MovieGEN: A Movie Recommendation System

Eyrun A. Eyjolfsdottir, Gaurangi Tilak, Nan Li

Computer Science Department, University of California Santa Barbara
eyrun@cs.ucsb.edu
gaurangi_tilak@cs.ucsb.edu
nanli@cs.ucsb.edu

Abstract

In this paper we introduce MovieGEN, an expert system for movie recommendation. We implement the system using machine learning and cluster analysis based on a hybrid recommendation approach. The system takes in the users' personal information and predicts their movie preferences using well-trained support vector machine (SVM) models. Based on the SVM prediction it selects movies from the dataset, clusters the movies and generates questions to the users. Based on the users' answers, it refines its movie set and it finally recommends movies for the users. In the process we traverse the parameter space, thus enabling the customizability of the system.

Introduction

Consumers today in the world with the internet and its associated information explosion are faced with the problem of too much choice. Right from looking for a restaurant to looking for good investment options, there is too much information available. To help the consumers cope with this information explosion, companies have deployed recommendation systems to guide the consumer. The research in the area of recommendation systems has been going on for several decades now, but the interest still remains high because of the abundance of practical applications and the problem rich domain. A number of such online recommendation systems are implemented and are in use such as the recommendation system for books at Amazon.com and Libra, for movies at MovieLens.org, CDs at CDNow.com (from Amazon.com), etc.

Recommendation systems are special types of expert systems in the sense that they combine the knowledge of the expert in a given domain (for the product type being recommended) with the user's preferences to filter the available information and provide the user with the most relevant information. Personalization of the recommendations works by filtering a candidate set of items (such as products or web pages) through some representation of a personal profile. Two main paradigms for the filtering are content-based approach and collaborative approach. Most recommendation systems use a hybrid approach, which is a combination of these two approaches. A content based recommendation system uses the user's past history to recommend new items where as a collaborative approach uses the preferences of other people with similar tastes for recommending items to the user.

We have developed MovieGEN, a movie recommendation system that recommends movies to users based on their personal information and their answers to questions based on movies. We use a user model created using SVM based learning techniques. Using this model we can predict the genres and the period of the movies that the user prefers based on the user's personal information. This incorporates the collaborative approach i.e. we predict the users choices based on the choices of other similar users. We implement a variation of the content based approach by taking into consideration the user choices not based on the user's past history but based on the answers he gives to the questions asked by the system. The system has been developed in Java and currently uses a simple console based interface.

Background and Related Work

Over the past decade, a large number of recommendation systems for a variety of domains have been developed and are in use. These recommendation systems use a variety of methods such as content based approach, collaborative approach, knowledge based approach, utility based approach, hybrid approach, etc.

Lawrence et al. 2001 describe a personalized recommendation system to suggest new products in supermarkets to shoppers based on their previous purchase behavior. This system developed at IBM research has been implemented as a part of SmartPad, a PDA based remote shopping system. This system uses content based filtering with collaborative filtering used to refine the recommendations.

Most of the online recommendation systems for a variety of items use ratings from previous users to make recommendations to current users with similar interests. One such system was designed by Jung, Harris, Webster and Herlocker (2004) for improving search results. The system encourages users to enter longer and more informative search queries, and collects ratings from users as to whether search results meet their information need or not. These ratings are then used to make recommendations to later users with similar needs.

MovieLens is an online movie recommendation system. When a user logins for the first time, the user is asked to rate certain movies that the user has seen. These ratings are then used to recommend other movies to the user that the user has not seen. It also uses collaborative filtering based on ratings by similar. These two approaches are combined to create personalized recommendations. The GroupLens Research Group has also created a version of MovieLens for use with a mobile device such as a PDA that is not always connected to the internet.

Another facet that can be used to create improved recommendations is the contextual information. This is especially true in case of recommendation systems for holiday plans, restaurants, etc. where other factors such as the time, location, companion, etc. also play an important role in the users choice. A multidimensional recommendation approach uses the contextual information along with the hybrid approach. Weng, Lin and Chen (2007) carried out an evaluation study to show that using additional customer profiles and using multidimensional analysis to find key factors affecting customer choices helps to increase the recommendation quality. They used the multidimensional recommendation model (MD recommendation model) proposed by Adomavicius and Tuzhilin (2001) as the foundation to establish a recommendation structure with multidimensional data collection and analysis ability to solve the movie recommendation problems.

