example in the overview of the system (seen Figure 1) to explain the steps. For this example, the rating scale used is $[1, 5]$.

We build an agent for the target user that contains separate text-regression algorithms for every review source.

3.2.1 Text Regression

We start this process by creating data points which consists of the reviews by every source separately and marking them with target user's rating for the corresponding item. That data point is then converted to a feature vector where the features are the words in the review and their values are their respective TF-IDF [19] values.

We choose a machine-learning regression algorithm, such as Neural Network [13], Linear Regression [33], Gaussian Process [37], etc. Because of its simplicity and speed, we chose Gaussian Process to predict the ratings from the given words. In Figure 1 *Learner A1* will only find the common item $I1$ among target user's ratings and source $A1$'s reviews. *Learner A1* will learn that the word sequence **'Chick-Flick'** is likely to result in the target user rating that item 5. Similarly *Learner A2* will find 2 common items, $I1$ and $I3$, getting the reviews from $A2$ for those items. *Learner A2* will learn that the word **'Romantic'** by $A2$ means high ratings by target user and **'Historic'** by $A2$ means low ratings by target user. *Learner A2* will also realize that, word **'Nice'** by $A2$ is not useful for predicting target user's ratings.

When predicting the target user's rating for an item the user has not seen, regression algorithms for different review sources predict separately. Those predictions will then be averaged using the trust values in the ensembler step.

Learners which do not have a review written for the item we are trying to predict the rating for cannot make prediction and we exclude those sources. For example, *Learner A2* will not be able to make a prediction for $I5$.

The final prediction for an item I_X, $\hat{r_{I_X}}$ is calculated as:

$$\hat{r_{I_X}} = \frac{\sum_{u \in Rev(I_X)} Trust(L_u) \times Prediction(L_u, I_X)}{\sum_{u \in Rev(I_X)} Trust(L_u)}, \quad (4)$$

where $Rev(I_X)$ is the set of reviewers who have provided review for item I_X, $Trust(u)$ is the trust value for that reviewer, and $Prediction(L_u, I_X)$ is the prediction by *Learner* u, L_u, for item I_X.

3.2.2 Trust Calculation

Unlike existing systems, LIRA assigns trust for the learners based on their capability, instead of relying on social connections or rating similarity. As mentioned in Section 3.1, data is split into training and weight calculation sets. First, all the learners are trained using the training set. Then we make all the learners predict every item in the weight calculation set. This approach is similar to the use of validation set in common suprvised machine learning approaches.

