VRules: An Effective Association-based Classifier for Videos

Ling Chen, Sourav S. Bhowmick and Liang-Tien Chia
Center for Multimedia and Network Technology
School of Computer Engineering
Nanyang Technological University, Singapore 639798
{pg02322722, assourav, asltchia}@ntu.edu.sg

ABSTRACT

Video classification is an important step towards multimedia understanding. Most state-of-art approaches which apply HMM to capture the temporal information of videos have the limitation by assuming that the current state of a video depends only on the immediate previous state. Nevertheless, this assumption may not hold for videos of various categories. In this paper, we present an effective video classifier which employs the association rule mining technique to discover the actual dependence relationship between video states. The discriminatory state transition patterns mined from different video categories are then used to perform classification. Besides capturing the association between states in the time space, we also capture the association between low-level features in spatial dimension to further distinguish the semantics of videos. Experimental results show that the performance of our association rule based classifier is quite promising.

1. INTRODUCTION

With the ever-growing digital libraries and video databases, it is increasingly important to understand multimedia data automatically. Video classification is the first step toward multimedia content understanding [3]. Typical applications of video classification are efficient video database indexing and retrieval. Different methods have been proposed in the literature for classifying videos into predefined categories. Existing works on video classification can be roughly divided into two groups, one considers only the spatial information of videos [10] and the other takes both the spatial and temporal information of videos into account [2][5]. The advantage of the former method is that they work simply and generate classifiers that can be understood directly, whereas the latter succeeds in capturing the temporal dynamic characteristics of videos, which is a critical cue in understanding video content. Due to its capability in grasping the temporal statistical properties of stochastic processes, the Hidden Markov Model (HMM) was frequently applied by the approaches of the second group. Without complicating the model, most of the approaches used the first-order HMM theory, which assumes that the current state of a video depends only on the previous state. However, this assumption may not hold for videos of various categories, which results in certain limitations. For example, consider the video of a basketball game. From a state of the middle court, it might move into a state of the right court or a close-up of a player. The previous state before the middle court may affect which state it enters into. For instance, if the previous state of the middle court state is a left break up, it probably enters into a state of the right court. Whereas, if the previous state is a left dunk, it possibly enters into a close up. In this case, being the current state depends on the previous two states. Unfortunately, it is unknown beforehand which n-order HMM is appropriate for videos of different categories. The dependency relationship between the sequential states of videos can only be decided correctly by mining the video datasets. Hence, in this paper, we consider building an association rule based classifier which uses mined associations between video states to perform classification task.

Since a single feature may not provide sufficient information, multiple features are usually considered for accurate classification. Basically, existing works on video classification utilized multiple features in two manners. One is to concatenate low-level features directly to form a long feature vector [7]. The other is to create classifiers for each feature and then combine the classifiers to make a final decision [6]. The former is straightforward but has the "curse of dimensionality" problem when many features are involved. The latter will suffer from inappropriate combination strategies. In this paper, we consider multiple features in a way different from both of the above approaches. As observed in [1], the association between low-level features can help discriminate the image contents very well, which is also applicable to videos. Hence, we take advantage of multiple features by capturing the association between them in sequential video states to further distinguish the semantics of videos.

As an overview, our association rule based classifier, which is referred as *VRules*, is built as follows. In the first place, we preprocess the set of training videos. Each training video sequence is partitioned into shots which are modelled as states. Two low-level features, color and motion, are then extracted from each state. We cluster the color and motion features separately and assigned symbols to states according to their feature values. Thus, each training video is transformed into

a sequence of state symbols. Next, we mine frequent *state transition pattern*s from the transformed training videos, namely, the set of state symbol sequences. Association rules relating the discovered state transition patterns with video class labels are generated subsequently. For example, an association rule with the form $<s3, s5> \Rightarrow C2$ indicates that a video very likely belongs to class *C2* if it frequently moves into the state *s5* from *s3*. Note that the association between low-level features are considered implicitly as each state represents a pair of color and motion values. Finally, we test the accuracy of the constructed classifier by predicting the class labels for unknown videos based on *VRules*.

