USING SEMANTIC SIMILARITY TO ENHANCE ITEM-BASED COLLABORATIVE FILTERING

Xin Jin and Bamshad Mobasher
School of Computer Science, Telecommunication and Information Systems
DePaul University, Chicago, IL, 60604
USA
{xjin, mobasher}@cti.depaul.edu

Abstract

Collaborative Filtering (CF) systems address the problem of making personalized recommendation using knowledge discovery techniques. By predicting user ratings on new items based on historical ratings of other users, CF systems can give reasonable recommendations to new users. Traditionally, user-based CF algorithm can give predictions and recommendations by finding similar users. However these algorithms often suffer from scalability, sparsity and first-rater problems. Recently, some researchers have proposed the item-based CF algorithms, which give predictions based on similar items' ratings. These can alleviate the scalability problems, but these algorithms suffer from the sparsity and first-rater problems. In this paper, we present two algorithms which use semantic similarity to enhance item-based Collaborative Filtering. In both algorithms, we will extract semantic information about items and compute semantic similarity between them. In the first algorithm, we combine the semantic and rating similarities to find most similar items to a target item. In the second algorithm, we use the combined similarity to fill in the original ratings matrix, and then we run the first algorithm on this less sparse ratings matrix. Our experiments show that both algorithms can achieve better prediction accuracy than traditional item-based CF algorithms. Furthermore, the second algorithm can alleviate the sparsity problem.

Key Words

Collaborative Filtering, Semantic Similarity, Recommendation Systems

1. Introduction

Recommendation systems have been widely used in e-commerce. Companies such as Amazon.com and CDNOW provide interesting and powerful recommendation services. Collaborative filtering (CF) [1,2,3] has been a primary approach to building recommendation systems. The goal of a CF system is to recommend new items or predict the utility of a certain item for a certain user based on the user's preference and other users' opinions. In a CF scenario, there is a list of m users $U = \{u_1, u_2, \ldots, u_m\}$ and a list of n items $I = \{i_1, i_2, \ldots, i_n\}$. Each user has a rating vector V, where v_k represents the user's rating for item i_k. (The user's opinion and preference are explicitly given by the rating score.) The task of CF system is for a given active user (or called target user) u_a, (1) to predict the rating score for an unrated item i_k (called target item) or (2) to recommend some items that may be interesting to user u_a. In this paper, we will concentrate on task 1, since if we can achieve task 1, task 2 can be easily achieved with some subjective recommendation strategy.

Currently, there are mainly two types of CF algorithms, user-based and item-based. User-based CF algorithms use some statistical techniques to find a set of users, called user neighbors for the target user. Then different methods can be adopted to combine the user neighbors' ratings to produce a prediction rating for the target user. User-based CF algorithms have shown to be able to give relatively accurate prediction and are used in many applications. However, they have some shortcomings, such as ratings sparsity and scalability problems.

In reality, most commercial recommendation systems deal with tens of thousands of users and items. In these systems, each user may have rated less that 1% of all the items, which makes the ratings matrix very sparse, making it difficult to find user neighbors. Also the similarity measures among users, thus the predicted ratings based on such sparse data tend to be unreliable.

Another problem associated with such systems is the "first-rater" problem. In practice, many new items are added into the recommendation systems every day or week. Since new items have been rated by few users, these items are unlikely to be recommended. To deal with these problems, some researchers have proposed item-based CF algorithms [4,9,10]. The idea is, rather than find similar users (user neighbors), the system tries to find similar items that are rated or purchased by different users in some similar ways. Then, for a target item, predictions can be generated by taking a weighted average of the target user's ratings on these similar items. It has been shown that item-based CF algorithms can achieve comparable or even better prediction accuracy than user-based CF algorithms [4]. Furthermore, since the item similarity can be pre-computed offline, it can alleviate the scalability problem that exists in user-based CF algorithms. However, item-based CF algorithms still suffer from the first-rater and sparsity problems.

In this paper, we will make the use of semantic similarity among items to enhance item-based Collaborative Filtering. We provide two algorithms. In both algorithms we first extract the semantic information about items and compute the semantic similarity of items. Then, in the first algorithm, we combine the semantic similarity and the ratings similarity measures, and use the combined measure to find similar items to the target item. In the second algorithm, we use the combined similarity to fill in the original ratings matrix and alleviate the rating sparsity. Then we use the first algorithm to find similar items and generate predictions.