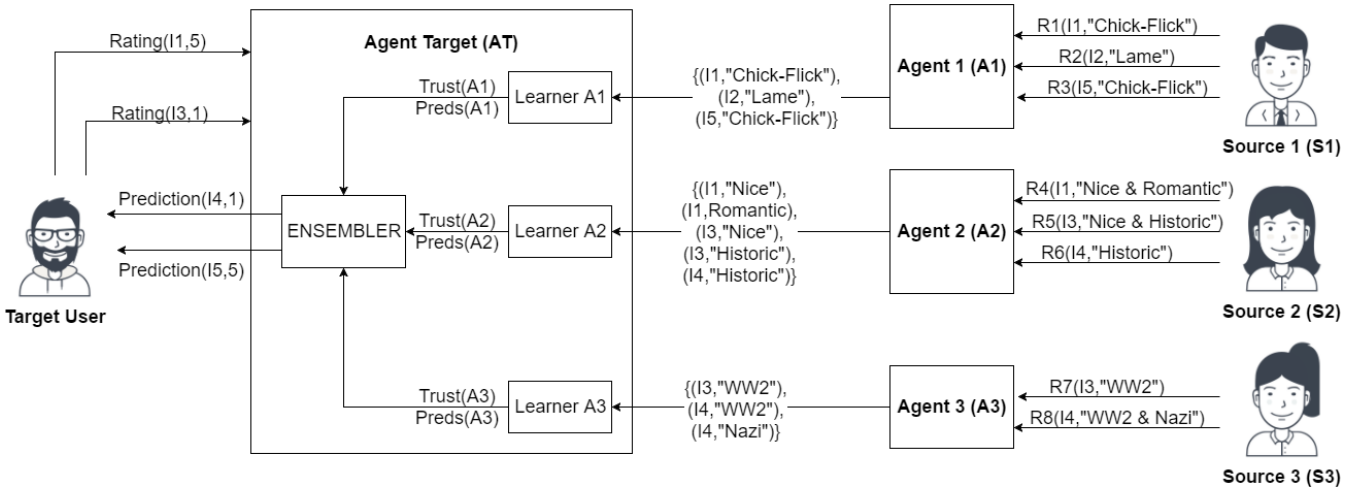

Figure 1: Overview of LIRA

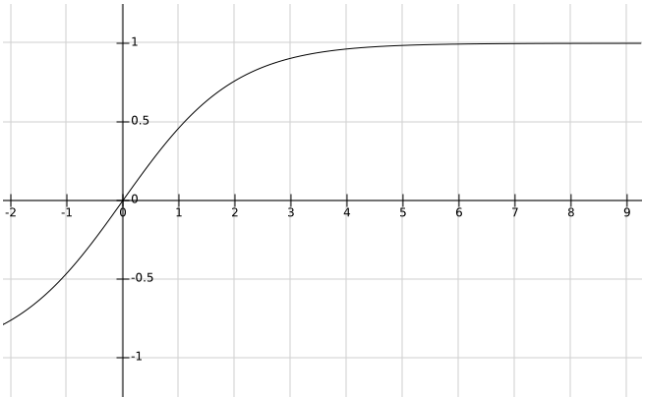

Figure 2: The adjusted sigmoid function.

The Root Mean Squared Error (RMSE) for learner L is

$$RMSE(L) = \sqrt{\frac{\sum_{I \in P_L} (Prediction(L,I) - r_I)^2}{|P_L|}}, \quad (5)$$

where P_L is the set of items which L can make predictions on and $Prediction(L,I)$ is the rating prediction of L for item I.

This error measure, though necessary, is not sufficient. If we only consider the RMSE, a $Learner_A$ which made only one, albeit perfectly accurate, prediction will have the same trust value as a $Learner_B$ who was able to perfectly predict a large number of items. Therefore we include the number of predictions in our trust calculation. We use a sigmoid function to ensure that the magnitude of this count does not suppress the error value. As the number of predictions is non-negative, we choose the following sigmoid function such that $sig(0) = 0$ and $sig(\infty) = 1$ (see Figure 2):

$$sig(x) = \frac{2}{1 + e^{-x}} - 1 \quad (6)$$

We need to combine these two values calculated for every reviewer, to get the final trust value. Trust must be positively correlated with the number of common items; so we choose the sigmoid function, with the number of common items as argument, as the numerator of the trust expression. The error measure calculated must be negatively correlated with the trust value. As the best case is having error equal to 0, we avoid choosing it as the denominator. To make sure the trust value lies between $[0, 1]$ we add 1 to error and make that the denominator. For tuning the relative importance of the number of common items and error, we use an α parameter as the exponent of the denominator. The resultant trust value is given by the following expression:

$$trust(L) = \frac{sigmoid(|P_L|)}{(1 + RMSE(L))^{\alpha}}. \quad (7)$$

We choose $\alpha = 3$ based on experiments.

4. EXPERIMENTAL SETUP

To make sure that we have a satisfactory number of reviews per user, which will be enough for the agents to learn and predict, we increased the density of the user-item bipartite graph by removing every user and item which has less than N reviews. N was chosen separately for every category of reviews, from the interval $N \in [10, 15]$. N is chosen to be the maximum value where a bipartite sub-graph exists, where every user and item node's degree is greater than or equal to N.

Instead of following the traditional method of randomly splitting the data into 2 sets (training and testing), we have built a slightly different approach. Because our algorithm works on every user separately, we make the data split on the user level. For every user we choose a random split of 80%-20% of their ratings and use the former one as the training and weight calculation set, and the latter for testing our algorithm.