Main contributions of the paper are summarized as follows. First, an association rule based classification model is proposed, which captures the temporal information of videos as the HMM-based approaches do, but avoids making any Markov assumption. Second, not only the association between sequential video states but also the association between low-level features are utilized together to improve the accuracy of the classifier. Furthermore, wild card operators are introduced into the set of symbols to tolerate some uncertain video states, which improve the robustness of the classifier (the details will be discussed in Section 2).

The rest of the paper is organized as follows. In Section 2, we discuss the preprocessing of training videos. Section 3 presents the association rules and its mining techniques. We explain the tasks in the training and testing phases of the classification model respectively in Section 4. Experimental results of classifying videos into four predefined categories, basketball, football, news and sitcom, are given in Section 5. We discuss related research work in Section 6 and draw some conclusions in Section 7.

2. VIDEO PREPROCESSING

In this section, we discuss preprocessing the set of training videos to transform them into the state symbol sequences. Basically, the following three steps are involved. First, we model each video process in the time space consisting of a sequence of states. Second, the color and motion features are extracted from the videos based on states. In order to reduce the combination possibilities of different feature values, similar features are then clustered. Third, the video states are assigned symbols according to the clusters that their features belong to.

2.1 State Modelling

Most HMM-based video classification approaches do not need to find out the states explicitly as the states are hidden. On the contrary, the states should be defined specifically in our approach before any frequent patterns can be mined. When deciding how to appropriately model a video as a sequence of states, we are motivated by the following consideration. Different types of videos have different patterns and certain patterns occur repeatedly. For example, the newsreader appear in regular interval in news videos. In soccer videos, the penalty area, the middle field and the close up appear again and again. Therefore, with the purpose of capturing these patterns in videos, it is advisable to model the shots or scenes of a video as states. Shot is the physical boundary of the video and it is relatively easier to be detected than scene, which is the semantic boundary of the video. Thus, we model the shots of videos as states (our experimental results show that the shots suffice to find interesting patterns).

We modified a published algorithm [8] which used the motion prediction information in MPEG videos to detect the shot boundaries. Basically, a Frame Dissimilarity Ratio (FDR) is defined to compute the dissimilarity between the I-, P- and B-frames based on the number of macroblocks (MB) of different types. In [8], the definition of FDR was not distinguished for I- and P-frames and both are defined as the ratio of the number of forward predicted MBs to the number of bi-directionally predicted MBs in its previous B-frame. Actually, if the shot occurs in a P frame, it can be identified as its majority of MBs will be intracoded. In addition, [8] defined the FDR for B-frames as the maximum value of $\frac{Fw_n}{Bi_n}$ and $\frac{Bk_n}{Bi_n}$, where Fw_n, Bk_n and Bi_n represent the number of forward predicted, backward predicted and bi-directionally predicted MBs respectively in the n-th frame. Thus, it would erroneously identify a B-frame as a shot boundary if both the values of Fw_n and the Bk_n are large while the value of Bi_n is small. Therefore, we modify the definition of FDR as follows to improve the accuracy of shot boundary detection.

$$FDR_n = \begin{cases} \frac{Fw_{n-1}}{Bi_{n-1}} & \text{for I-frame} \\ \frac{In_n}{Fw_n} & \text{for P-frame} \\ \frac{|Fw_n - Bk_n|}{Bi_n} & \text{for B-frame} \end{cases}$$

Where the notation of In_n represents the number of intracoded MBs in the n-th frame. Consequently, frames with their FDRs greater than some threshold will be detected as the shot boundaries (interested readers can refer to [8] for the selection of threshold). After segmenting each video into a sequence of shots, we model each shot as a state.

