A Dynamic Web Recommender System Using Hard and Fuzzy K-Modes Clustering

Panayiotis Christodoulou[1], Marios Lestas[2], and Andreas S. Andreou[1]

[1] Department of Electrical Engineering / Computer Engineering and Informatics, Cyprus University of Technology
{panayiotis.christodoulou,andreas.andreou}@cut.ac.cy
[2] Department of Electrical Engineering, Frederick University of Cyprus
eng.lm@frederick.ac.cy

Abstract. This paper describes the design and implementation of a new dynamic Web Recommender System using Hard and Fuzzy K-modes clustering. The system provides recommendations based on user preferences that change in real time taking also into account previous searching and behavior. The recommendation engine is enhanced by the utilization of static preferences which are declared by the user when registering into the system. The proposed system has been validated on a movie dataset and the results indicate successful performance as the system delivers recommended items that are closely related to user interests and preferences.

Keywords: Recommender Systems, Hard and Fuzzy K-Modes Clustering.

1 Introduction

Recommender Systems (RS) have become very common on the World Wide Web, especially in e-Commerce websites. Every day the number of the listed products increases and this vast availability of different options creates difficulties to users of finding what they really like or want. RS can be seen as smart search engines which gather information on users or items in order to provide customized recommendations back to the users by comparing each other [1]. Although RS give a strategic advantage they present some problems that have to be dealt with. Different techniques and algorithms are continuously being developed aiming at tackling these problems.

Existing strategies, despite their wide availability, come with problems or limitations related to the adopted architecture, the implementation of new algorithms and techniques and the considered datasets. For example, some existing RS make recommendations for the interested user utilizing static overall ratings which are calculated based on the ratings or preferences of all other users of the system. Other systems recommend items to the current user based on what other users had bought after they viewed the searched item.

The system proposed in this work makes recommendations based on the preferences of the interested user, which are dynamically changed taking into account previous searches in real time. This approach is enhanced by the utilization of static

H. Papadopoulos et al. (Eds.): AIAI 2013, IFIP AICT 412, pp. 40–51, 2013.

preferences which are declared by the user when registering into the system. The clustering procedure, being the heart of the recommendation engine, is of particular importance and a number of techniques such as Entropy-Based, Hard K-modes and Fuzzy K-modes have been utilized. The proposed system has been tested using the MovieLens1M dataset, which was linked with IMDB.com to retrieve more content information. The final results indicate that the proposed system meets the design objectives as it delivers items which are closely related to what the user would have liked to receive based on how s(he) ranked the different categories depending on what (s)he likes more and her/his previous behavior.

The remainder of the paper is organized as follows: Section 2 provides an overview of the clustering techniques that were used by the proposed system and discusses related work on the topic. Subsequently, section 3 describes the notions behind the technical background of the proposed system. Section 4 focuses on the way the system works, describing the structure of the dataset and discussing briefly the intelligent algorithms that were designed and used. This section also illustrates the experiments carried out followed by a comparison between the two clustering techniques. Finally, section 5 offers the conclusions and some future research steps.

2 Literature Overview

Recommender Systems (RS) are of great importance for the success of Information Technology industry and in e-Commerce websites and gain popularity in various other applications in the World Wide Web [1]. RS in general help users in finding information by suggesting items that s(he) may be interested in thus reducing search and navigation time and effort. The recommendation list is produced through collaborative or content-based filtering. Collaborative filtering tries to build a model based on user past behavior, for example on items previously purchased or selected, or on numerical ratings given to those items; then this model is used to produce recommendations [10]. Content-based filtering tries to recommend items that are similar to those that a user liked in the past or is examining in the present; at the end a list of different items is compared with the items previously rated by the user and the best matching items are recommended [11].

The work described in [5] discusses the combination of collaborative and content-based filtering techniques in a neighbor-based prediction algorithm for web based recommender systems. The dataset used for this system was the MovieLens100K dataset consisting of 100000 ratings, 1682 movies and 943 users, which were also linked with IMDB.com to find more content information. The final results showed that the prediction accuracy was strongly dependent on the number of neighbors taken into account and that the item-oriented implementation produced better prediction results than the user-oriented. The HYB-SVD-KNN algorithm described in this work was four times faster than the traditional collaborative filtering.