The reminder of the paper is organized as follows. Section 2 introduces the general item-based CF approach. Section 3 presents our two algorithms using item-based semantic similarity to enhance CF algorithms. In Section 4, we conduct experiments to compare our algorithms to the standard item-based CF algorithm. We will discuss some related work in Section 5 and conclude in Section 6.

2. Item-based CF algorithm

In the foregoing discussion we assume that we have a list of m users $U = \{u_1, u_2, \ldots, u_m\}$, a list of n items $I = \{i_1, i_2, \ldots, i_n\}$ and a ratings matrix R_{m*n}, the value $R_{p,q}$ represents the user u_p's rating on item i_q, if u_p has not rated i_q, then $r_{p,q} = 0$, otherwise $r_{p,q}$ will be a non-zero value from a discrete scale. Our task is to predict user u_k's rating on item i_t, which is not yet rated by u_k. We call the user u_k the target user, and item i_t the target item.

Item-based CF algorithms consist of two processes, finding similar items and generating rating prediction based on similar items' ratings.

2.1 Finding Similar Items

The first step in computing the similarity of items p and item q is to identify all the users who have rated both items p and q. Many measures can be used to compute the similarity between items. Some of the most widely-used measures are given below:

• Cosine Similarity
Here, two items i_p and i_q are considered as two column vectors in the user ratings matrix R. The similarity between items is measured by computing the cosine of these two vectors.

$$sim(i_p, i_q) = \cos(i_p, i_q) = \frac{i_p \bullet i_q}{\sqrt{\| i_p \| * \| i_q \|}}$$

Here, "•" denotes the dot-product of two vectors.

• Correlation-based Similarity

$$sim(i_p, i_q) = \frac{\sum_{k=1}^{m} (R_{k,p} - \overline{R_p}) \bullet (R_{k,q} - \overline{R_q})}{\sqrt{\sum_{k=1}^{m} (R_{k,p} - \overline{R_p})^2 \bullet \sum_{k=1}^{m} (R_{k,q} - \overline{R_q})^2}}$$

Here, $R_{k,p}$ denotes the rating of user k on item p. $\overline{R_p}$ is the average rating of the item p.

• Adjusted Cosine Similarity
Since different users have different rating styles. For example, in moving rating scenario, rating scale between 1 and 5, some users may give rating 5 to a lot of movies they consider to be "not bad"; while some people are "strict" raters, for they only give rating 5 to those movies they like most. To offset the different scale problem, another similarity measure called Adjusted Cosine Similarity is presented.

$$sim(i_p, i_q) = \frac{\sum_{k=1}^{m} (R_{k,p} - \overline{R_k}) \bullet (R_{k,q} - \overline{R_k})}{\sqrt{\sum_{k=1}^{m} (R_{k,p} - \overline{R_k})^2 \bullet \sum_{k=1}^{m} (R_{k,q} - \overline{R_k})^2}}$$

Here, $\overline{R_k}$ is the average rating of user k.

2.2 Prediction Computation

After computing the similarity between items, we select a set of most similar items to the target item and generate a predicted rating for the target item using target user's ratings on the similar items. We use a Weighted Sum as follows.

$$R_{a,k} = \frac{\sum_{t=1}^{K} (R_{a,t} \bullet sim(i_k, i_t))}{\sum_{t=1}^{K} sim(i_k, i_t)}$$

Here, $R_{a,k}$ denotes the prediction rating of target user a on item k. Only the K most similar items (K nearest neighbors of item k) are used to generate the prediction.

As claimed in [4], compared to user-based CF algorithms, item-based CF algorithms can achieve comparable or even better prediction accuracy. At the same time, they can alleviate the scalability problem that exists in user-based CF algorithms, because the item similarity can be pre-computed offline. However, item-based CF algorithms still suffer from the first-rater and sparsity problems.