2.2 Feature Extraction and Clustering

Now we extract features from videos based on states. We consider two important low-level features for videos: color and motion. A color feature similar to the one used in [5], which is insensitive to lighting change, is applied. Specifically, we extract the color feature from the intracoded I-frames in each state. For each I-frame, the DC image is formed by the DC coefficients of all the 8×8 blocks over the I-frame. An average DC image is computed for all I-frames in the state and then it is converted into the RGB space. Lighting is discounted by normalization with the equation, {r, g}={R, G}/(R+G+B). We compress the color histogram by quantizing each r and g into 16 levels. A 16×16 DCT is carried out subsequently to improve the energy compaction. After applying zigzag scan, we get the first 21 coefficient as the feature vector. Motion feature is extracted from the P-frames in every state. First, we calculate the mean magnitudes of the motion vectors for each P-frame. Then we choose the median value of all mean magnitudes of P-frames in the state.

After feature extraction, each state has a pair of color and motion feature values. Since few states share the exactly same feature values, we should group similar features into clusters first. In order to capture the association between low-level features, we cluster the color and motion separately rather than concatenating them and clustering together. We used an agglomerative hierarchical clustering approach as it is good at controlling the distortion of clusters. By distortion, we mean the maximal Euclidean distance between any two points in a cluster. Initially, each feature value is taken

as a cluster. Clusters that are close in distance are then merged step-by-step according to the criteria that the distortion of clusters are minimized. The merging process stops if the number of specified clusters are reached. Basically, the number of clusters should be far less than the total number of states in the set of training videos. Note that, the clustering process is simplified because the entropy of clusters has not been addressed here. By entropy, we mean the cardinality of the clusters. Actually, we consider the entropy of clusters when assigning symbols to states in the next stage. We will revisit this point in the next subsection.

2.3 Symbol Assignment

After clustering, each state of a video can be denoted as $S = (C_p, M_q)$, which means that the color of the state belongs to a color group C_p and the motion of the state belongs to a motion group M_q. For example, the upper-left matrix in the Figure 1 shows the combination of color and motion groups. Each video state corresponds to a cell in the matrix. Recall that the entropy of clusters has not been addressed in the clustering process, some color/motion groups may contain very few states so they are too trivial to be interesting. In order to further reduce the combination possibilities of color and motion groups, we only assign state symbols to interesting combinations (cells). In particular, we consider a cell (C_p, M_q) interesting if it satisfies the following conditions. Let $|T|$ be the total number of states in the training videos, $|T_1|$ and $|T_2|$ be the number of clusters for color and motion features respectively. Let *count(X)* be a function which computes the number of states belonging to the group X,

1. $count(C_p) \geq |T|/T_1$;
2. $count(M_q) \geq |T|/T_2$;
3. $count((C_p, M_q)) \geq |T|/(T_1 \times T_2)$.

That is, a cell is interesting if the number of states in the corresponding row (column) is greater than the average number of states in each row (column) and the number of states in the cell is greater than the average number of states in each cell.

Then, states corresponding to interesting cells are assigned state symbols. States corresponding to the same cell are assigned the same symbols. For example, the two state S_i and S_{i+1} in Figure 1 are assigned the same symbol because they have the same pair of color and motion groups. For states corresponding to any uninteresting cell, we do not distinguish them and assign a wildcard operator "?" to represent them all. For example, the state S_{i+2} in Figure 1 is assigned the symbol "?" because either C_1(the first column) or M_1(the first row) contains less than the average number of states, or the pair (C_1, M_1)(the left-most cell in the first row) contains less than the average number of states.

The wildcard operator "?" indicates that the state can be any unspecific state. By introducing "?" as a unique symbol, we are allowed to discover state transition patterns between not only specific states, i.e.<*s18*, *s2*>, but also specific and unspecific states, i.e.<*s18*, ?>. The latter is useful for videos because the content of video varies extensively. For example, consider the news videos. The state containing the newsreader is probably assigned a specific symbol because it occurs frequently, while the states between two occurrences of the newsreader may be quite uncertain so that they are probably assigned the symbol "?". Thus, we have the chance to discover a frequent state transition pattern between specific and unspecific states for news videos.