Ekstrand and Riedl [6] presented an analysis of the predictions made by several well-known algorithms using the MovieLens10M dataset, which consists of 10 million ratings and 100000 tag applications applied to 10000 movies by 72000 users.

Users were divided into 5 sets and for each user in each partition 20% of their ratings were selected to be the test ratings for the dataset. Therefore, five recommender algorithms were run on the dataset and the predictions of each algorithm for each test rating were captured. The results showed that the item-user mean algorithm predicted the highest percentage of ratings correctly compared to the other algorithms. This showed that the algorithms differed in the predictions they got wrong or right. Item-item got the most predictions right but the other algorithms correctly made predictions of up to 69%.

The work presented in [7] described long-tail users who can play an important role of information sources for improving the performance of recommendations and providing recommendations to the short-head users. First, a case study on MovieLens dataset linked with IMDB.com was conducted and showed that director was the most important attribute. In addition, 17.8% of the users have been regarded as a long tail user group. For the proposed system 20 graduated students were invited to provide two types of recommendations and get their feedbacks. This resulted in 8 users out of 20 to be selected as a long tail user group (LTuG), two times higher than the MovieLens case study. Finally, it was concluded that the LTuG+CF outperformed CF by 30.8% higher precision and that user ratings of the LTuG could be used to provide relevant recommendations to the short head users.

The work of McNee et al.[8] suggested that recommender systems do not always generate good recommendations to the users. In order to improve the quality of the recommendations, that paper argued that recommenders need a deeper understanding of users and their information. Human-Recommender Interaction is a methodology of analyzing user tasks and algorithms with the end goal of getting useful recommendation lists and it was developed by examining the recommendation process from an end user's respective. HRI is consisted of three pillars: The Recommendation Dialog, the Recommender Personality and the User Information seeking Tasks and each one contains several aspects.

Karypis et al. [9] presented a class of item-based recommendation algorithms that first determine the similarities between the various items and then use them to identify the set of items to be recommended. In that work two methods were used: The first method modeled the items as vectors in the user space and used the cosine function to measure the similarity between the items. The second method combined these similarities in order to compute the similarity between a basket of items and a recommender item. Five datasets were tested in the experimental part and the results showed that the effect of similarity for the cosine-based scheme improved by 0% to 6.5% and for the conditional probability based by 3% to 12%. The effect of row normalization showed an improvement of 2.6% for the cosine-based and 4.2% for the probability. The model size sensitivity test showed that the overall recommendations accuracy of the item-based algorithms does not improve as we increase the value of k. Finally they concluded that the top-N recommendation algorithm improves the recommendations produced by the user-based algorithms by up to 27% in terms of accuracy and it is 28 times faster.

This paper describes the design and implementation of a new dynamic Web Recommender System using Hard and Fuzzy K-modes clustering. The system provides recommendations based on user preferences that change in real time taking also into account previous searching and behavior and based on static preferences which are declared by the user when registering into the system.

3 Technical Background

3.1 Entropy-Based Algorithm

The Entropy-based algorithm groups similar data objects together into clusters based on data objects entropy values using a similarity measure [2]. The entropy value H_{ij} of two data objects X_i and X_j is defined as follows:

$$H_{ij} = E_{ij} \log_2(E_{ij}) - (1 - E_{ij}) \log_2(1 - E_{ij}) \tag{1}$$

where i ≠ j.

E_{ij} is a similarity measure between the data objects X_i and X_j and is measured using the following equation:

$$E_{ij} = e^{-aD_{ij}} \tag{2}$$

where D_{ij} is the distance between X_i and X_j and a is calculated by

$$a = \frac{-\ln(0.5)}{\overline{D}} \tag{3}$$

and $\overline{D}$ is the mean distance among all the data objects in the table. From Equation (1) the total entropy value of X_i with respect to all other data objects is computed as:

$$H_i = -\sum_{\substack{j=1 \\ i \neq k}}^{n} [E_{ij} \log_2(E_{ij}) - (1 - E_{ij}) \log_2(1 - E_{ij})] \tag{4}$$

The Entropy-based algorithm passes through the dataset only once, requires a threshold of similarity parameter β and is used to compute the total number of clusters in a dataset, as well as to find the locations of the cluster centers [3]. More specifically, the algorithm consists of the following steps:

1. Select a threshold of similarity β and set the initial number of clusters $c = 0$.
2. Determine the total entropy values H for each data object X as shown by Equation (4).
3. Set $c = c + 1$.