3. Semantically Enhanced Item-based CF Algorithm

As noted above, item-based CF requires measuring similarities among items based on the user ratings of these items. When a new item is added, few, if any, such ratings exist. In such a case, if we can extract the semantic information of items and define a semantic similarity measure between items, then we can still find item similarities and use these to provide predicted ratings. This is the primary motivation for the use of semantic similarity to enhance item-based CF algorithms.

In this section, we discuss the computation of semantic similarities among items, and we present two algorithms that use semantic similarity to enhance item-based CF systems. In Algorithm 1, we combine items' semantic similarity with ratings similarity, and use the combined similarity to generate predictions. In Algorithm 2, we use this combined semantic similarity measure to estimate ratings for unrated items and fill in the original ratings matrix, thus alleviating the sparsity problem. W then use Algorithm 1 to generate predictions, as before.

3.1 Semantic Similarity

Semantic information about an item consists of the attributes of the item, the item's relationship to other items, its role in the relationship, and other meta-information. For example, in the movie scenario, a given movie has many attributes, such as title, director, cast, release date, genre, and MPAA rating. It also has relationships with other movies, for example, movies "Terminator 1", "Terminator 2" and "Terminator 3" have a common actor "Arnold Schwarzenegger" and have same genre information; "Steve Spielberg" has directed movies "E.T." and "Schindler's List". The semantic information describes the intrinsic features of each item. Combining semantic information about items with user judgments about these items should allow for more intelligent computation of item similarities, and hopefully, lead to better recommendations.

The notions of semantic matching among objects and classes has been a subject of considerable study in the past years [5,6,7]. In this paper we will not discuss these in detail, but we utilize some of these techniques to extract semantic information associated with items and to compute the semantic similarity between them.

3.2 Algorithm 1

In Algorithm 1, we combine items' semantic similarity with their rating similarity and use the combined similarity to generate predictions. Algorithm 1 consists of 4 processes:

• Rating similarity computation
As introduced in Section 2, we can adopt a similarity measure, such as Adjusted Cosine Similarity to compute the rating similarity $RateSim(i_p, i_q)$ from the ratings matrix R.

• Semantic similarity computation
Computing the semantic similarity, $SemSim(i_p, i_q)$, is domain dependent and requires knowledge of the underlying structure in and relationships among objects. In our experiment, we use some techniques introduced in [5] to compute semantic similarities among movie objects. The details of this semantic similarity measure among movies is given in Section 4.3.

• Combining rating and semantic similarities
Here, we use a simple linear combination. $TotalSim(i_p, i_q) = \alpha * SemSim(i_p, i_q) + (1-\alpha) * RateSim(i_p, i_q)$. $TotalSim(i_p, i_q)$ denotes the combined similarity of item p and item q. α is the combination parameter. If $\alpha = 0$, then $TotalSim(i_p, i_q) = RateSim(i_p, i_q)$. It will be the same as the traditional item-based CF algorithm. Finding an appropriate α value is not a trivial task. We choose the proper α value by performing sensitivity analysis on our experimental data.

• Prediction computation
As in traditional item-based CF algorithms, we use the Weighted Sum technique to compute the prediction value for target item k.

$$R_{a,k} = \frac{\sum_{t=1}^{K} (R_{a,t} \bullet TotalSim(i_k, i_t))}{\sum_{t=1}^{K} TotalSim(i_k, i_t)}$$

The only difference is that, here we will use the combined item similarity $TotalSim(i_p, i_q)$ instead of $RateSim(i_p, i_q)$ to generate prediction rating.

3.3 Algorithm 2

The sparsity of the ratings matrix makes it difficult to find similar users or items from among the tens of thousands of users and items in a typical database. Generally, similarity computations based on such sparse data tend to be unreliable causing recommendations or predictions to be inaccurate. In our second algorithm, we combine semantic similarity with rating similarity and use the combined similarity to fill in the original ratings. We then use Algorithm 1 to generate predictions using this less sparse matrix.

There are several ways to fill in the ratings matrix. The first method is to use the rating prediction method of Section 2.2. For all unrated items, we compute a predicted rating based on item similarities and use it to estimate the ratings for these items. Intuitively, this method will lead to the propagation of prediction error. Our experiments also confirm this assertion. The second method is to use semantic similarities to fill in the ratings matrix. However, in practice, since there are tens of thousands of items in the system, the semantic similarity matrix is also very sparse (Most similarity values are either 0 or a very small value).