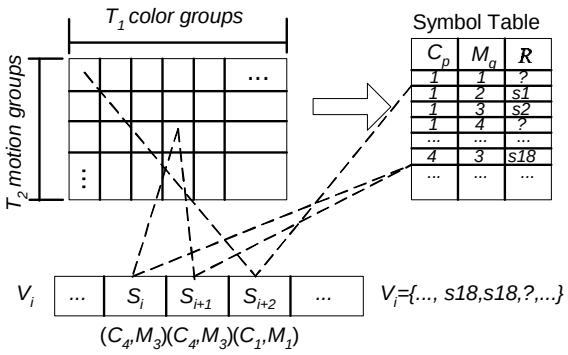

Figure 1: Symbol Assignment

In addition, for continuous specific states assigned the same symbol, we are not interested in the specific times the state occurs continuously. Rarely do two videos share a common state transition sequence with a symbol occurring continuously for exactly the same times, even if the videos belong to a same category. Hence, we replace the continuous occurrence of a symbol with a symbol appended with a "$\star$", which implies that the symbol occurs multiple times. The symbol "$\star$" can help to find, for a particular video type, whether a state usually transit to itself. However, for continuous unspecific states in a video, we use only one "?" in stead of "?,$\star$" because the transition between uncertain states does not make any sense. That is, a "$\star$" only follows a specific state.

Let $\Re$ be the complete set of symbols, containing k state symbols and two unique symbols "?" and "$\star$", $\Re = \{s1, s2, ... sk, ?, \star\}$ (The number k depends on the number of interesting pairs of color and motion groups in training videos). Each video V in the training database D, $V \in D$, can be transformed into a state symbol sequence $V = <v_1, v_2, ..., v_n>$, $v_i \in \Re$ $(1 \leq i \leq n)$.

3. RULE DISCOVERY

In this section, we first define the *state transition patterns* and the association rules relating the state transition patterns to video class labels. Then, we present the rule discovery process.

3.1 Patterns and Rules

Given a training video database D, let $\Re$ be the complete set of symbols, let $C = \{C_1, ..., C_l\}$ be the l class labels for videos in D. For each video V, let *C(V)* be the class label of video V. Then, $\forall\ V \in D$, $V = <v_1, v_2, ..., v_n>$, $v_i \in \Re$ $(1 \leq i \leq n)$ and $C(V) \in C$. For example, Table 1 shows an example database containing 5 video state symbol sequences from 3 classes. $\Re = \{s1, s2, s3, s4, ?, \star\}$.

Consider the database of transformed video state symbol sequences, we are interested in the *state transition patterns* contained in the sequences, which are defined as follows.

DEFINITION *1.* (State Transition Pattern) Given a set of state symbols $\Re = \{s1, s2, ... sk, ?, \star\}$, $P = <p_1, p_2, ..., p_m>$ is a *state transition pattern* if 1) $\forall\ i \in [1,m]$, $p_i \in \Re$, 2) $\forall\ i,j \in [1,m]$, if $p_i \neq \star$ and $p_j \neq \star$, then $p_i \neq p_j$.

From the definition of *state transition pattern*, we can observe that,

ID	Symbol Sequence	Class Label
V_1	< *s3*, *s2*, ?, *s4* >	C_1
V_2	< ?, *s3*, *s1*, *s3*, *s2* >	C_3
V_3	< *s3*, *s1*, ⋆ >	C_2
V_4	< ?, *s3*, *s2* >	C_1
V_5	< *s3*, ⋆, *s1*, ⋆, *s3*, *s1* >	C_2

Table 1: An Example Database of Video State Symbol Sequences

- It captures the transitions between more than two states. For example, consider the first video state symbol sequence in Table 1. If the first-order HMM is applied, it takes into account the following transitions: *s3*→*s2*, *s2*→ ?, ?→ *s4*. However, it fails to capture the transitions such as *s3*→*s2*→?→*s4*.
- It captures the transitions between distinct states and the transitions between a state and itself. Since we are interested in the frequent transitions between states rather than the common sequences shared by videos, we require that a state transition pattern contain no duplicate states except when they occur continuously, which means the state frequently transits to itself. For example, consider the last video in Table 1. It contains 5 state transition patterns: <*s3*, ⋆, *s1*, ⋆>, <*s3*, *s1*, ⋆>, <*s1*, ⋆, *s3*>, <*s1*, *s3*> and <*s3*, *s1*>.

