Intelligent Decision Technologies 9 (2015) 233–242
DOI 10.3233/IDT-140219
IOS Press

Tuning machine learning algorithms for content-based movie recommendation

Maria Brbić* and Ivana Podnar Žarko
Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia

Abstract. Machine learning algorithms are often used in content-based recommender systems since a recommendation task can naturally be reduced to a classification problem: A recommender needs to learn a classifier for a given user where learning examples are characteristics of items previously liked/bought/seen by the user. However, multi-valued and continuous attributes require special approaches for classifier implementation as they can significantly influence classifier accuracy. In this paper we propose novel approaches for handling multi-valued and continuous attributes adequate for the naïve Bayes classifier and decision trees classifier, and tune it for content-based movie recommendation. We evaluate the performance of the resulting approaches using the MovieLens data set enriched with movie details retrieved from the Internet Movie Database. Our empirical results demonstrate that the naïve Bayes classifier is more suitable for content-based movie recommendation than the decision trees algorithm. In addition, the naïve Bayes classifier achieves better results with smart discretization of continuous attributes compared to the approach which models continuous attributes with a Gaussian distribution. Finally, we combine our best performing content-based algorithm with the k-means clustering algorithm typically used for collaborative filtering, and evaluate the performance of the resulting hybrid approach for a movie recommendation task. The experimental results clearly show that the hybrid approach significantly increases recommendation accuracy compared to collaborative filtering while reducing the risk of over specification, which is a typical problem of content-based approaches.

Keywords: Recommendation systems, content-based movie recommendation, machine learning, naïve Bayes classifier, decision trees, k-means clustering, continuous attributes, multi-valued attributes

1. Introduction

In a world faced with exponential data growth, we often find it hard to discover relevant information in due time. Such information overload has naturally led to the need for personalized solutions capable to guide the quest for relevant information which is a prerequisite for decision making. Recommender systems fall within the category of tools that ease the process of searching for adequate items from potentially huge item sets since they are designed to filter out irrelevant information and identify potentially interesting and relevant items. Therefore, the basic task of a recommender system is to provide support in a decision making process in presence of a large amount of information. Machine learning algorithms are adequate for recommender implementation as they can be used to learn a user profile. The main idea is to build a model based on the items that were liked or disliked by a user in the past, which gives a prediction into which category a new item would belong to. Thus, machine learning algorithms enable us to recognize patterns in user behavior and to identify items potentially relevant and interesting to the user.

There are three main classes of algorithms used in recommender systems: content-based filtering, collaborative filtering and a hybrid solution combining content-based and collaborative approach. Content-based filtering predicts user preferences based on the characteristics of items which a user has liked or bought in the past. The machine learning algorithms are inherently adequate to implement the content-

*Corresponding author: Maria Brbić, Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia. Tel.: +385 989031084; E-mail: mariabrbic@gmail.com.

ISSN 1872-4981/15/$35.00

based approach since a recommendation task can naturally be reduced to a classification problem: A recommender needs to learn a classifier for a given user where learning examples are characteristics of items which the user has liked/bought/seen in the past. Thus, machine learning algorithms have been successfully applied in content-based recommender systems in many domains. The second approach, collaborative filtering, aims to find similarities between a user and other users who have already used the system, and recommends those items which were liked/bought/seen by similar users. Machine learning methods which are commonly used for this task include Bayesian networks, clustering algorithms, latent semantic analysis, neighborhood-based algorithms. Hybrid systems combine both content-based and collaborative approaches. Several studies have demonstrated greater accuracy of hybrid approaches compared to purely collaborative or content-based approaches [10].

In this paper we systematically describe the machine learning algorithms for content-based recommendation which use either the naïve Bayes classifier or decision trees classifier. The algorithms are tuned to handle multi-valued and continuous attributes since such attributes cause serious shortcomings to classifier implementation and usage in practice. We perform experimental evaluation to compare the proposed solutions and, in addition, evaluate the performance of a hybrid approach which combines our best-performing content-based solution with the k-means clustering algorithm for collaborative filtering. The evaluation is performed using the MovieLens data set [9] which has been extended with movie details retrieved from the Internet Movie Database (IMDB) [12].