In this case, estimation may not be an effective strategy for solving the sparsity problem. So, our solution is to combine the rating and semantic similarities, and use the combined similarity value to fill in the raw ratings matrix. In our experiments, we use a filling threshold μ. For each unrated item, we find all its similar items whose similarity values (combined similarity) exceed the threshold μ, and use the weighted sum of these similar items' ratings as the estimated value. Unlike some other methods [18] that fill in values for every unrated item to get a full rating matrix, we only fill values for some of the unrated items. Thus, our method is less time-consuming while achieving comparable results.

After filling the raw ratings matrix, we run the Algorithm 1 in Section 3.2 on the resulting matrix to compute the item similarities and predict ratings for the target user.

4. Evaluation

In this section, we describe the experimental evaluation of the two algorithms presented in the previous section.

4.1 Data Set and Methodology

For our experiments we used the Internet Movie Database (IMDB) [8], an online movie database, to automatically extract semantic attributes of movies. These attributes included genre, director, cast, release date, etc. For user ratings of movies we used the data from movielens.org. This data includes a random selection of 100,000 user ratings (we only consider those users who had rated 20 or more movies). Furthermore, we randomly chose 90% of the ratings data as training data set and the 10% as test data set.

We compared our two CF algorithms (Algorithm 1 in Section 3.2 and Algorithm 2 in Section 3.3) to standard item-based CF algorithms. For each user rating in the test data set, we used the three algorithms to generate predicted ratings for unrated movies. The performance of these algorithms was measured by comparing the predicted ratings to the actual rating.

4.2 Evaluation Metrics

In our experiments, we used MAE (Mean Absolute Error) as our evaluation metric to measure prediction accuracy, since it is the most widely used metric in CF research. MAE is a measure of the deviation of prediction from the actual value. Let's say, for item *i*, we have an actual rating p_i and a prediction rating q_i. We first compute the absolute error between them and then compute the average absolute error of all *N* rating-prediction pairs as the MAE.

$$MAE = \frac{\sum_{i=1}^{N} | p_i - q_i |}{N}$$

The goal of an algorithm is to achieve as low a MAE as possible.

4.3 Experiment Procedure

For traditional item-based algorithm as described in Section 2, we ran the algorithm with different neighborhood sizes and reported the MAE.

For Algorithm 1, first we used a spider program to extract movie information from IMDB website. Currently, we only extract these attributes for each movie: name of the movie, year, director, genre, cast, MPAA rating, plot. We preprocess the data, that is, we conduct some association analysis on movie "genre" information and define a genre hierarchy. We group "year" of movies into several time intervals. For other attributes, we treat each as a bag of words. Now for each attribute, we can adopt a proper method to measure the similarity between different movies. We define a linear combination to combine all these attributes, e.g., for movie *i* and *j*, we define their semantic similarity as:

$$SemSim(i,j) = a_1 * CastSim(i,j) \\ + a_2 * DirectorSim(i,j) + a_3 * GenreSim(i,j) + ...$$

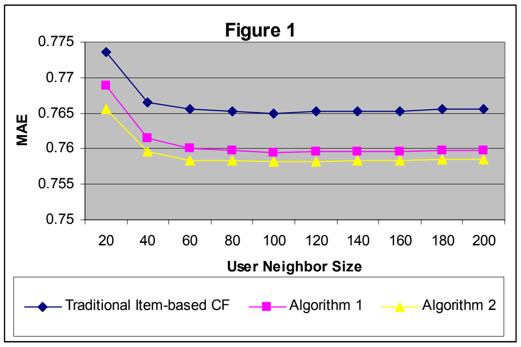
Here, a_1, a_2, a_3, … are parameters which are predefined after doing sensitivity analysis using statistical techniques. The next step is to combine semantic similarity with rating similarity and generate predictions as described in Section 3.2.