Machine learning constitutes an essential step in our approach. It is a technique that simulates the way human brain operates to equip computers with intelligence. It has been extensively utilized in data mining and knowledge discovery domains. Within the current literature, machine learning techniques that have been extensively studied include artificial neural network (ANN) (Cai and Shi, 2003; Huang et al., 2006), SVM (Joachims, 1998; Sun, 2003; Tong and Chang; Trafalis, 2006; Vapnik, 1998; Wu et al., 2006; Zhang et al., 2003; Zhao and He, 2006), Markov model (Abramson and Cohen, 2006), decision trees (Zheng, 1998), inference rules (Turney, 2003), fuzzy network (Sun, 2003), Bayesian network (Gencay, 2001) etc. For any machine learning model, the data sets are composed of two parts, the input and the output. The output is usually the subjects of interest, in other words, the part that we want to predict or classify, while the input is the set of elements that might have impacts upon the output. Machine learning attempts to correlate the output and the input, by approximating functions in between whose formulations are unknown.

In order to compensate the insufficiencies of ANN, Vapnik proposed a new machine learning and data mining tool, support vector machine (Vapnik, 1998), which is based on statistical learning. SVM is able to overcome problems such as over-fitting and local minimum and is sufficient for high generalization (Sun, 2003), which renders its application into text classification (Joachims, 1998), image processing (Tong and Chang), time series analysis (Trafalis, 2006; Wu et al., 2006) etc. Zhang et al. (2003) utilized SVM for underwater acoustic targets classification and proposed a new SVM algorithm based on three conventional SVM algorithms, linear SVM, non-linear SVM and multi-class SVM. In the meanwhile, they compared the performance of SVM for this specific problem with other machine learning techniques, such as K-NN and ANN, and concluded that SVM is able to achieve the best results. Sun (2003) incorporated fuzzy logic into SVM and created a new multi-layer SVM, which was later applied into numerical regression problem. It was proved that the improved SVM was able to conquer the negative effects caused by outliers and noisy data. Zhao and He (2006) claimed that conventional SVM fails to give satisfying results for classification upon unbalanced data sets, and conducted comparison study between *v*-SVM and conventional SVM, together with correlation analysis between parameter *v* in *v*-SVM and parameter *C* in conventional SVM. Besides, SVM has been widely utilized in financial data time series forecasting as well (Trafalis, 2006).

K-means is used in our approach as the cluster analysis tool. K-means is one of the most widely-used partitioning methods in the data mining community, and has been studied and applied in a wide range of domains, including bioinformatics (Guralnik and Karypis, 2001; Zhong et al., 2005), pattern recognition (Estlick et al, 2001; Saegusa and Maruyama, 2007; Filho et. Al, 2003), text classification (Steinbach, 2000), etc. Zhong et al. (2005) proposed an improved version of the conventional K-means algorithm, which instead of randomly choosing the initial set of cluster centers, used greedy algorithm to form the initial cluster centers. This improved K-means algorithm was then applied to discover patterns out of protein sequences. They claimed that the new K-means algorithm achieved a better cluster result for protein pattern recognition problem, since it was able to find more intricate patterns that conventional K-means was not able to detect. Saegusa and Maruyama (2007) and Filho et al. (2003) utilized K-means in solving image processing problems. Tsai (2002) proposed a K-means based approach, namely the multiple-searching generic K-means algorithm (MSGKA), to cluster data in large-scale databases according to their attributes. Empirical studies indicated that MSGKA was capable of obtaining a smaller cluster error and a faster convergence speed compared to other conventional algorithms such as fast self-organizing (SOM) and conventional K-means. A new algorithm combining both K-means (partition clustering) and hierarchical clustering was introduced in (Chen et al., 2005), and was applied to two recently-published microarray data sets, both of which were from the Stanford

Microarray Databases. Xu and Franti (2004) presented a composite algorithm named as Heuristic K-means, which combined K-means with dynamic programming and nonlinear-principal-direction-based algorithm. Experiment results demonstrated that Heuristic K-means produced better cluster results for the given data sets than principal components analysis (PCA) and KD-tree algorithm.

Methodology

Machine Learning Based Preference prediction

Machine learning model constitutes an essential part in our expert system. As aforementioned, we utilize machine learning technique to predict movie preference for a user based on what he or she tells the system. The machine learning model of the system takes in the input from the user, quantifies it into a vector and then feeds this vector into the machine learning algorithm. A well trained machine learning model is then capable of predicting an output vector for this specific input vector. Eventually, the machine learning model transforms this output vector into humanly legible format so that it presents the predicted movie preference for the user.

SVM, an effective and efficient machine learning tool that has been extensively studied within the machine learning community, is utilized in this paper as the machine learning algorithm. As what has been described in previous sections, SVM is incorporated in our system to establish a correlation analysis between personal particulars of a user and his or her personal preference for movies. For each set of user input, a SVM is trained based on a predefined set of training samples, which increases in size after each time the system is used. Figure 1 visualizes the general procedure of machine learning mechanism implemented in our expert system.