To evaluate our recommender system, we used a dataset of product reviews from the web marketplace *Amazon*, provided by McAuley *et al.* [30, 31]. We chose to use 5 of the 24 datasets provided. We chose the Game, Apps and Kindle (which mostly consists of books) domains as we expect that reviewers words are more descriptive of the products for these domains. The Music domain is similar to these domains, in terms of diversity in personal taste, but is likely to be more difficult to describe the items in words. We

Table 1: Properties of datasets

Dataset	# Users	# Items	# Ratings
Apps	4094	2285	97656
Music	1049	990	22772
Game	2816	2141	52158
Health	1251	915	40283
Kindle	3613	3807	116438

also added the Health domain as we wanted to include in our evaluation set a domain which is not very dependent on personal taste. The information about the datasets is given in Table 1

For the evaluation of our approach, we compared our results in 5 datasets, to 5 of the existing recommendation algorithms. We have chosen the simple *UserKNN*[17] and *ItemKNN*[17] algorithms, which are collaborative filtering approaches, as the baseline methods.

We also picked *RegSVD* algorithm proposed by Paterek [36], and 2 algorithms *BiasedMF* and *SVD++*, both proposed by Koren [18]. These three algorithms are all matrix factorization approaches which try extracting topics from the ratings.

4.1 All Ratings Experiment

For this set of experiments we have used all the ratings and reviews in the dataset. LIRA works separately for every user, so for every user LIRA goes over every other users' review to predict the rating of the target user.

4.2 100 User Experiment

This set of experiments are done to test LIRA's capability of making use of the reviews which are on different sources (websites, blogs, critique reviews). We randomly choose 100 users from the dataset. For those 100 users, we use their own ratings but the reviews from all the users in the dataset to predict their ratings. We still use the ratings of the chosen 100 users, to test the competing algorithm.

This approach coincides with the following real life example: Consider a newly built movie recommendation web site which has few (in our example 100) users. As this system will only have the ratings of these few users, the only rating data you can use for the recommendation systems is that. However LIRA can make use of outside sources for the recommendation, such as critique reviews, fan blogs, etc.

The fact that the website itself have only 100 users, make the data available for the existing recommendation systems to be very sparse. Being able to make use of reviews from different sources, LIRA does not get affected by this sparsity.

We made sure that the same set of 100 randomly selected users and exactly the same training and test sets are used to test 5 competing algorithms and the ABRA model.

4.3 Error Measurement

We have chosen to compare the success of the competing algorithms and ABRA using one of the most commonly used error functions in recommendation systems.

Given that r is the actual rating, $\hat{r}$ is the predicted rating, P is set of predictions. We chose the error measure "Mean Absolute Error(MAE)" [44]. The definition of MAE is:

$$MAE = \frac{\sum_{p \in P} |r_p - \hat{r}_p|}{|P|} \quad (8)$$

5. RESULTS

When we run all 6 algorithms on the same dataset which consists of all the ratings and reviews from that domain, we see on Table 2 that LIRA outperforms all competing algorithms on 5 of the domains (See Figure 3 for visual comparison). This is likely because these 5 domains are the ones where the reviewers can express their opinion in a more structured, subjective manner. For example reviewers usually use words like "addictive" or "time-killer" for a game which does give information about the item in question. On the other hand, for "Health" domain, the opinions are less subjective, which might reduce the usefulness of the information extracted from reviews.

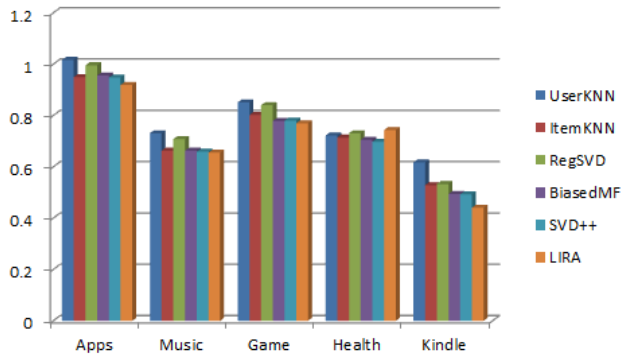

Figure 3: Comparison of MAE values on each domain by each algorithm, in the experiment "All Ratings"

"100 User Experiment" was conducted on randomly selected users from each domain. We see the results of these experiments on Table 3. The results show that LIRA overperformed all other algorithms. From the figure and table, we can see that while in most of the cases, the competing algorithms are suffering from data sparsity in these experiments, RegSVD getting affected the most, LIRA is still able to make accurate recommendations. (See Figure 4 for visual comparison). These results show that even with a few users in the system, LIRA does not lose predictive power.