In particular, we propose a novel approach for handling multi-valued attributes when using either the naïve Bayes classifier or decision trees classifier. In addition, we present a solution for the discretization of continuous attributes which uses a genetic algorithm as a preprocessing step in the naïve Bayes classification. This approach is compared to the approach in which attributes with continuous values are modeled with a Gaussian distribution. Our empirical results demonstrate that the naïve Bayes classifier is more suitable for content-based recommendation of movies from the MovieLens data set than the decision trees classifier. They also show that better results can be achieved with discretization of continuous attributes compared to modeling continuous attributes with a Gaussian distribution for the naïve Bayes classifier. Finally, we combine the naïve Bayes classifier which applies discretization of continuous attributes with the k-means clustering algorithm, and evaluate the performance of the resulting hybrid approach for a movie recommendation task. Experimental results, again obtained using the MovieLens data set, demonstrate that the hybrid approach is less accurate than the content-based approach, but it significantly increases recommendation accuracy compared to collaborative filtering. Moreover, the hybrid approach provides better diversity and reduces the risk of over specification, which is a typical problem of content-based solutions.

The paper is structured in the following way: Section 2 discusses related work. In Section 3 we present central ideas of the naïve Bayes classifer and decision trees classifier and discuss solutions for handling multi-valued and continuous attributes. In Section 4 we present our experimental results and Section 5 concludes the paper.

2. Related work

A number of classification algorithms have been applied for content-based recommendation in a variety of domains. For example, some of the well-known online movie databases such as Rotten Tomatoes and IMDB are known to be content-based. Content-based recommendation systems describe items that can be recommended to a user with a set of attributes. When items are described with unstructured text, it is necessary to convert free text data to a structured representation such that machine learning algorithms can be used to learn a user profile [15].

The naïve Bayes classifier has been applied in several content-based recommendation systems. In [16] the naïve Bayes classifier is used to learn user profiles such that the system can predict which web sites on a given topic would be interesting to a user. The naïve Bayes classifier is compared with the nearest neighbor algorithm, decision trees algorithms, Rocchio's algorithm and neural nets. It is shown that the naïve Bayes classifier performs at least as well as these approaches and that decision trees algorithms is not well suited for this task. The naïve Bayes classifier is also applied in a book recommendation system, called LIBRA [18]. The developed system uses semi-structured information about the items that can be recommended. Standard decision trees algorithms such as ID3 and C4.5 are efficiently applied in [7] on the retail data set. According to [15], decision trees algorithms are often used with structured data and do not perform

well for unstructured classification tasks. We have decided to analyze the performance of the decision trees algorithm for a movie recommendation task since in our use case the data is structured and it does not have a large number of attributes. Various machine learning algorithms have been applied in content-based recommendation systems, but the problem of multi-valued attributes, which is common in content-based recommendation systems, has not been discussed so far. Thus, in this paper we focus on potential solutions to this problem. We show, also for the first time, how different handling of continuous attributes in the naïve Bayes classifier can affect recommendation accuracy.

Machine learning algorithms that can be successfully applied for collaborative filtering are clustering algorithms. According to [4], clustering algorithms applied for collaborative filtering can achieve comparable recommendation quality as the basic collaborative filtering approach and significantly improve the online performance. In this paper we use the k-means clustering algorithm for collaborative filtering which is similar to the k-medians clustering algorithm proposed in [3] that, in addition to evaluating the k-medians clustering algorithm, also analyzes a number of other machine learning approaches including classification algorithms, methods for dimensionality reduction and probabilistic rating models. The k-means clustering algorithm and its variations are compared with Gibbs Sampling in [14] for the case of CD purchase data. Results have shown that repeated clustering steps using k-means clustering usually do not improve accuracy.

A survey of hybrid recommenders is given in [17]. In particular, [17] defines possible combination methods including the weighted hybrid recommender system which we used in our implementation. In [1], the authors have shown that their cascade-hybrid music recommender achieves significantly higher performance than the standard recommendation techniques based only on collaborative or content-based approach. Another hybrid movie recommender system that is also tested on the MovieLens data set is proposed in [5]. The content-based recommender is based on neural networks, while the collaborative filtering part of the system uses neighborhood-based method. Another example of a movie recommendation system which uses a hybrid approach for recommendation is MovieGEN [8]. The developed system uses a support vector machine (SVM) to predict the genres and period of the movies that the user prefers. It applies the k-means clustering algorithm to cluster movies, instead of clustering users, as it is done in [3]. After the clusters are formed, the system generates questions to provide better recommendations to users. This hybrid system is hard to compare with our implementation since the information used in the evaulation (genre and period) differs a lot from the one used in our experiments. In addition, MovieGEN uses personal user information which is not available in our data set. The discussion of recently developed recommendation systems applied in different domains such as medicine, tourism, education, etc. can be found in [11].