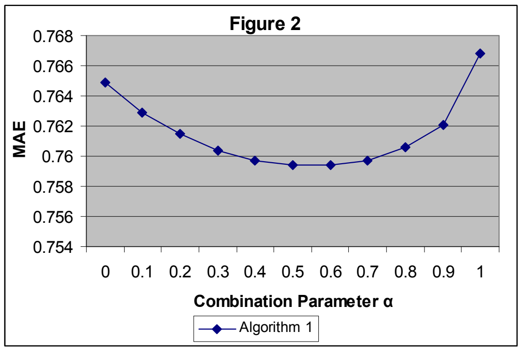
For Algorithm 2, as in Algorithm 1, we compute the semantic similarity and the combine similarity values. Then we perform the filling process described in Section 3.3. After filling, we run Algorithm 1 on the new ratings matrix.

We conducted three experiments. In the first experiment, for each item neighborhood size *K*, we ran the three algorithms and measured MAE for each algorithm at *K*. In the second experiment we chose different combination parameters (from 0 to 1) to combine semantic similarity and rating similarity values, and then ran Algorithm 1. We measured MAE of Algorithm 1 at different combination parameter levels. Finally, in the third experiment, we chose different filling thresholds μ and performed the filling process. In this case, we measured the MAE of Algorithm 2 at different filling threshold values.

4.4 Experiment Result

Our experimental results are shown in Figure 1, Figure 2 and Table 1. Figure 1 shows that both algorithms perform better than the standard item-based CF algorithm, at all neighborhood sizes. Furthermore, Algorithm 2 performs slightly better than Algorithm 1. Figure 2 shows that, using Algorithm 1, at different combination levels, combining semantic similarity with rating similarity can improve the prediction accuracy of item-based CF algorithms. When combination parameter α is 0.5, we can achieve the best result. When $\alpha = 1$, which means only using item semantic similarity to find similar users and predict rating, we can achieve MAE of 0.7668, which is still comparable to traditional ratings similarity based predictions.





Experiments also show that in Algorithm 2, using a filling threshold μ >=0.8, we can always get a "smaller" MAE (Table 1).

μ	0.95	0.9	0.85	0.8
Original ratings matrix sparsity rate	94.3%	94.3%	94.3%	94.3%
Filled ratings matrix sparsity rate	94%	93.7%	93.3%	92.9%
MAE	0.7591	0.7591	0.7588	0.7581

Table 1

Note that Sparsity rate in Table 1 is computed as unrated slot number divided by all slot numbers in the ratings matrix *R*.

5. Discussion and Related Work

Using semantic information about the items in our algorithms, we have alleviated the sparsity and first-rater problems in CF systems. At the same time, we can achieve better or comparable prediction accuracy. We think the reason may be that, in item-based CF systems, the goal is to find similar items and use them to make predictions. The item rating similarities can capture users' dislikes and likes of items, which is a kind of subjective "indictor" of item similarities. On the other hand, semantic similarity can capture certain intrinsic relationships between items. Thus it can serve as an objective "indictor" of similarities. The combination of semantic similarity and rating similarity can help us to capture both the subjective and the objective criteria to different degrees, and thus make better predictions.

Recently there has been increasing interest in item-based CF algorithms. One of the first Item-based CF algorithms was developed by Billsus et al. [9]. Sarwar et al. [4] compared their Item-based CF algorithm to other CF algorithms. Deshpande et al. [10] presented an Item-based Top-*N* recommendation system which used item-item as well as item-itemset similarities to generate recommendations. Web usage mining and other techniques also have been used to address the scalability problem of collaborative filtering [11,12].

A number of previous studies have attempted to combine content information and domain knowledge with collaborative filtering. In [13], domain knowledge has been used in recommendation systems. The Fab system [14] integrated content information with collaborative filtering by presenting a public "topic" filter and a personal "interest" filter to represent users' profiles. Users' feedback was collected to refine these two filters. Cotter et al. [15] used a content-based and a collaborative filtering method to generate separate recommendations and merged them to form a final recommendation set. Pazzani [16] used the Winnow algorithm to learn user profiles as a set of words, which were then used to generate predictions. Basu et al. [17] used inductive logic programming to transform the prediction problem into a classification problem. Melivlle et al. [18] presented a content-boosted collaborative filtering algorithm, combining a pure content-based predictor and a pure collaborative filtering predictor to generate final predictions. They also performed filling on the raw rating matrix to get a full pseudo rating matrix.