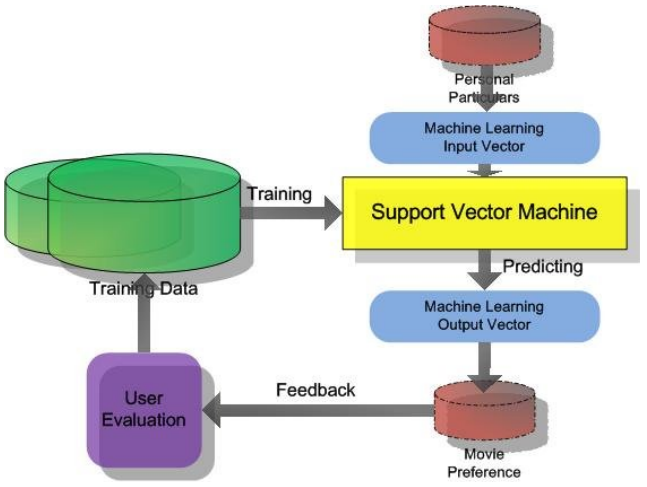

Figure 1: SVM based machine learning model in our movie recommendation system, taking in user personal particulars and predicting movie preference

Data Description

As aforementioned, machine learning is a process involving training and predicting. Before a machine learning model can be utilized to conduct analysis upon new data sets, it is indispensable to train the model, in other words, to equip it with intelligence, with the data that has already been collected. Those data serve as the prior knowledge of the specific domain of interest. Usually for supervised machine learning problems, the training data are composed of both input and output parts, the latter of which are in most cases the classification labels or the regression values of the training samples.

In our system, we have two sets of data, the training data set and the testing data set. The training data set comprises of different training samples, each of which is a combination of an input vector and an output vector. The testing data set comprises of different testing samples, each of which contains only an input vector, while the output vector is to be predicted by the machine learning. Feature extraction is a step of critical significance in our system. Based on feature extraction techniques, we quantify each sample into a vector of integer numbers.

Let $S_t(i)$ denote training sample i, then $S_t(i)$ can be represented as

$$S_t(i) = \{V_t^I(i), V_t^O(i)\}, \qquad (1)$$

where $V_t^I(i)$ and $V_t^O(i)$ stand for the input vector and the output vector for training sample i, respectively. $V_t^I(i)$ is a quantification of the user's personal particulars, whose structure can be shown as

$$V_t^I(i) = \begin{bmatrix} \mathrm{Age}(i) \\ \mathrm{Gender}(i) \\ \mathrm{Occupation}(i) \\ \mathrm{Area}(i) \\ \mathrm{Hobby}(i) \end{bmatrix}, \qquad (2)$$

where all the entries are integers. Age(i) is the age bracket of the user, Gender(i) is the gender of the user, Occupation(i) is the occupation category of the user, Area(i) is the area the user is working in and Hobby(i) represents the biggest hobby category the user has. In order to quantify personal traits into integers, we designed reference tables that correlate the above to integers. Table 1 shows the reference table we have used to quantify the input part of a sample.

Table 1: Mapping between personal particulars and predefined integers

Input Attribute	Value	Quantified Value
Gender	Male	1
	Female	2
Occupation	Academia	1

	Corporate	2
	Government	3
	Retired	4
	Entertainment	5
	NGO	6
	Business	7
Area	Science	1
	Art	2
	Social Science	3
	Finance	4
	Management	5
	Industry	6
Hobby	Sports	1
	Performing Art	2
	Fine Art	3
	Literature	4
	Socializing	5

Likewise, $V_t^O(i)$ is a quantification of the user's movie preference, whose structure can be shown as

$$V_t^O(i) = \begin{bmatrix} \text{Priority}^1(i) \\ \text{Priority}^2(i) \\ \text{Priority}^3(i) \\ \text{Year}(i) \end{bmatrix}, \tag{3}$$

where all entries are integers. $\text{Priority}^1(i)$, $\text{Priority}^2(i)$ and $\text{Priority}^3(i)$ represent the user's first, secondly and thirdly prioritized movie preference, respectively. Year(i) denotes the year of the movie the user likes. Similarly, we designed reference tables that correlate the above to integers. Table 2 shows the reference table we have used to quantify the output part of a sample.

Table 2: Mapping between movie preference and predefined integers

Output Attribute	Value	Quantified Value
Priority^j, $j = \{1, 2, 3\}$	Action	1
	Adventure	2
	Animation	3
	Comedy	4
	Crime	5
	Drama	6
	Family	7
	Fantasy	8
	History	9
	Horror	10
	Musical	11
	Mystery	12
	Romance	13
	Sci-Fi	14
	Thriller	15
	War	16
	Sport	17
Year	1950s	1
	1960s	2
	1970s	3
	1980s	4
	1990s	5
	2000s	6

Regarding testing data, we take in the input from one user each time. Let $S_p(j)$ denote testing sample j, then $S_p(j)$ can be represented as

$$S_p(j) = \{V_p^I(j)\}, \tag{4}$$

where $S_p(j)$ does not have output part, which will be predicted from the machine learning model.