3. Algorithm description

In this section we present two classifiers for content-based filtering: the naïve Bayes classifier and decision trees classifier as well as potential solutions to the problems that we encountered when learning classifiers for movie recommendation tasks. We describe potential solutions for handling multi-valued and continuous attributes for both algorithms.

3.1. Naïve Bayes classifier

The naïve Bayes classifier is a probabilistic classification method based on the Bayes theorem which assumes that attributes are independent given a class label. Although this assumption does not hold in most real-world applications, the naïve Bayes classifier is often used in practice because, in addition to its simplicity, it frequently outperforms more complex classification algorithms.

Using the naïve Bayes assumption and Bayes theorem, the posterior probability $P(C_i|x_1,\ldots,x_n)$ that an item belongs to class C_i given n attribute values $x_1,\ldots,x_n$ is proportional to

$$P\left(C_i\right)\prod_{j=1}^{n}P(x_j|C_i),$$

where $P\left(C_i\right)$ is the probability of class C_i membership and $P(x_j|C_i)$ is the probability that attribute j takes value x_j for a given class value C_i. The probabilities can be estimated from a training set using the maximum likelihood estimation method. In our implementation we use the Laplace smoothing to avoid zero probabilities. In the prediction phase an item is assigned to the most likely class, i.e., class with the highest probability.

3.1.1. Multi-valued attributes

Multi-valued attributes are those attributes that can obtain multiple values for the same instance. An example is a movie feature "actor" because this feature usually obtains more than one value, since a number of leading actors are usually listed in movie descriptions. To the best of our knowledge, different approaches for handling multi-valued attributes have not been discussed in the literature so far. We have considered two potential approaches for handling multi-valued attributes within our naïve Bayes classifier as described below. The first approach chooses a single distinguishing value for a multi-valued attribute and ignores others, while the second approach considers all values attached to a multi-valued attribute as potentially distinguishing features.

In the first approach, during the learning phase which calculates the conditional probability that an attribute takes a value given a class value, multi-valued attributes are processed as if the attribute takes all values independently given a class value. For example, if attribute a_i takes values $\{v_1, v_2\}$ for a learning example while the class of a given learning example is C_j, then it is seen in the following way: attribute a_i takes value $\{v_1\}$ given a class C_j and it takes value $\{v_2\}$ given the class C_j. This information is used to estimate the probabilities $P(a_i = v_1|C_j)$ and $P(a_i = v_2|C_j)$ which are calculated separately. Thus, a single attribute value from the set of values for a given learning example is used independently to estimate its probability for a given class value irrespective of other values appearing in a multi-valued attribute.

In the prediction phase, for a given example we identify a value from the set of values for a given multi-valued attribute with the highest probability, and consider only the value with the maximum probability, while other values are ignored. For example, if a multi-valued attribute a_i took n values $\{v_1, \ldots, v_n\}$ for a given example, we find conditional probabilities for all attribute values $P(v_1|C_j), \ldots, P(v_1|C_j)$, and consider only $\max_k P(a_i = v_k|C_j)$ when calculating the posterior probability that an example belongs to class C_i. Other conditional probabilities are ignored when calculating the posterior probability for a given class C_j. An intuitive explanation for handling multi-valued attributes this way is the following: Users often choose movies in such way that a single movie attribute value, e.g., their favorite actor, affects the selection of a particular movie while other values of that attribute are negligible, e.g. other actors in the movie.

The second approach proposes another solution for handling multi-valued attributes in the prediction phase. Instead of taking into account the maximum probability as previously explained, we consider the product $\prod_{k=1}^{n} P(a_i = v_k|C_j)$ for modeling conditional probability that an attribute takes values given a class value. Thus the formula for finding the most likely class is the following:

$$arg\ \max_j P(C_j) \prod_{i=1}^{m} \prod_{k=1}^{n} P(a_i = v_k|C_j),$$

where m represents all attributes and n all attribute values.