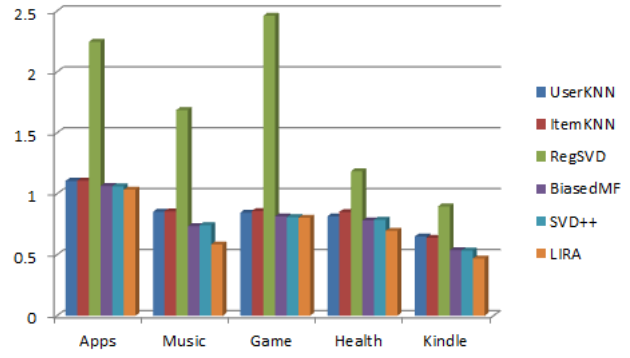

Figure 4: Comparison of MAE values on each domain by each algorithm, in the experiment "100 Users"

Table 2: MAE of 5 competing algorithms and LIRA model on the all ratings experiment

	UserKNN	ItemKNN	RegSVD	BiasedMF	SVD++	LIRA
Apps	1.017	0.950	0.996	0.956	0.948	**0.919**
Music	0.731	0.663	0.708	0.663	0.659	**0.655**
Game	0.851	0.802	0.840	0.779	0.780	**0.770**
Health	0.722	0.714	0.731	0.705	**0.698**	0.743
Kindle	0.617	0.529	0.533	0.494	0.493	**0.441**

Table 3: MAE of 5 competing algorithms and LIRA model on the 100 users experiment

	UserKNN	ItemKNN	RegSVD	BiasedMF	SVD++	LIRA
Apps	1.107	1.109	2.247	1.065	1.063	**1.034**
Music	0.852	0.857	1.687	0.734	0.743	**0.585**
Game	0.846	0.859	2.462	0.816	0.809	**0.803**
Health	0.816	0.851	1.184	0.781	0.787	**0.696**
Kindle	0.651	0.639	0.897	0.538	0.536	**0.469**

6. CONCLUSIONS

While most of the work on recommendation systems is based on ratings and rating similarities, limited representative power of ratings created the need for new recommendation systems which can make of use of the other information available to improve recommendation accuracy. With the introduction of Web 2.0 and burgeoning of online platforms, which rely on user-generated content, a rapidly increasing volume of reviews are available on online or off-site entities. Reviews written by users are more representative and detailed representation of a user's feeling about an item. Descriptive reviews also express users' opinions about the features of candidate items, and helps produce a deeper understanding of their preferences.

In this paper we have proposed an agent-based approach, using the trust aspect of multi-agent societies, that uses regression algorithms to predict the ratings of a target user for a candidate item. Our approach creates agents for both target users and reviewers to create accurate recommendations. LIRA consists of targets user's agent, which listens for messages from the other agents, where the messages are words or sequence of words in their reviews. In LIRA, the agent of the target user learns predictive knowledge from those messages which are subsequently used for producing recommendations. LIRA assigns trust to each source's agent using their accuracy on a set of unknown items. Final prediction of LIRA is calculated using a trust-weighted average of sources' individual prediction. This approach allows the the recommendation system to differentiate two users, who use the same word for different reasons, by inferring from their past reviews where identical words have different predictive values for the target user when used by different sources.

Another key feature of LIRA is that it does not use the ratings of the sources and uses only target user's ratings. This results in the LIRA approach having two key advantages. Firstly, it can use reviews from different sources, such as different websites or fan blogs, without needing those texts to be accompanied with a rating value. This makes LIRA useful for new systems, as they will not have many users to write reviews for them. Secondly, by not taking the sources ratings into account, LIRA treats reviews as descriptive texts. For example, if one or more sources complain about a movie, because they found it "too romantic", LIRA can infer that the movie might be recommended to a target user who does enjoy romantic movies.

Since LIRA does not deal with the meaning of the words, and only use them as signals between agents, it is not important which language the review was written in. As long as sources are consistent with their choice of language or writing style, LIRA will be able to effectively leverage their reviews.

Last, but not the least, a key advantage of LIRA is that it does not require the target users to write reviews. It is well-known that people write reviews much more infrequently than they rate items online. Hence LIRA can be used in many scenarios where review based recommender systems that require target users to write reviews can be used.