SVM Regression

SVM regression constitutes the primary machine learning algorithm in our system. SVM is a supervised machine learning technique that has been pervasively used in classification and regression analysis.

As a branch of statistics theory, SVM is based on Vapik-Chervonemkis (VC) theory and structural risk minimization. Given a limited number of training samples, SVM is able to find an optimal hyper-plane that best classifies the data points in the data set, with the best generalization error. Details regarding SVM algorithm will not be covered here in this paper. Readers who are interested can refer to paper [17] about the underlying mathematics. Compared to traditional machine learning techniques, such as ANN, SVM has the following advantages:

(1) SVM overcomes the over-fitting problem exhibited by ANN due to the fact that it is based on the structural risk minimization theory and has a better generalization error.

(2) The essence of SVM is to solve a quadratic programming (QP) optimization problem under linear constraints. Therefore SVM is able to find the global optimal solution, which overcomes the local minimum problem in ANN.

In our system, we utilize SVM to conduct correlation analysis between the users' personal particulars and their movie preferences. This is implemented by SVM regression. Suppose we have N training samples, we can use a matrix tuple $<I, O>$ to denote the training data set, where I is the input data matrix for the SVM and O is the output data matrix for the SVM. Therefore we have

$$I = \begin{bmatrix} \text{Age}(1) & \text{Gender}(1) & \text{Occupation}(1) & \text{Area}(1) & \text{Hobby}(1) \\ \text{Age}(2) & \text{Gender}(2) & \text{Occupation}(2) & \text{Area}(2) & \text{Hobby}(2) \\ \text{Age}(3) & \text{Gender}(3) & \text{Occupation}(3) & \text{Area}(3) & \text{Hobby}(3) \\ \dots & & & & \\ \text{Age}(N) & \text{Gender}(N) & \text{Occupation}(N) & \text{Area}(N) & \text{Hobby}(N) \end{bmatrix}, \tag{5}$$

where each row represents a training sample. Similarly, we can have

$$O=\begin{bmatrix} \text{Priority}^1(1) & \text{Priority}^2(1) & \text{Priority}^3(1) & \text{Year}(1) \\ \text{Priority}^1(2) & \text{Priority}^2(2) & \text{Priority}^3(2) & \text{Year}(2) \\ \text{Priority}^1(3) & \text{Priority}^2(3) & \text{Priority}^3(3) & \text{Year}(3) \\ \cdots & & & \\ \text{Priority}^1(N) & \text{Priority}^2(N) & \text{Priority}^3(N) & \text{Year}(N) \end{bmatrix} .(6)$$

For testing data, each time there is one user who uses the system, so the number of testing data set is one. The difference lies in that the testing sample does not have a predefined output. The system predicts an output for the user.

Cluster Analysis Based Question Generation and Movie Recommendation

Once we predict the genres and period the user prefers based on his personal information using SVM, we use this information to select movies from the dataset, generate questions about these movies and finally return a refined movie recommendation to the user. Figure 2 shows the flow for getting to recommended movies from the input.

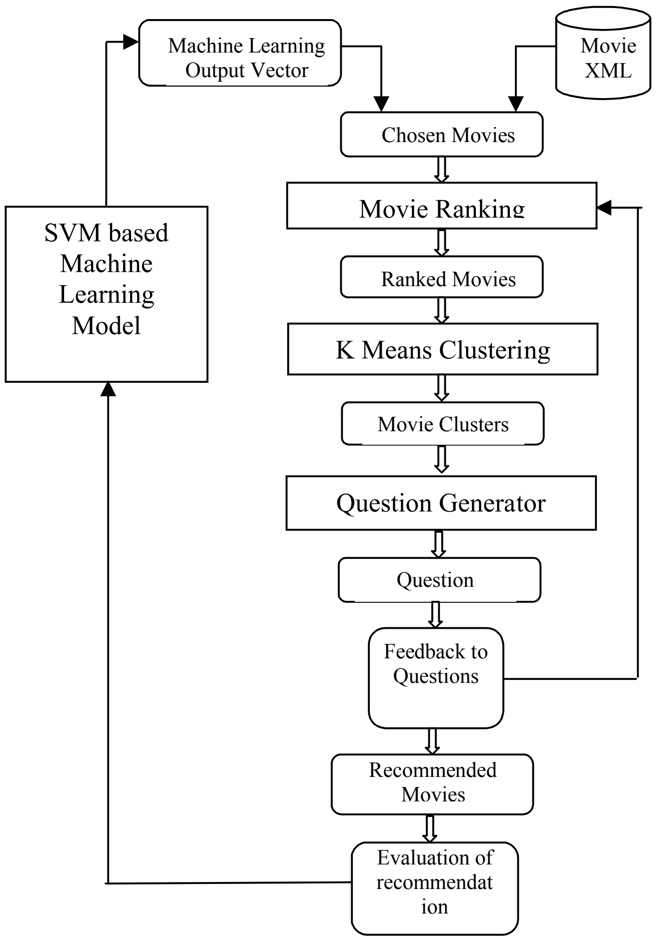

Figure 2: Clustering based recommendation, takes in the input from the SVM and generates the recommendations by asking questions to the user