The main advantage of this approach is that it does not ignore attribute values, while its main disadvantage compared to the first approach is that when making recommendations, movies can only be ranked according to grade predictions, i.e., whole numbers, and not by probabilities. We have tested both approaches in our experiments and the first one has shown slightly better results. Therefore we choose the first approach for our movie recommender and use it in all our experiments.

3.1.2. Continuous attributes

Two common methods for handling continuous attributes when applying the naïve Bayes classifier are found in the literature: modeling continuous attribute values with a Gaussian distribution and discretization of continuous attribute values. We compare these two different approaches for handling continuous attributes.

In the first approach we assume that the continuous values are distributed according to a Gaussian distribution. During the learning phase we estimate the mean and variance for all continuous attributes appearing in each class from the learning set. In the prediction phase we calculate the conditional probability that a continuous attribute takes a value for a given class by using the probability density function for normal distribution. For example, suppose that in the learning phase we estimate that a continuous attribute a has the mean μ_C and variance σ_C^2 for a given class C. Next, suppose that the continuous attribute a takes a value v for a given example which needs to be classified. The conditional probability $P(a = v|C)$ can be calculated in the following way:

$$P(a = v|C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} e^{-\frac{(v-\mu_C)^2}{2\sigma_C^2}}.$$

In the second approach we use discretization of continuous attributes. In a pre-processing step we discretize

continuous data into partitions of equal lengths. The problem is thus to identify the number of classes for discretization of continuous values, as the classifier accuracy greatly depends on the number of classes used for attribute discretization. We have decided to implement a genetic algorithm whose task is to find an optimal number of classes for discretization. The fitness function being optimized by the algorithm is the classifier accuracy. The algorithm chooses a number of classes used for discretization and checks the accuracy achieved with the selected number of classes based on k-fold cross validation. In implementation we use steady-state genetic algorithm. Each chromosome is represented as a bit string. We used uniform crossover for the crossover operator, bit inversion for the mutation operator and 3-way tournament selection for the selection operator. The genetic algorithm enables us to find approximately the best accuracy that can be achieved by discretization of continuous attributes.

3.2. Decision trees

Decision tree learning is a non-parametric method for supervised learning where a learned function is represented by a decision tree. A learned tree can be converted into a set of if-then rules which are human-readable. A tree consists of nodes, branches, and leaves, where nodes represent an attribute value test, branches correspond to attribute values, and leaves represent class labels. A widely used algorithm for generating decision trees is ID3 and its extension C4.5 developed by Ross Quinlan [13]. ID3 is a greedy algorithm which builds a decision tree by recursively splitting training data into subsets based on an attribute value test. The attribute value test used by ID3 is information gain defined as the expected reduction in entropy caused by partitioning the data according to the attribute [19]. Thus in each iteration, ID3 splits the data on the attribute with the highest information gain. Inductive bias of ID3 is the following: preference is given to shorter trees and trees that place attributes with a high information gain near the root.

We slightly modified the ID3 algorithm such that it can handle multi-valued attributes. Our approach for handling continuous attributes corresponds to the idea proposed in the C4.5 algorithm.

Instead of using information gain, we use gain ratio [19]. The problem with information gain is that it gives preference to attributes with a large number of possible values. Such attributes have the highest information gain because they split the training data into very small subsets and can predict the class label over the training data almost perfectly, which leads to overfitting. Since in our use case there are some attributes which can take a large number of distinct values (for example, the attribute "actor"), we want to use a measure which penalizes such attributes, i.e., the gain ratio. The gain ratio of attribute A with respect to set of training examples D is defined in the following way:

$$\mathit{GainRatio}\,(D, A) = \frac{\mathit{InformationGain}\,(D, A)}{\mathit{SplitInformation}\,(D, A)},$$

where *SplitInformation* penalizes the selection of attributes with a large number of distinct values.

Split information and information gain of attribute A with respect to the set of training examples D are defined as follows:

$$\mathit{SplitInformation}\,(D, A) = -\sum_{v\epsilon\ \mathit{Values}\ (A)} \frac{|D_v|}{|D|} \log_2 \frac{|D_v|}{|D|}$$

and,

$$\mathit{InformationGain}\,(D, A) = \mathit{Entropy}\,(D) - \sum_{v\epsilon\ \mathit{Values}\ (A)} \frac{|D_v|}{|D|}\ \mathit{Entropy}\ (D_v),$$

where D_v is a subset of D for which attribute A has value v and *Values* (A) is the set of all possible values for attribute A.