Obviously, given a video state symbol sequence V, the number of state transition patterns it contains is ($|V|$-1). For example, consider the second video in Table 1. The length of the sequence is 5, so it contains 4 state transition patterns, <?, *s3*, *s1*>, <*s3*, *s1*>, <*s1*, *s3*, *s2*>, <*s3*, *s2*>. Before defining the *support* of a state transition pattern in a video sequence, we define the *support relationship* between two state transition patterns as follows. Given two state transition patterns $P = <p_1, p_2, ..., p_m>$ and $Q = <q_1, q_2, ..., q_n>$, we say P is *supported* by Q , denoted as $P \preceq Q$, if $m \leq n$, and $p_1 = q_1, ..., p_m = q_m$. In other words, P is supported by Q only if P is a prefix of Q.

Then, given a video sequence V, let $S(V)$ be the set of state transition patterns contained in V ($|S(V)|=|V|$-1), we define the *absolute support* of a state transition pattern P in V, denoted as $\sigma^A(P, V)$, as the number of state transition patterns contained in V supporting P. That is, $\sigma^A(P, V) = |\{Q \in S(V) \mid P \preceq Q\}|$. For example, suppose P=<*s3*, *s1*>, the absolute support of P in the second video in the Table 1, $\sigma^A(P, V_2)$ is 1 because there is only one state transition patterns in V_2 supporting P.

Definition *2.* (Support of State Transition Pattern) Given a training database D which consists of $|D|$ videos, the *support* of a state transition pattern P in D, denoted as $\sigma(P, D)$, is defined as the fraction of state transition patterns contained in videos in D which support P.

$$\sigma(P, D) = \frac{\sum_{i=1}^{|D|} \sigma^A(P, V_i)}{\sum_{i=1}^{|D|}(|S(V_i)|)}$$

P is a *frequent state transition pattern* with respect to D if $\sigma(P, D) \geq \sigma^{min}$, where σ^{min} is a user-defined minimum support. For example, suppose the user-defined σ^{min} is 10%, the state transition pattern P = <*s3*, *s1*> is frequent as $\sigma(P, D)$=25%$\geq \sigma^{min}$ (the support is 25% because P is supported four times by the total 16 state transition patterns contained in the videos in D).

The goal of our association rule is to relate frequent state transition patterns to video class labels. Hence, the rules have the form as $P \Rightarrow C_l$, where P is a frequent state transition pattern and C_l is a video class label. Such an association rule implies that if an unknown video contains any state transition pattern which supports P, it probably belongs to class C_l.

Similar to the definition of the support of a state transition pattern, the support of an association rule can be defined as follows.

Definition *3.* (Support of Association Rule) The *support* of an association rule $P \Rightarrow C_l$ is the fraction of state transition patterns which are contained in videos belonging to class C_l in training database D and support P.

$$\sigma(P \Rightarrow C_l) = \frac{\sum_{C(V_i)=C_l} \sigma^A(P, V_i)}{\sum_{i=1}^{|D|}(|S(V_i)|)}$$

$\sigma(P \Rightarrow C_l)$ can take values between [0, 1]. The higher the support, the more statistically meaningful the rule. For example, consider the database in Table 1, σ(<*s3*, *s1*>⇒C_2)= 3/16 = 18.75% while σ(<*s3*, *s1*>⇒C_3)= 1/16 = 6.25%.