Choosing and Ranking

We query our dataset i.e. the XML file of movies and select the movies that have at least one of the genres that the user likes as predicted by the SVM. We then rank all the movies we get based on the user's preference information. The genre with the highest preference contributes 3 points to the rank, the next one 2 points and the third one 1 point. If the movie is from the decade of the user's preference it receives additional 2 points.

Clustering

Next we cluster the ranked list of movies using the *k-means* algorithm.

K-Means Clustering Algorithm –

1. Take k random values as cluster centers.
2. Let each item belong to the cluster whose cluster center it is closest to.
3. For each of the k clusters, find a new cluster center by taking the mean of its items.
4. Repeat steps 2 and 3 until the cluster centers are stable.

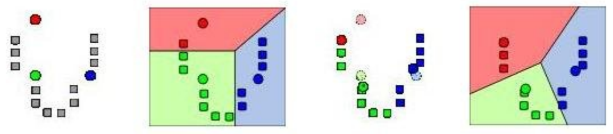

Figure 3: The 4 steps of the K-Means algorithm

Distance Measure –

Each movie is a vector of attributes; Name, Genre, Starring, Director, Year, Age Group, Gender, Rank. In order to cluster the movies by using k-means we need a *distance measure* between two movies. We start with a distance of 0. For each attribute two movies have in common we subtract the number of points that it gives (see Table 3) from the distance, a lower distance value means the movies are closer together.

Table 3: Shows the number of points that can be subtracted from a distance measure between two movies, for each attribute.

Attributes	Points	Max number of Points
Genres	Number of genres in common	Lower number of genres
Starring	Number of actors in common	Lower number of actors
Directors	Number of directors in common	Lower number of directors
Year	(5 - the number of decades between the movies) / 2	2.5
Age Category	1 if they belong to the same category, else 0	1
Gender	1 if they are suited for the same gender group, else 0	1
Rank	(highest rank – the difference in ranks) / (highest rank / 4)	4

Application –
Applying the k-means, we start by taking 5 movies, one in every n/5th place (n being the number of movies) from the ranked list of movies, as cluster centers. We do it this way instead of choosing them randomly because it saves us a few iterations since their rank somewhat categorizes them. Using the distance measure described above we are able to calculate a movie's distance from each of these centers and see to which cluster it belongs. For each cluster the new cluster center becomes the movie with the median distance from the last center. This is repeated until we get close to fixed cluster centers, but since this is a discrete space convergence is not guaranteed so we put an upper limit to the number of iterations.

Generating Questions

Generate question –
Next we want to generate a question from one of the clusters. We choose the cluster with the highest mean rank. From its cluster center, we generate a queue of questions, one for each attribute. The first question is the one whose attribute has the highest occurrence in the cluster, while the last question is the one whose attribute has the lowest occurrence. We ask the user the first question. If a question about that attribute has already been asked, we choose the next question in line. To make the conversation between the user and the computer more interesting, we have a number of possible ways to ask a question about each genre, which we choose randomly from when generating a question.

Change rank based on answer –
User's answers must include: "yes", "no" or "don't know" for the computer to understand the answer. If the user answers "yes" the movies that have the attribute being asked about increase in rank, if the answer is no the rank is decreased and if it is "don't know" nothing changes. Only the movies within the cluster about which the question was asked are affected. For example, if the attribute with the highest occurrence in the top ranked cluster is Steven Spielberg, the user is asked a question like:

"Do you like Steven Spielberg's movies?"

If the user answers "yes", all movies in that cluster which have Steven Spielberg as director will have their rank increased, if the answer is "no" their rank will decrease.

Recluster and repeat –
After changing the rank, we recluster the movies and generate a new question. Since the rank is used in calculating the distance metric, this will give us different clusters, eventually having all the highest ranked movies in the same cluster. This repeated five times, so the user gets asked five questions based on what the computer knows so far at each point. Finally it shows the user the movies with the highest rank.

Feedback to the Machine learning model

Once the user receives the recommended movies, we request the user to give a feedback about the recommendations. We use this feedback to generate training data samples for the machine learning model. If the user agrees to give feedback and says he likes the recommendations, then we extract the three most common genres and year from the recommended movies and use this data along with the user's personal information to generate a training vector that is fed back into the SVM model. If the user feels that the recommendations are not good, then we ask the user to give us inputs regarding his preferred genres and year so that we can train our SVM model to perform better.