3.2.1. Multi-valued attributes

When building a decision tree, each value from the set of possible values of a multi-valued attribute is observed as a distinct value and does not depend on other values in the set of values. Since every value is observed individually, for every value from the set of values a distinctive branch is created. Consequently, the equations for calculating the information gain and split information need to be slightly modified. Namely, if all attributes are single-valued, then the following holds:

$$\sum_{v\epsilon\ \mathit{Values}\ (A)} \frac{|D_v|}{|D|} = 1,$$

because the summation goes through all attribute values and every attribute has taken one of the possible values. But if there are multi-valued attributes, this condition does not hold and the equations for information gain and split information will not be consistent for all attributes.

Thus, the equations are modified in the following manner: $|D|$ does not correspond to the total number of learning examples, but to the total number of all potential values associated with an attribute. Intuitively, this corresponds to modifying the set of the learning examples as follows. For example, suppose that there is one learning example where multi-valued attribute a has taken n vales. Then, from the original learning example, n new learning examples are generated which differ only in the value of attribute a. The process is repeated for each learning example. The cardinality of the newly generated learning example set for attribute a corresponds to $|D|$ in the equations for calculating information gain and split information. Also, it is obvious that $|D|$ is not unique anymore as it depends on the number of values taken by multi-valued attributes. With proposed modification, the following applies to the previous equation:

$$\sum_{v\epsilon \; \mathit{Values}\,(A)} \frac{|D_v|}{|D_A|} = 1,$$

where $|D_A|$ is the total number of values an attribute A has taken, i.e., the set cardinality generated as explained above.

In the prediction phase, when classifying an example and testing a multi-valued attribute specified by a decision tree node, each attribute value is sorted to the corresponding descendant nodes. Therefore, the result is a single subtree for each value from the set of values of a multi-valued attribute. The process of moving down the tree branches and attribute value testing is repeated for each subtree independently. The process ends when the leaf is reached in each subtree and the final prediction is calculated as the mean of predictions for all subtrees.

3.2.2. Continuous attributes

Algorithm C4.5 introduces an approach for handling continuous attributes [19]. For each continuous attribute it defines a new Boolean attribute that is true if an attribute value is less than a predefined threshold and false otherwise. The idea is to choose an attribute value for the threshold that achieves the highest information gain. The threshold is found as follows. Attributes values are sorted in an increasing order to identify adjacent values that differ in their target classification. Such values are chosen as threshold candidates and the candidates are then evaluated by computing the information gain. The candidate with the highest information gain becomes the threshold.

4. Experiments and results

We evaluate the described algorithms using the MovieLens data set [9] which is freely available for research purposes from the MovieLens web site by GroupLens Research at the University of Minnesota. We use the data set which contains 943 users, 1628 movies and 100,000 ratings. Ratings are in the range from 1 to 5. Movie information which this data set contains includes a movie title, year and genres. We have retrieved additional movie details required for content-based recommendations through the Open Movie Database API (OMDB API) [2] and Internet Movie Database API (IMDB API) [6]. Both APIs provide responses in XML and JSON format. Movie details can be fetched using a title and year of production or an IMDB movie id. Using OMDB API, the information about actors, directors, writers, duration, IMDB rating, number of IMDB votes is retrieved. IMDB API is used only for retrieving information about languages and countries. All the retrieved data, including movie genre and year of production (already available in the original MovieLens data set), is used for learning classifiers for the content-based algorithms. A new Boolean attribute is defined for each genre. Attributes actors, directors, writers, languages and countries are multi-valued attributes while attributes year, duration, IMDB rating and number of IMDB votes are continuous attributes. All these attributes types are handled as described in Section 3.

We use two performance measures to compare our algorithms. The first one is algorithm accuracy defined as the percentage of correctly classified examples. The second measure is the mean absolute error defined as an average of the absolute value of a difference between prediction and excepted outcome.

We evaluate the described algorithms for the selected representative users: 1) user with the least number of rated movies (the total of 19 movies), 2) user with the greatest number of rated movies (737 movies) and 3) four users with an average number of rated movies (106 movies). Thus, the evaluation is performed on six users in total.

We use leave-one-out cross validation for algorithm evaluation. For the content-based approach we evaluate the following algorithms: the naïve Bayes classifier which models continuous attributes with Gaussian distribution, the naïve Bayes classifier which discretizes continuous attributes where the number of classes for discretization is learned with a genetic algorithm and the decision trees algorithm, ID3 algorithm modified for handling multi-valued and continuous attributes.