4. Select the data object $X_{\min}$ with the least entropy $H_{\min}$ and set $Z_c = X_{\min}$ as the c_{th} cluster centre.
5. Remove $X_{\min}$ and all data objects having similarity β.
6. If X is empty then stop, otherwise go to step 3.

The Entropy-based algorithm was used in this paper to compute the number of clusters in the dataset, as well as to find the locations of the cluster centers.

3.2 Hard K-Modes Clustering

The K-Modes algorithm extends the K-means paradigm to cluster categorical data by removing the numeric data limitation imposed by the K-means algorithm using a simple matching dissimilarity measure or the hamming distance for categorical data objects or replacing means of clusters by their mode [4].

Let X_1 and X_2 be two data objects of X_1 defined by m attributes. The dissimilarity between the two objects is stated as:

$$d(X_1, X_2) = \sum_{j=1}^{m} \delta(x_{1j}, x_{2j}) \tag{5}$$

where:

$$\delta(x_{1j}, x_{2j}) = \begin{cases} 0, x_{1j} = x_{2j} \\ 1, x_{1j} \neq x_{2j} \end{cases} \tag{6}$$

In the case of Hard K-Modes clustering, if object X_i in a given iteration has the shortest distance with center Z_l, then this is represented by setting the value of the nearest cluster equal to 1 and the values of the other clusters to 0.
For $a = 0$:

$$w_{li} = \begin{cases} 1, \text{if } d(Z_l, X_i) \leq d(Z_{h,} X_i), 1 \leq h \leq k \\ 0, \text{otherwise} \end{cases} \tag{7}$$

In the above equation w_{li} is the weight degree of an object belonging to a cluster.

3.3 Fuzzy K-Modes Clustering

Fuzzy K-Modes algorithm is an extension of the Hard K-Modes algorithm and it was introduced in order to incorporate the idea of fuzziness and uncertainty in datasets [3].

For $a > 1$:

$$w_{li} = \begin{cases} 1, & \text{if } X_i = Z_l \\ 0, & \text{if } X_i = Z_h, h \neq 1 \\ \dfrac{1}{\sum_{h=1}^{k} \left[\dfrac{d(Z_l, X_i)}{d(Zh, X_i)} \right]^{1/a-1}}, & \text{if } X_i \neq Z_l \text{ and } X_i \neq Z_h, 1 \leq h \leq k \end{cases} \quad (8)$$

If a data object shares the same values of all attributes within a cluster then it will be assigned entirely to that cluster and not to the others. If a data object in not completely identical to a cluster then it will be assigned to each cluster with a membership degree.

4 Methodology and Experimental Results

4.1 Proposed System

Figure 1 shows a schematic representation of the proposed system. First the user registers into the system and ranks the different categories depending on what (s)he likes more using a weight ranking system. The ranking of the categories is called static information because this type of information can only be changed after a certain period of time when the user will be prompted by the system to update her/his rankings as their interest in certain categories may have changed. The system requires a certain number of searches to be conducted first so as to understand the user behavior (dynamic information) and then it starts recommending items. A dynamic bit-string is created after the first searches and is updated with every new search. This string is compared with each movie in the dataset to eliminate those movies that the user is not interested in depending on search profile thus far. If there is at least one 1 in the compared bit string then the movie moves is inserted into a lookup table. After that the system creates the clusters depending on the new dataset size (i.e. the movies in the lookup table) and the entropy threshold similarity value β which is assumed to be constant; however, its value needs to be tuned based on the size of the dataset in order to reach optimal performance. The next step is the update of the clusters to include the static information of the user. This is performed so as to eliminate the problem encountered by the system after having a specific object belonging to two or more clusters. Therefore, the new clusters also include the static information depending on how the user ranks the categories.