Experiment

Correlation Analysis Using SVM Regression

Training Data Collection

In order to correlate movie preference with personal particulars for users, we gathered training data from randomly chosen demographic groups. People constitute a central part in our data collection process. We sent out data collection requests to different groups of people through e-mails, flyers, phone calls and other various communication ways, and we eventually succeeded in collecting data from people of different age brackets, occupations, cultural backgrounds etc. The current training data at hand is limited by its size, however, our system is designed in a scalable way that more data can be incorporated in the future.

Experiment Design

The SVM tool used in this experiment is the open source LIBSVM library written in Java, developed by Chih-Chung Chang and Chih-Jen Lin at Department of Computer Science, National Taiwan University, which has lately become a prevalent tool in SVM research community (Schauland et al., 2006; Thanh-Nghi and Fekete, 2007). The following table shows some parameters that need to be configured for the LIBSVM library.

Table 4: Parameters in LIBSVM library

-s	The type of the SVM In our experiment, we use ε–SVR (epsilon-SVR) as the SVM regression model.
-t	The type of the kernel function We choose RBF function as our core function.
-p	The epsilon in loss function of epsilon-SVR We choose different *–p* values in our experiment.

-c	The cost, the penalty parameter, in epsilon-SVR We choose different *–c* values in our experiment.
-g	Gamma in RBF Kernel function We use 1/5 as the value for gamma in our experiment, where 5 is the number of input attributes for the samples.
-epsilon	Tolerance of termination criterion We use 1e-8 in our experiment as the value for the tolerance.

An intermediate evaluation process was designed to evaluate the prediction result done by SVM. It compared the predicted preference vector with the actual preference vector provided by the user to obtain an error rate. In our experiment, different parameter spaces were traversed to enable an optimal combination of parameter settings. Based on our heuristic evaluation process, an order of 100 serves to be the best range for *–c*, 0.1 serves to be the best value for *–p* and 1e-4 serves to be the best choice for *–epsilon*.

Cluster Analysis Using K-means

Distance calculation

The basis for the clustering is the input that we get from the SVM. To refine the weights for the various attributes of the movies that are used in distance calculation, we initially tried with all the attributes having equal weights. However, we wanted to give more value to the input from the SVM, for clustering. Hence, we decided to give more weight to the rank attribute, which represents the rank of the movie with respect to the results from the SVM. Initially while calculating the distance between two movies, we were only checking if two attributes were equal or not. However, for the rank and year attribute, the difference in the values signifies how similar the movies are. So instead of checking for equality, we used the difference in the rank and year attribute values in the distance calculation.

Example –
In Table 5 we have shown samples of the outputs we get from the two distance calculation methods described above with the SVM input as ‘Animation Family Comedy 2000s’. The left one is the sample with the initial equal weight distance calculation. We see that as we do not take into account the similarities based on close ranks or year, we get some movies that don’t match well with the user’s preferences. On the other hand, the column on the right shows a sample with the same SVM inputs, but using a distance calculation that uses the difference in ranks and year. This gives us better results as movies that have close ranks have lower distance between them.

Table 5: The first column shows a sample run with equal weight distance calculation, and the second column shows a sample with rank based distance calculation.

I am guessing you are into Family. Am I right? yes Are you in the mood for Animation movies? yes I am guessing you are into Comedy. Am I right? yes Do you want to watch a kids movie? yes Do you like Adventure movies? no I think, you dont want to watch a kids movie! Is that correct? no Recommendation: Happy Feet Beauty and the Beast The Lion King Madagascar Ice Age: The Meltdown	Are you in the mood for Family movies? yes Do you want to watch a kids movie? Yes Do you like Animation movies? yes Are you in the mood for Adventure movies? no Do you like Comedy movies? yes Do you want to watch a recent movie? yes Recommendation: Madagascar Ice Age: The Meltdown Happy Feet Shrek 3 Finding Nemo

Cluster to ask questions about

When we came up with the clusters of movies, the main question was which cluster to choose for asking the questions. The cluster that we choose should be the one with maximum similarity to the user's choice. Initially we chose the cluster with the maximum number of movies so that the user's answer would affect the maximum possible number of movies. However this cluster does not really represent the movies that have the maximum similarity to the user's choice. The attribute that represents the user's choice is the rank of the movie. So the cluster with the movies having highest rank is the one with maximum similarity to the user's choice. So we decided to ask questions about the cluster with the highest mean rank.

Table 6 shows how the two approaches trigger different questions. In the left the questions are asked based on the cluster with the largest size, given that the three favorite genres are animation, family and comedy, suggesting that we do not want to watch a kids movie does not make sense. Finally, knowing that we do want to watch a kids movie, suggesting a romantic movie is also not appropriate. On the right side the systems asks questions based on the cluster with the maximum mean rank. All the questions are related to the desired genres.