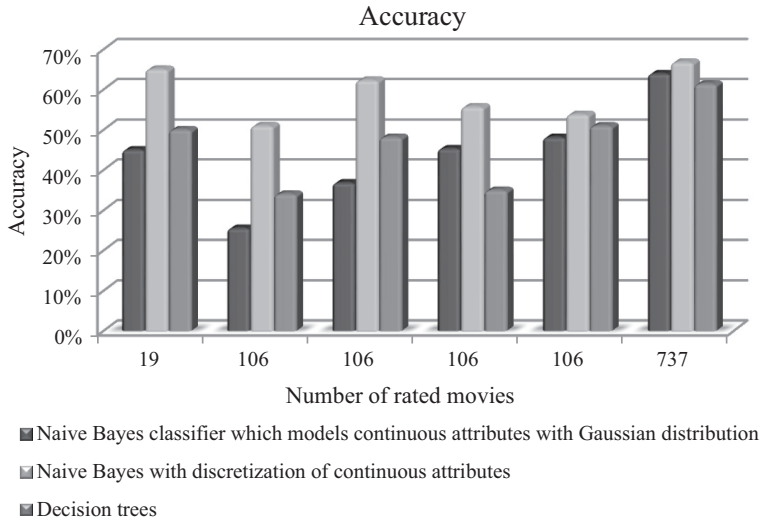

Fig. 1. Algorithm accuracy of the content-based approach.

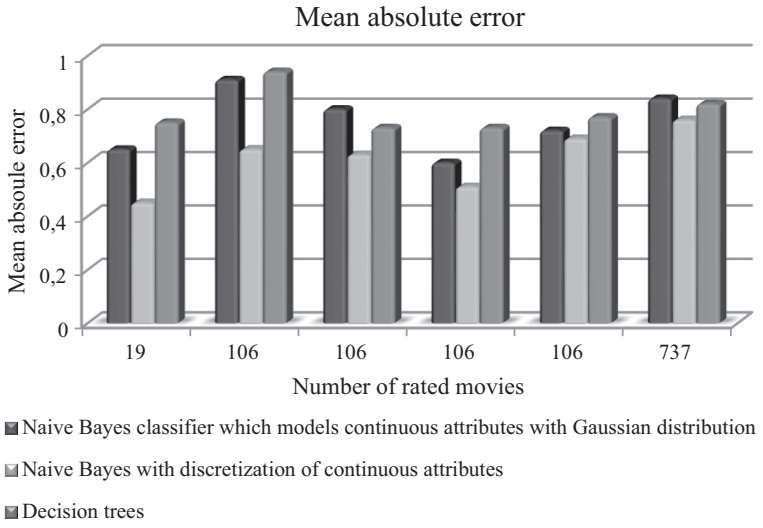

Fig. 2. Mean absolute error of the content-based approach.

Figure 1 shows the classification accuracy of the implemented algorithms for the content-based approach, while Fig. 2 shows the mean absolute error of these algorithms.

The results shown in Figs 1 and 2 demonstrate that the naïve Bayes algorithm with discretization where the number of classes is learned with a genetic algorithm outperforms the other two classifiers. The accuracy of the naïve Bayes algorithm with continuous attributes discretization varies between 51% to 67% and the mean absolute error is in the range from 0.45 to 0.76. It is evident that for all classifiers both the accuracy and mean absolute error for users with the same number of rated movies can vary a lot. The reason is simply that some users behave "typically" and it is a lot easier to learn their preferences compared to users whose behavior is unusual and harder to predict. The accuracy of the decision trees classifier varies from 34% to 61%, while the precision of the naïve Bayes classifier which models continuous attributes with Gaussian distribution varies from only 25% to 64% for the user with 737 rated movies. Thus, the performance of the naïve Bayes classifier greatly depends on the approach handling continuous attributes. Results also show that for most users the decision trees classifier is more accurate than the naïve Bayes classi-