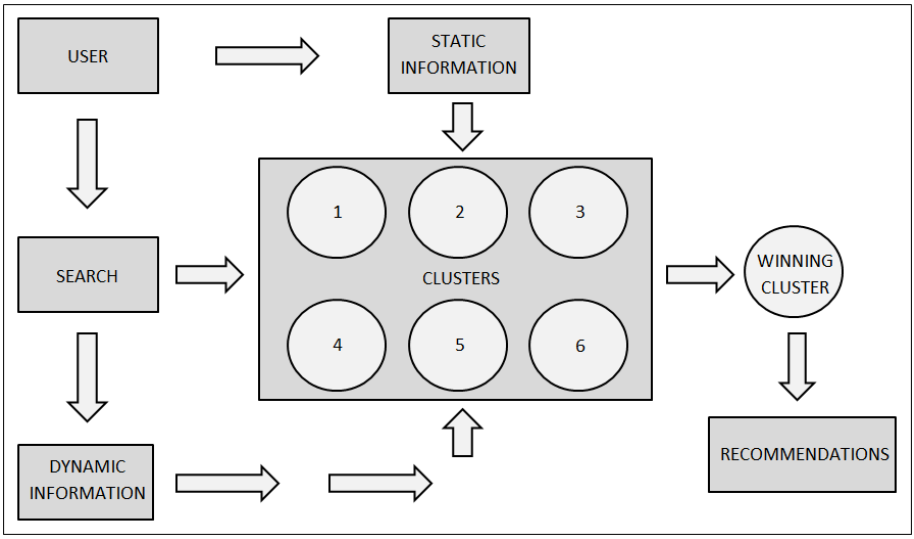

Fig. 1. How the proposed RS works

When the user performs searching queries the keyword used is compared with each cluster center and the proposed system finds the most similar one depending on the searched item (winning cluster); this cluster is then used to provide the recommendations. In the meantime the searched keyword is saved by the system as part of the dynamic information which is thus updated in real time.

4.2 Dataset

The dataset used in the proposed system is an extension of the Movie Lens 1M dataset that can be found at GroupLens.org. The Movie Lens 1M dataset is consisted of 1 million ratings from 6000 users on 4000 movies. The proposed system is not using the user ratings so we deal only with the movies. From the 4000 movies some movies are duplicated so we had to remove them thus concluding with a final dataset numbering a total of 3883 movies. Then we linked the final dataset with IMDB.com, the world's largest movie database, to retrieve more information about the categories of each movie.

Table 1 shows the different movie categories. In total there are 18 different categories. Therefore, the experimental dataset used for the encoding and testing of the proposed system was a matrix of 3883 movies in rows times 18 movie categories in columns.

4.3 Experimental Results

Three users with different characteristics that searched for various items were used to test the proposed recommendation schema on the movie dataset described in

section 4.1. As previously mentioned, the system recommended movies based on the Hard and Fuzzy K-Modes clustering. The different characteristics used by the system to predict accurate recommended items were: (i) the total number of the final clusters based on the threshold similarity value β, (ii) the ranking of the movie categories according to the user interests and, (iii) the past history of different searches that each of the users conducted.

Table 1. Movie Categories

Column	Movie Category
1	Animations
2	Children
3	Comedy
4	Adventure
5	Fantasy
6	Romance
7	Drama
8	Action
9	Crime
10	Thriller
11	Horror
12	Sci-Fi
13	Documentary
14	War
15	Musical
16	Mystery
17	Western
18	Film-Noir

The Root Mean Square Error (RMSE) was used as the evaluation metric to assess the accuracy of the results, which is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (x_{obs,i} - x_{\mathrm{mod}\,el,i})^2}{n}} \tag{9}$$

where $X_{obs,i}$ and $X_{model,i}$ are the observed and modeled values at the *i-th* sample respectively.

Table 2 shows how one of our users ranked the different movie categories. This information is inserted into the clusters and changes them to include the weight reflecting the interests of the user.

Table 2. UserA Movie Rankings

1. Drama	10. Crime
2. Adventure	11. Mystery
3. Animation	12. Thriller
4. Action	13. Documentary
5. War	14. Romance
6. Children	15. Film-Noir
7. Musical	16. Sci-Fi
8. Comedy	18. Horror
9. Western	19. Fantasy

Table 3 shows the ten first searches that UserA conducted in order for the system to understand his/her behavior by examining the movie categories searched more. If the value of a specific category exceeds a specific value, in our case this value is equal to three, then the system selects that category as a "frequently searched". In the case of UserA the categories searched more are Adventure and Drama. After the ten first searches the system starts to recommend items to the interested user based on how (s)he ranked the movie categories, what (s)he has searched more and the searched keywords. The analysis of the specific searches follows.