Table 6: The first column shows a sample run where questions are asked based on the biggest cluster, the second column shows a sample where the cluster with the highest mean rank is used.

`Are you in the mood for Family movies?`	`Are you in the mood for Family movies?`
`Yes`	`yes`
`I think, you don't want to watch a kids movie! Is that correct?`	`Do you want to watch a recent movie?`
`Uuuhh no`	`yes`
`I am guessing you are into Comedy. Am I right?`	`Are you in the mood for Comedy movies?`
`Yes`	`yes`
`Are you in the mood for Animation movies?`	`I am guessing you are into Animation. Am I right?`
`Yes`	`yes`
`Do you want to watch a recent movie?`	`Are you in the mood for Adventure movies?`
`Yes`	`no`
`Do you like Romance movies?`	`I am guessing you are into Fantasy. Am I right?`
`No!`	`Yes`

Another parameter to consider about the clustering process is how many clusters should be used. The k-means algorithm relies on the cluster number as input so it is up to the implementer to decide the number. At first we tried specifying a cluster size, so the number of clusters grew proportionally to the number of movies in the data set. Our hypothesis was that this way movies clustered together would be highly similar so that asking a question based on that cluster would be informative for all its movies. However, when the number of clusters grows big, similar movies fall into different categories and getting feedback on such a small number of movies at once doesn't have much effect on the future questions as most of the clusters will still be the same. We decided to fix the number of clusters, since there are only so many ways movies can be grouped. This way, having a large database, we are also able to affect more movies at each question, and therefore fewer questions need to be asked.

Movie Data Collection

Any expert system relies heavily on an extensive dataset. In order to get reliable results it is important that we have a good dataset. Looking through some freely available movie datasets, we came across a few datasets that provided some of the information that we were looking for, but we did not find any that had all the information. This is because most of the movie recommendation systems rely on user given movie rating rather than the features of the movie itself. Our system on the other hand is mainly based on the features of the movie. Hence, it was important for us to have a dataset that will have all the information about the movie features. For use in this system we built a movie dataset that had the information about the features of the movies that were being used in out system. We created an XML file to store the movies. A sample movie node in the file is shown in Figure 4.

```
<MOVIE>
    <NAME>Titanic</NAME>
    <GENRE>Romance</GENRE>
    <GENRE>Drama</GENRE>
    <STARRING>Leonardo DiCaprio</STARRING>
    <STARRING>Kate Winslet</STARRING>
    <DIRECTOR>James Cameron</DIRECTOR>
    <YEAR>1990s</YEAR>
    <AGE_GROUP>Adults</AGE_GROUP>
    <GENDER>Both</GENDER>
    <RATING>7.2</RATING>
    <OSCAR>11</OSCAR>
  </MOVIE>
```

Figure 4: Sample movie node from the Movies XML

However, due to limited resources, the size of the dataset that we could create is not as extensive as would be ideal. In order to evaluate the scalability of our system for a large dataset, we used one of the freely available movie datasets. This dataset has 1000 movies but it only has information about few of the features that we are using i.e. genres of the movie and the year. The results that we obtained from this dataset were reasonable given that we had fewer features for each movie. For example, starting with a SVM input of ‘Adventure Action Comedy 1990s’, the system came up with reasonable questions asking about action movies, 90s movies, drama, thrillers, etc. and recommended movies that matched with the user’s preferences.

Conclusion

In this paper we have introduced MovieGEN, an expert system for movie recommendation. SVM-based machine learning on training data and K-means cluster analysis on result sets of database inquiry constitute the key models of our system. The system takes in the users’ personal information and predicts their movie preferences. Afterwards it clusters the movies and generates questions to refine the recommendation. Finally it suggests movies for the users.
By the nature of the system, it is not a straightforward task to evaluate the performance since there is no right or wrong recommendation; it is just a matter of opinion. Based on informal evaluations that we carried out we got a positive response from the users.
A larger data set will enable more meaningful results using our system. Additionally we would like to incorporate different machine learning and clustering algorithms and study the comparative results. Eventually we would like to implement a web based user interface that has a user database, and has the learning model tailored to each user.

References

Abramson A., Cohen I., *Asymptotic Stationarity of Markov-Switching Time-Frequency Garch Processes*, 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, May 2006, 3: 14-19.

Adomavicius G., Tuzhilin A., *Extending Recommender Systems: A Multidimensional Approach* (2001)

Cai C.L., Shi Z.Z., *A modular neural network architecture with approximation capability and its applications*, Proceedings of the 2nd IEEE International Conference on Cognitive Informatics, 2003, pp. 60.

Chen T.S., Tsai T.H., Chen Y.T., Lin C.C., Chen R.C., Li S.Y., Chen H.Y., *A combined K-means and hierarchical clustering method for improving the clustering efficiency of microarray*, Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems, December 13-16 2005, pp. 405-408.