We presented experimental results from two sets of experiments. In the first set there was a rating accompanying every review. This experiment showed that making use of just the reviews of other users, ignoring ratings, performs better than just using the ratings and ignoring reviews in most of the domains. Domains where LIRA was more successful than its competitors are likely those where people are more capable and likely to write more descriptive reviews. The second set of experiments show that when there is very little user and rating data available, only LIRA can make use of other sources of information to make accurate recommendations and thus outperform its competitors.

6.1 Future Work

One of the properties of LIRA, which might look like a disadvantage, is that it needs to train regression algorithms for every user-source pair. However it should be mentioned that the agents work independently from each other which makes this approach highly parallelizable. We are currently also working on an approach which provides the same advantages that LIRA provides, without having to create separate learners for every source.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, June 2005.

[2] D. Agarwal and B.-C. Chen. flda: matrix factorization through latent dirichlet allocation. In *Proceedings of*

the third ACM international conference on Web search and data mining, pages 91–100. ACM, 2010.

[3] E. Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[4] Y. Bao, H. Fang, and J. Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2–8, 2014.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[6] L. Chen and F. Wang. Preference-based clustering reviews for augmenting e-commerce recommendation. *Knowledge-Based Systems*, 50:44–59, 2013.

[7] J. M. Davis. The transitivity of preferences. *Behavioral science*, 3(1):26–33, 1958.

[8] S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web*, pages 1041–1042. ACM, 2008.

[9] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[11] J. A. Golbeck. Computing and applying trust in web-based social networks. 2005.

[12] N. Hariri, Y. Zheng, B. Mobasher, and R. Burke. Context-aware recommendation based on review mining. *General Co-Chairs*, page 27, 2011.

[13] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

[14] S. Homoceanu, M. Loster, C. Lofi, and W.-T. Balke. Will i like it? providing product overviews based on opinion excerpts. In *Commerce and Enterprise Computing (CEC), 2011 IEEE 13th Conference on*, pages 26–33. IEEE, 2011.

[15] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 57–64. ACM, 2009.

[16] P. Kazienko and K. Musiał. *Recommendation framework for online social networks*. Springer, 2006.

[17] H.-N. Kim, A.-T. Ji, I. Ha, and G.-S. Jo. Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. *Electronic Commerce Research and Applications*, 9(1):73–83, 2010.

[18] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[19] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[20] C. W.-k. Leung, S. C.-f. Chan, and F.-l. Chung. An empirical study of a cross-level association rule mining approach to cold-start recommendations. *Knowledge-Based Systems*, 21(7):515–529, 2008.

[21] A. Levi, O. Mokryn, C. Diot, and N. Taft. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 115–122. ACM, 2012.

[22] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[23] H. Liu, J. He, T. Wang, W. Song, and X. Du. Combining user preferences and user opinions for accurate recommendation. *Electronic Commerce Research and Applications*, 12(1):14–23, 2013.

[24] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[25] D. J. MacKay. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998.

[26] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[27] P. Massa and P. Avesani. Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers. *International Journal on Semantic Web and Information Systems*, 3(1):39–64, 2007.

[28] P. Massa and B. Bhattacharjee. Using trust in recommender systems: an experimental analysis. In *Trust Management*, pages 221–235. Springer, 2004.

[29] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[30] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2015.

[31] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

[32] S. E. Middleton, N. R. Shadbolt, and D. C. De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004.

[33] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[34] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000.

[35] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th international*

conference on Intelligent user interfaces, pages 167–174. ACM, 2005.

[36] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[37] C. E. Rasmussen. Gaussian processes for machine learning. 2006.

[38] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[39] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[40] M. Soleymani, A. Aljanaki, F. Wiering, and R. C. Veltkamp. Content-based music recommendation using underlying music preference structure. In *Multimedia and Expo (ICME), 2015 IEEE International Conference on*, pages 1–6. IEEE, 2015.

[41] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, pages 675–684. ACM, 2004.

[42] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, pages 114–129, 1998.

[43] J. Wang, J. Yin, Y. Liu, and C. Huang. Trust-based collaborative filtering. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 4, pages 2650–2654. IEEE, 2011.

[44] C. J. Willmott and K. Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[45] D. Yang, D. Zhang, Z. Yu, and Z. Wang. A sentiment-enhanced personalized location recommendation system. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 119–128. ACM, 2013.