Besides *support*, another metric *confidence* is commonly used in association rule mining to measure the strength of an association rule. It can be defined here correspondingly as follows.

$$\rho(P \Rightarrow C_l) = \frac{\sum_{C(V_i)=C_l} \sigma^A(P, V_i)}{\sum_{V_i \in D} \sigma^A(P, V_i)}$$

That is, the confidence of the rule reflects the conditional probability that a video belongs to class C_l if its state transition patterns support P. However, in a database with unevenly distributed classes, the parameter is biased in favor of dominant class. Therefore, we use another metric *weighted confidence*, defined in [9], which weighs the absolute support of a pattern in a class with the class probability.

Definition *4.* (Weighted Confidence of Association Rule) Let D_l be the set of videos in D which belong to class C_l, let $\overline{D_l}$ be the remaining videos in D. The *weighted confidence* of the rule $P \Rightarrow C_l$, denoted as $\rho^w(P \Rightarrow C_l)$, is,

$$\rho^w(P \Rightarrow C_l) = \frac{\sigma(P, D_l)}{\sigma(P, D_l) + \sigma(P, \overline{D_l})}$$

Then the value of ρ^w lies in [0, 1] as well. The higher the ρ^w, the stronger the rule. For example, consider the rule <*s3*,*s1*>⇒C_3 with respect to the database in Table 1. ρ(<*s3*,*s1*> ⇒ C_3) = 1/4 = 25% while ρ^w(<*s3*,*s1*> ⇒ C_3) = $\frac{1/4}{1/4+3/12}$ = 50%. Obviously, the strength of a rule related with a minority class is improved by weighted confidence.

Therefore, given a training video database D, our goal is to learn a set of state transition rules. Each rule has the form $P \Rightarrow C_l$, [σ, ρ^w]. Both the support and strength of the rule should be no less than some user-defined thresholds, that is, $\sigma \geq \sigma^{min}$ and $\rho^w \geq \rho^{min}$.

3.2 Mining Process

Now, we discuss the process of association rule mining. We take the database of transformed video state symbol sequences as input, we aim to discover all rules with their support and weighted confidence no less than some user-defined thresholds.

3.2.1 Data Structure

In order to find all association rules efficiently, we construct a trie-like data structure, *unique prefix tree* (*UP-tree*), to register all state transition patterns contained in training videos. Simply, state transition patterns sharing a common prefix hang off a common node.

DEFINITION *5.* (Unique Prefix Tree) A unique prefix tree (*UP-tree*) is a tree structure T, which consists of a root node and a set of *unique prefix subtrees* as the children of the root. Each root of unique prefix subtree has a *symbol label* to indicate the state the node stands for. Each node in the unique prefix subtree consists of two fields: *symbol label* and *count array*. *Symbol label* registers which state this node represents. *Count array* is an array of counts which respectively records the number of state transition patterns contained in videos of different classes supporting the state transition pattern represented by the portion of the path reaching this node. Additionally, the root of the UP-tree also maintains a *count array* which records the total number of state transition patterns contained in videos of different categories.

Based on the definition, a UP-tree can be constructed intuitively by scanning the database of transformed video state symbol sequences once. First, we create the root of a UP-tree, T, and initialize its count array. For each video sequence V in D, we update the count array of the root node according to the length and the class label of the video. For example, consider the video V_5 in Table 1, <*s3*, ⋆, *s1*, ⋆, *s3*, *s1*>. Since its length is 6 and it belongs to the class C_2, we increment the second element of the root's count array by 5. Then we need to find the state transition patterns in each video. This can be done by finding the n-th state transition pattern from the n-th symbol in the video sequence. For example, consider the video V_5 again. From the first symbol, we find the state transition pattern <*s3*, ⋆, *s1*, ⋆> and from the second symbol, we find the state transition pattern <*s3*, *s1*, ⋆> and so on and so forth. For each found state transition pattern, we update the UP-tree with it. For example, consider the first state transition pattern contained in video V_5: <*s3*, ⋆, *s1*, ⋆>. We first search whether the root of the UP-tree has a child with its *symbol label* as *s3*. If yes, we go on handling the left symbols in the pattern from this node; otherwise, we need to create a new node labelled as *s3*. If the node is not a root of a unique prefix subtree, we further update its *count array* according to the class label of current video sequence. For example, since the video V_5 belongs to class C_2, the second element in the *count array* of the current node is incremented by 1. Finally, the *UP-tree* constructed for the training database in Table 1 is shown in Figure 2.