Estlick, M. et al., *Algorithmic transformations in the implementation of K-means clustering on reconfigurable hardware*. FPGA 2001, pp. 103-110.

Filho A.G.S. et al., *Hyperspectral Images Clustering on Reconfigurable Hardware Using the K-Means Algorithm*, Proceedings of the 16th symposium on Integrated Circuits and Systems Design 2003, pp. 99-104.

Gencay R., Qi M., *Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging*, IEEE Transactions on Neural Networks, July 2001, 12(4): 726-734.

Guralnik, V., Karypis, G., *A scalable algorithm for clustering protein sequences*. in Proc. Workshop Data Mining in Bioinformatics (BIOKDD), 2001, pp. 73-80.

Huang G.B., Chen L., Siew C.K., *Universal Approximation Using Incremental Constructive Feedforward Networks*, IEEE Transactions on Neural Networks, July 2006, 17(4): 879-892.

Joachims T., *Text categorization with SVM: learning with many relevant features*. Proceedings of ECM, 10th European Conference on Machine Learning, 1998.

Jung S., Harris K., Webster J., Herlocker, J.L. *SERF – Integrating Human Recommendations with search* (2004)

Lawrence R.D., Almasi G.S., Kotlyar V., Viveros M.S., Duri S., *Personalization of supermarket product recommendations* (2001)

Miller B.N., Albert I., Lam S.K., Konstan J.A., Riedl J., *MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System*

Saegusa T., Maruyama T. *Real-Time Segmentation of Color Images based on the K-means Clustering on FPGA*, 2007 International Conference on Field-Programmable Technology, December 12-14 2007, pp. 329-332.

Schauland S., Kummert A., Su-Birm P., Uri I., Zhang Y., *Vision-Based Pedestrian Detection--Improvement and Verification of Feature Extraction Methods and SVM-Based Classification*, 2006 IEEE Intelligent Transportation Systems Conference, pp. 97-102.

Steinbach, M., Karypis, G., Kumar, V., *A Comparison of Document Clustering Techniques*. KDD Workshop on Text Mining, 2000.

Sun, Z.H., Sun, Y.X., *Fuzzy support vector machine for regression estimation*. IEEE International Conference on Systems, Man and Cybernetics, Oct 5-8 2003,Vol. 4, pp. 3336-3341.

Thanh-Nghi D., Fekete J.D., *Large Scale Classification with Support Vector Machine Algorithms*, ICMLA 2007, Sixth International Conference on Machine Learning and Applications, Dec. 13-15 2007, pp. 7-12.

Tong, S., Chang, E., *Support vector machine active learning for image retrieval*. Proceedings of ACM International Conference on Multimedia, pp. 107-118.

Trafalis, T.B., Ince, H., *Support vector machine for regression and applications to financial forecasting*. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, July 24-27 2000, Vol. 6, pp. 348-353.

Tsai C.F., Chen Z.C., Tsai C.W., *MSGKA: an efficient clustering algorithm for large databases*, 2002 IEEE International Conference on Systems, Man and Cybernetics, October 6-9 2002, 5: 6.

Turney P.D., Littman M.L., "*Measuring praise and criticism: inference of semantic orientation from association*, ACM Transactions on Information Systems, 2003, 21: 315-346.

Vapnik, V., *Statistical learning theory*, New York: Wiley, 1998.

Weng S.S., Lin B.S., Chen W.T., *Using contextual information and multidimensional approach for recommendation* (2007)

Wu, C.H., Ho, J.M., Lee, D.T., *Travel-time prediction with support vector regression*, IEEE Transactions on Intelligent Transportation System, December 2004, 5(4): 276-281.

Xu M.T., Franti P., *A heuristic K-means clustering algorithm by kernel PCA*, 2004 International Conference on Image Processing, October 24-27 2004, 5: 3503-3506.

Zhang X.H., Lu Z.B., Kang C.Y., *Underwater acoustic targets classification using support vector machine*, Proceedings of the International Conference on Neural Networks and Signal Processing, December 14-17 2003, 2: 932-935.

Zhao, Y.G., He, Q.M., *An unbalanced dataset classification approach based on v-support vector machine*, The Sixth World Congress on Intelligent Control and Automation, June 21-23 2006, 2: 10496- 10501.

Zheng Z.J., Webb G.I., Ting K.M., *Integrating boosting and stochastic attribute selection committees for further improving the performance of decision tree learning*, Proceedings of the Tenth IEEE International Conference on Tools with Artificial Intelligence, 1998. Nov 10-12 1998, pp. 216-223.

Zhong W., Altun G., Harrison R., Tai P.C., Pan Y., *Improved K-means clustering algorithm for exploring local protein sequence motifs representing common structural property*, IEEE Transactions on NanoBioscience, September 2005, 4(3): 255-265.