Table 3. UserA – Ten first searches

ID	Movie Title	Movie Categories
23	Assasins	Thriller
31	Dangerous Minds	Drama
86	White Squall	Adventure , Drama
165	The Doom Generation	Comedy , Drama
2	Jumanji	Adventure, Children, Fantasy
167	First Knight	Action, Adventure, Drama, Romance
237	A Goofy Movie	Animation, Children, Comedy, Romance
462	Heaven and Earth	Action, Drama, War
481	Lassie	Adventure, Children
1133	The Wrong Trousers	Animation, Comedy

Close inspection of the results listed in Table 4 reveals the following findings:

i. The proposed fuzzy algorithm is more accurate on average than the Hard implementation as it adjusts dynamically the knowledge gained by the RS engine.
ii. Both algorithms always suggest movies in ascending order of error magnitude, while there are also cases where movies bear the exact same RMS value; this is quite natural as they belong to the same cluster with the same degree of membership.
iii. The error depends on the category of the movie searched for each time; therefore, when this type perfectly matches the clustered ones the error is minimized.

Table 4. UserA recommendation results and evaluations per clustering method and search conducted – RMSE values below each method corresponds to the average of the five searches

Searches and Clustering Methods		Recommendations				
		1	*2*	*3*	*4*	*5*
#1: Dadetown Documentary	**Hard** 0.3333	Two Family House Drama 0.3333	Tigerland Drama 0.3333	Requime for a Dream Drama 0.3333	Remember the Titans Drama 0.3333	Girlfight Drama 0.3333
	Fuzzy 0.3333	Othello Drama 0.3333	Now and Then Drama 0.3333	Angela Drama 0.3333	Dangerous Minds Drama 0.3333	Restoration Drama 0.3333
#2: A Christmas Tale Comedy and Drama	**Hard** 0.0	Bootmen Comedy and Drama 0.0	Beautiful Comedy and Drama 0.0	Duets Comedy and Drama 0.0	A Knight in New York Comedy and Drama 0.0	Mr.Mom Comedy and Drama 0.0
	Fuzzy 0.0	Waiting to Exhale Comedy and Drama 0.0	To Die For Comedy and Drama 0.0	Kicking and Screaming Comedy and Drama 0.0	Big Bully Comedy and Drama 0.0	Nueba Yol Comedy and Drama 0.0
#3: Yellow Submarine Animation and Musical	**Hard** 0.4588	Digimon Adventure, Animation and Children 0.4082	For the Love of Benji Adventure and Children 0.4714	The Legend of Lobo Adventure and Children 0.4714	Tall Tale Adventure and Children 0.4714	Barneys Adventure and Children 0.4714
	Fuzzy 0.4059	Pete's Dragon Adventure, Animation, Children and Musical 0.3333	Gullivers Travels Adventure, Animation and Children 0.4082	Digimon Adventure, Animation and Children 0.4082	Bedknobs and Broomsticks Adventure, Animation and Children 0.4082	The Lord of the Rings Adventure, Animation, Children and Sci-Fi 0. 4714
#4: Young Guns Action, Comedy and Western	**Hard** 0.2576	Butch Cassidy Action, Comedy and Western 0.2576	Action Jackson Action and Comedy 0.2357	Last Action Hero Action and Comedy 0. 2357	Mars Attack Action, Comedy, Sci-Fi and War 0.4082	Tank Girl Action, Comedy, Musical, Sci-Fi 0.4082
	Fuzzy 0.2357	I Love Trouble Action and Comedy 0.2357	Beverly Hills Cop III Action and Comedy 0.2357	The CowBoy Way Action and Comedy 0.2357	Beverly Hills Ninja Action and Comedy 0.2357	Last Action Hero Action and Comedy 0.2357

Table 5 summarizes the mean RMS error values for four additional users tested on five different searches. Due to size limitations we omitted the details of the searched categories as these resemble the ones presented for UserA. It is once again clear that the algorithm behaves successfully, with average error values below 0.5 with only one exception (first search of UserB) and with consistently better performance being observed for the Fuzzy implementation.