6. Conclusion and Future Work

In this paper, we presented two algorithms that use semantic similarities among items to enhance item-based CF algorithms. Our experiments shows that our algorithms can achieve better results than standard item-based CF algorithms. At the same time, the use of semantic information allows the system to be able to give reasonable recommendations even for new items, thus alleviating the "first-rater" problem commonly associated with CF systems. The use of semantic similarities also provides a more robust way for estimating a user's ratings and filling in the sparse ratings matrix. Our experiments also show that reducing the sparsity of the ratings data in this way improves prediction accuracy.

We think more research may be done in these directions. In our experiments we found that the generation of item semantic similarity is not a trivial question, and this measure clearly has a great influence on the output of the algorithms. While semantic similarity measures are by nature domain dependent, it might be useful to study the appropriate measures that can be used in this context for specific classes of objects commonly encountered in CF applications. Another area of future work is to boost the accuracy of the proposed algorithms by using more advanced models for computing semantic similarities.

References

[1] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry, Using Collaborative Filtering to Weave an Information Tapestry, *CACM*, *35*(12), 1992, 61-70.

[2] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, GroupLens: An Open Architecture for Collaborative Filtering of Netnews, *CSCW94: Conference on Computer Supported Coorperative Work,* Chapel Hill, NC, 1994, 175-186.

[3] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, An Algorithmic Framework for Performing Collaborative Filtering, *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999.

[4] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, Item-based Collaborative Filtering Recommendation Algorithms, *Tenth International World Wide Web Conference*, Hong Kong, 2001.

[5] M.A. Rodriguez and M.J. Egenhofer, Determining Semantic Similarity among Entity Classes from Different Ontologies, *IEEE Transactions on Knowledge and Data Engineering*, *15*(2), 2003, 442-456.

[6] L. Palopoli, D. Sacca, G. Terracina, and D. Ursino, Uniform Techniques for Deriving Similarities of Objects and Subschemes in Heterogeneous Databases, *IEEE Transactions on Knowledge and Data Engineering*, *15*(1), 2003, 271-294.

[7] P. Ganesan, H. Garcia-Molina, and J. Widom, Exploiting Hierarchical Domain Structure to Compute Similarity, *ACM Transactions on Information Systems*, *21*(1), 2003, 63-94.

[8] Internet Movie Database. Http://www.imdb.com.

[9] D. Billsus and M.J. Pazzani, Learning Collaborative Information Filters, *Proceedings of the International Conference on Machine Learning,* Madison, WI, 1998.

[10] M. Deshpande and G. Karypis, Item-based Top N Recommendation Algorithm, Tech. Report 03-0002, University of Minnesota, 2003.

[11] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization, in *Data Mining and Knowledge Discovery*, Kluwer Publishing, *6*(1), 2002, 61-82.

[12] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, Improving the Effectiveness of Collaborative Filtering on Anonymous Web Usage Data, in *Proceedings of the IJCAI 2001 Workshop on Intelligent Techniques for Web Personalization (ITWP01)*, Seattle, 2001.

[13] B. Mobasher and H. Dai, Using Ontologies to Discover Domain-Level Web Usage Profiles, in *Proceedings of the Second Workshop on Semantic Web Mining*, at *PKDD'02*, Helsinki, Finland, 2002.

[14] M. Balabanovic and Y. Shoham, Fab: Content-based, Collaborative Recommendation, *Communications of the ACM*, *40*(3), 1997.

[15] P. Cotter and B. Smyth, PTV: Intelligent Personalized TV Guides, The *Twelfth Conference on Innovative Applications of Artificial Intelligence*, Austin, TX, 2000, 957-964.

[16] M.J. Pazzani, A Framework for Collaborative, Content-based and Demographic Filtering, *Artificial Intelligence Review*, *13*(5-6), 1999, 393-408.

[17] C. Basu, H. Hirsh, and W. Cohen, Recommendation as Classification: Using Social and Content-Based Information in Recommendation, *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.

[18] P. Melville, R.J. Mooney, and R. Nagarajan, Content-Boosted Collaborative Filtering, *Proceedings of the SIGIR2001 Workshop on Recommender Systems*, New Orleans, LA, 2001.