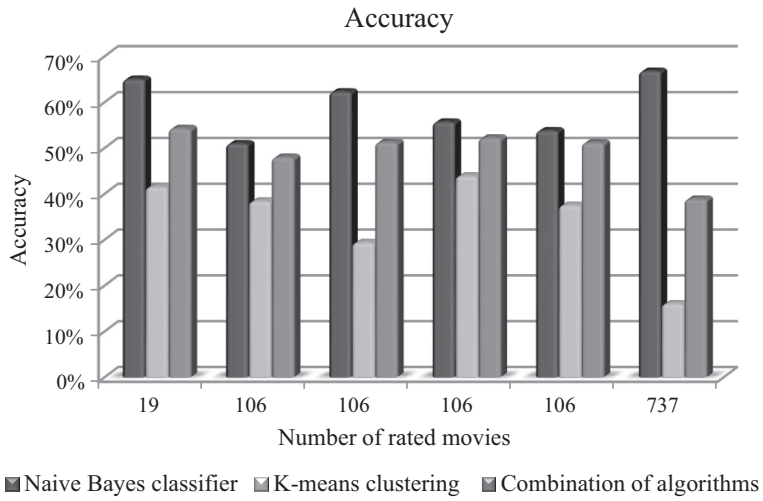

Fig. 3. Accuracy of the content-based, collaborative and hybrid approach.

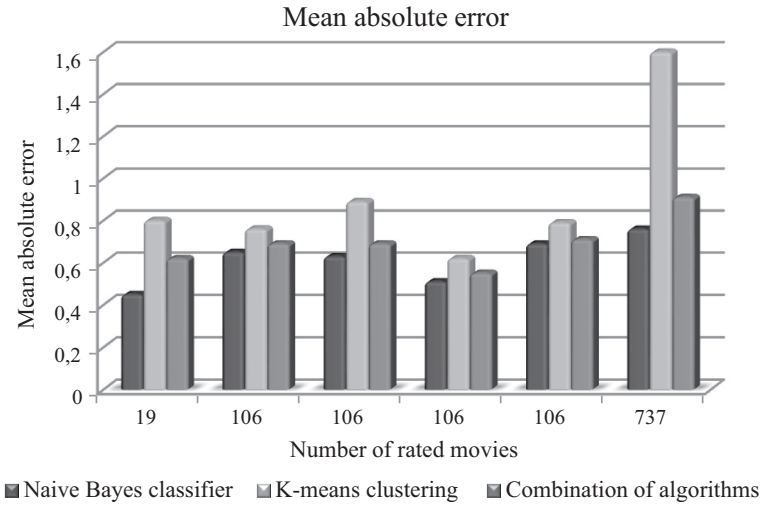

Fig. 4. Mean absolute error of the content-based, collaborative and hybrid approach.

fier which models continuous attributes with Gaussian distribution.

Figure 3 compares the accuracy of our best-performing content-based approach, collaborative approach and hybrid approach, while Fig. 4 shows the mean absolute errors. The best performing content-based algorithm is the naïve Bayes classifier with discretization of continuous attributes where the number of classes for discretization is learned with a genetic algorithm. The collaborative approach uses the k-means clustering algorithm, while the hybrid approach combines the previously mentioned algorithms.

The k-means clustering algorithm which we use for the collaborative approach is very similar to the k-medians algorithm described in [3]. The only difference is that we use Euclidian distance function instead of the absolute distance function and the mean of user ratings instead of their median, as originally proposed in [3]. We have evaluated the performance of the algorithm for different number of user clusters: 5, 10, 20, and 40. The best results were achieved with ten clusters and thus we have decided to take ten clusters when comparing the k-means clustering algorithm to other algorithms. Since the initial centroids were chosen randomly, we ran 20 algorithm trials both for the collaborative and hybrid approach. When evaluating the clustering algorithm performance for a user, the model which results in calculated centroids is learned from

the data incorporating all users except one, and then the leave-one-out cross validation is used to calculate the accuracy and mean absolute error. In our hybrid approach we take the mean of the rating predictions of the collaborative and content-based approach to predict a user rating.

Figures 3 and 4 show that the naïve Bayes classifier with discretization of continuous attributes shows significantly higher accuracy and lower mean absolute error than the k-means clustering algorithm. In addition, the naïve Bayes classifier also outperforms the hybrid approach. However, the main advantage of the collaborative approach compared to the content-based approach is greater diversity of recommendations since the content-based approach has the problem of overspecialization, i.e., a user is getting similar recommendations. Our results show that the accuracy of the collaborative approach can be significantly increased if we take into account also the content-based item characteristics.