3.2.2 Rule Extraction

Obviously, the UP-tree registers the complete information of state transition patterns and the information for calculating the support and confidence of association rules. Hence, qualified association rules can be extracted from the constructed UP-tree without referring to the training database D any more.

All association rule can be mined from the UP-tree by employing a depth-first traversal. The rule mining process is described in Algorithm 1, we explain it with the constructed UP-tree in Figure 2. Suppose the user-defined threshold of

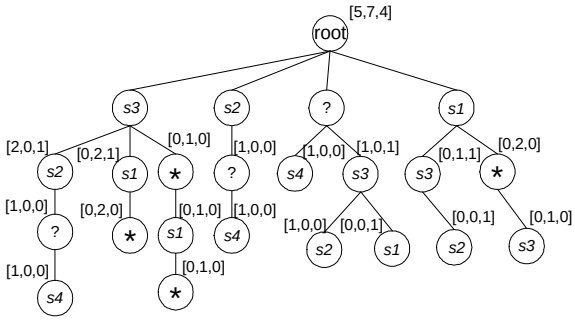

Figure 2: Unique Prefix Tree

σ^{min} and ρ^{min} are 10% and 75% respectively. Consider the left-most path of the UP-tree in Figure 2 first. When the depth-first traversal reaches the node labelled as *s2*, we get the state transition pattern <*s3*,*s2*>. Then, for each element of the node's count array, we compute the support and weighted confidence of the corresponding association rule. For example, the count array of *s2* is [2,0,1], we calculate the support and weighted confidence for rule <*s3*,*s2*>⇒C_1 by considering the first element in the array. Then, σ(<*s3*,*s2*>⇒C_1)=2/16=12.5% ≥ σ^{min}; ρ^{w}(<*s3*,*s2*>⇒C_1)= $\frac{2/5}{2/5+1/11}$=81.5%≥ ρ^{min} (the information of 16, 5 and 11 comes from the count array of the root node). Hence, <*s3*,*s2*>⇒C_1 is a qualified state transition rule which will be returned as a result. The support and weighted confidence of rules <*s3*,*s2*>⇒C_2 and <*s3*,*s2*> ⇒C_3 can be computed similarly. However, the two rules will be discarded as their support is not great enough. Since a qualified rule is found for node *s2*, we continue the depth-first traversal and obtain the state transition sequence <*s3*, *s2*, ?> (Line 9 of function DFSMine). No qualified rules will be discovered for the node ?, where the current depth-first traversal can be terminated. The complete set of association rules can be mined in the similar way by recursively traverse the remaining tree.

4. VRULES CLASSIFIER

In this section, we discuss the two phases for classification task, *training phase* and *testing phase*.

4.1 Training Phase

The training phase takes the training database of videos with known class variables as input. The goal is to learn a classification model, which is a set of association rules in our case. Basically, there are four steps involved in this phase.

- First, we need to preprocess the training videos such as modelling the videos in time space as state sequences, extracting features from videos based on states, clustering features and assigning symbols to states. The output of this step is the database of transformed video state symbol sequences.
- Given the database of transformed video state symbol sequences, this step aims to mine discriminatory association rules, which relate the frequent state transition patterns to video class labels. The output of this step is the set of association rules satisfying the user-defined threshold of support and weighted confidence.