Table 5. Summary of recommendation results and mean evaluations per clustering method for four more users

User	Method	Searches				
		1	*2*	*3*	*4*	*5*
B	Hard	0.5774	0.3480	0.4572	0.4557	0.2357
	Fuzzy	0.5360	0.3368	0.4082	0.4335	0.2552
C	Hard	0.3135	0.3333	0.3437	0.3714	0.2357
	Fuzzy	0.2552	0.2357	0.3437	0.2747	0.2357
D	Hard	0.0	0.4082	0.3333	0.3999	0.4461
	Fuzzy	0.0	0.4082	0.3333	0.3908	0.4082
E	Hard	0.3782	0.3610	0.3714	0.4885	0.2943
	Fuzzy	0.3782	0.3333	0.3333	0.4673	0.2943

5 Conclusions

Web Recommender Systems are nowadays a powerful tool that promotes e-commerce and advertisement of goods over the Web. High accuracy of recommendations, though, is not an easy task to achieve. This paper examined the utilization of the Hard and Fuzzy K-modes algorithms in the recommendation engine as a means to approximate the interests of users searching for items on the Internet. The proposed approach combines static information entered by the user in the beginning and dynamic information gathered after each individual search to perform clustering of te available dataset. Recommendations are given to the user by comparing only cluster centers with the search string thus saving execution time.

A series of synthetic experiments were executed to validate the RS system using the MovieLens1M dataset, which was linked with IMDB.com to retrieve more content information. More specifically, five different users with various interests on movies performed five different searches and the recommendations yielded by the Hard and Fuzzy K-Modes algorithms were assessed in terms of accuracy using the well-known RMS error. The results revealed small superiority of the Fuzzy over the Hard implementation, while recommendations were quite accurate.

Future work will concentrate on conducting more experiments and comparing with other approaches using collaborative or content-based filtering. In addition, a dedicated website will be developed through which real-world data will be gathered for experimentation purposes. Finally, the system will be enhanced with new functionality

which will prompt the user, after a certain period of time, to update her/his static information so as to generate more accurate recommendations as interest in certain categories may have changed.

References

1. de Nooij, G.J.: Recommender Systems: An Overview. MSc Thesis, University of Amsterdam (November 2008)
2. Principia Cybernetica Web, Entropy and Information, http://pespmc1.vub.ac.be/ENTRINFO.html
3. Stylianou, C., Andreou, A.S.: A Hybrid Software Component Clustering and Retrieval Scheme, Using an Entropy-based Fuzzy *k*-Modes Algorithm. In: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, Washinghton DC, USA, vol. 1, pp. 202–209 (2007) ISBN:0-7695-3015-X
4. Khan, S.S.: Computation of Initial Modes for K-modes Clustering Algorithm. In: Proceedings of the 20th International Joint Conference on Artificial Intelligent, San Francisco, USA, pp. 2784–2789 (2007)
5. Speigel, S., Kunegis, J., Li, F.: Hydra: A Hybrid Recommender System. In: Proceedings of the 1st ACM International Workshop on Complex Networks Meet Information & Knowledge Management, New York, USD, pp. 75–80 (2009)
6. Ekstrand, M., Riedl, J.: When recommender fail: predicting recommending failure for algorithm selection and combination. In: Proceedings of the 6th ACM Conference on Recommender Systems, New York, USA, pp. 233–236 (2012)
7. Jung, J.J., Pham, X.H.: Attribute selection-based recommendation framework for short-head user group: An empirical study by MovieLens and IMDB. In: Proceedings of the Third International Conference on Computational Collective Intelligence: Technologies and Applications, Berlin, Germany, pp. 592–601 (2011)
8. McNee, S.M., Riedl, J., Konstan, J.A.: Making Recommendations Better: An Analytic Model for Human- Recommender Interaction. In: Proceedings of the CHI 2006 Extended Abstracts on Human Factors in Computing Systems (2006)
9. Karypis, G.: Evaluation of Item-Based Top-N Recommendation Algorithms. In: Proceedings of the 10th International Conference on Information and Knowledge Management, New York, USA, pp. 247–254 (2001)
10. Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems Handbook. Springer (2011) ISBN-13: 978-0387858197
11. Van Meteren, R., van Someren, M.: Using Content-Based Filtering for Recommendation. In: Proceedings of the ECML 2000 Workshop: Machine Learning in New Information Age, Barcelona, Spain, pp. 47–57 (2000)