5. Conclusion

In content-based recommendation systems it is often the case that items are described by attributes with continuous values and/or multi-valued attributes. In this paper we have introduced some modifications to the standard naïve Bayes classifer and decision trees classifer in order to handle multi-valued and continuous attributes for a movie recommendation. We have compared two approaches for handling continuous attributes for the naïve Bayes classifer: modeling of continuous attributes with a Gaussian distribution and discretization of continuous attributes when the optimal number of classes for discretization is determined by use of a genetic algorithm. The experimental results obtained by processing the MovieLens data set where movies details are retrieved from the IMDB have shown that the approach with the discretization of continuous attributes outperforms the approach which models continuous attributes with a Gaussian distribution. Also, the experiments have shown that the naïve Bayes classifier with discretization of continuous attributes can achieve greater accuracy compared to the decision trees classifier. We have implemented a hybrid recommender system combining the k-means clustering algorithm and our best performing algorithm, naïve Bayes which discretizes continuous attributes. Although the collaborative approach ensures greater diversity of recommendations, experiments have shown that our content based approach has significantly greater accuracy compared to a purely collaborative approach. It also performs better than the combination of algorithms implemented in the hybrid approach. According to these initial experimental results which are very promising, this algorithm can achieve quite accurate recommendations. It is important to emphasize that altough this algorithm achieves high accuracy and small mean absolute error, it is computationally expensive since it requires the use of a genetic algorithm for attribute discretization.

References

[1] A.S. Lampropoulos, P.S. Lampropoulou and G.A. Tsihrintzis, A Cascade-Hybrid Music Recommender System for Mobile Services based on Musical Genre Classification and Personality Diagnosis, *Multimedia Tools and Applications* **59**(1) (2012), 241–258.

[2] B. Fritz, The OMDB API, http://omdbapi.com/.

[3] B. Marlin, *Collaborative Filtering: A Machine Learning Perspective*, M.S. thesis, Department of Computer Science, University of Toronto, 2004.

[4] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering, *The Proceedings of the Fifth International Conference on Computer and Information Technology*, 2002.

[5] C. Christakou and A. Stafylopatis, A hybrid movie recommender system based on neural networks, in: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, 2005, pp. 500–505.

[6] D. Clatworthy, IMDB API, http://deanclatworthy.com/imdb/.

[7] D. Nikovski and V. Kulev, Induction of compact trees for personalized recommendation, in: *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 575–581.

[8] E.A. Eyjolfsdottir, G. Tilak and N. Li, *MovieGEN: A Movie Recommendation System*, University of California Santa Barbara: Technical Report, 2010.

[9] GroupLens Research at the University of Minnesota: MovieLens Datasets, http://www.grouplens.org/node/73.

[10] G. Adomavicius and A. Tuzhilin, Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, *IEEE Transactions on Knowledge and Data Engineering* **17**(6) (2005), 734–749.

[11] G.A. Tsihrintzis, M. Virvou and L.C. Jain, eds, Multimedia Services in Intelligent Environments- Recommendation Services, in: *Smart Innovation, Systems and Technologies (SIST) Book Series*, (Vol. 25), Springer 2013.

[12] International Movie Database, http://www.imdb.com/.

[13] J.R. Quinlan, Induction of Decision Trees, *Machine Learning* **1** (1986), 81–106.

[14] L.H. Ungar and D.P. Foster, Clustering Methods for Collaborative Filtering, in: *AAAI Workshop on Recommendation Systems*, 1998.

[15] M.J. Pazzani and D. Billsus, Content-based Recommendation Systems, in: *The Adaptive Web: Methods and Strategies of Web Personalization*, (Vol. 4321), Springer, 2007, pp. 325–341.

[16] M.J. Pazzani and D. Billsus, Learning and Revising User Profiles: The identification of interesting web sites, *Machine Learning* **27** (1997), 313–331.
[17] R. Burke, Hybrid recommender systems: Survey and experiments, *User Modeling and User-Adapted Interaction* **12**(4) (2002), 331–370.
[18] R.J. Mooney, P.N. Bennett and L. Roy, Book Recommending Using Text Categorization with Extracted Information, in: *The AAAI-98/ICML-98 Workshop on Learning for Text Categorization and the AAAI-98 Workshop on Recommender Systems*, Madison, WI, July 1998.
[19] T. Mitchell, Machine Learning, McGraw Hill, 1997.