Algorithm 1 RuleExtraction

Input:
A UP-tree T, user-defined minimum support σ^{min} and strength ρ^{min}

Output:
All state transition rules R

Description:
1: **for** all child node C of the root of the UP-tree T **do**
2: R=**DFSMine**(C, σ^{min}, ρ^{min}, R)
3: **end for**
4: return R
1: **function** DFSMine(N, σ^{min}, ρ^{min},R)
2: **for** each child node C of N **do**
3: Get the sequence P represented by this portion of the path reaching C
4: **for** each count C.count[l] **do**
5: **if** $\sigma(P, l) \geq \sigma^{min}$ && $\rho(P, l) \geq \rho^{min}$ **then**
6: add rule $P \Rightarrow$class l to R
7: **end if**
8: **end for**
9: **if** C is not a leaf node && a rule is found at C **then**
10: **DFSMine**(C, σ^{min}, ρ^{min},R)
11: **end if**
12: **end for**
13: **end function**

- Once the set of association rules are discovered, we need an ordering scheme to arrange the rules in descending order of their predictive power. To make the classifier concise, rules with weak discrimination will be pruned.
- Finally, a default class label should be decided to make the classifier complete. That is, a classifier should cover all possible testing videos. When no matching rule can be used to predicate a testing video, it will be classified into the default class.

Since the first two steps are discussed in details in Section 2 and 3, we focus on the last two steps in this section.

4.1.1 Ordering Rules

To ensure that only rules with high predictive power are chosen as the classifier, we impose a total order on the complete set of discovered association rules. We decide the precedence relationship between rules using a method similar to the one proposed in [4].

Given two association rules r^i and r^j, r^i precedes r^j, denoted as $r^i \prec r^j$, if

- The weighted confidence of r^i is greater than that of r^j, or
- if the two rules have the same weighted confidence, then the support of r^i is greater than that of r^j, or
- if the two rules have the same support, r^i has a shorter state transition pattern than r^j, or
- if the length of the state transition patterns of the two rules are same, the symbols in state transition pattern of r^i lexicographically precede the symbols in that of r^j.

After ordering the set of association rules, we prune those with weak predictive power or those that cannot improve the classification accuracy. For example, we can iteratively add a rule to the classifier, decide a default class for the remaining videos which cannot be covered by the classifier and count the error cases caused by current classifier and default class. The first rule that causes the least number of error cases is the cutoff rule. That is, rules after this cutoff rule will not be added into the classifier because these rules cannot improve the accuracy of the classifier. However, the process requires to scan the database multiple times to identify which cases are uncovered and which cases are wrongly classified. In order to make our learning process efficient, we simply discard rules which cannot distinguish between class C_l and $\overline{C_l}$. Thus, rules with their weighted confidence greater than 0.5 will be included in our classifier.

4.1.2 Deciding the Default Class

To make our classifier complete, we are required to decide on the default class. That is, if the state transition patterns contained in a testing video does not support any state transition pattern in the set of rules of the classifier, the testing video will be predicted to be the default class.

Usually, the default class can be defined differently according to different classification cost models. We assume that the classification for each class is equally weighted, then the default class can be defined as follows. Let R be set of rules in current classifier, where each $r^i \in R$ has the form $P^i \Rightarrow C_l^i$. Let Φ be the set of state transition patterns contained in training videos but not covered by rules in the classifier, $\Phi = \{Q \mid Q \in S(V) \wedge V \in D \wedge (\nexists r^i \in R \wedge P^i \preceq Q)\}$. Let Φ_l be the set of uncovered state transition patterns contained in videos belonging to class C_l, $\Phi_l = \{Q \in \Phi | Q \in S(V) \wedge C(V) = C_l\}$. We then select the majority class in Φ, that is, *default class* = $\text{argmax}_{c_l}\{|\Phi_l|\}$. Actually, to moderate the influence of videos containing more state transition patterns, we normalize Φ_l with the total number of state transition patterns contained in videos belonging to class C_l. Thus, we select the *default class* as follows,

$$default\ class = argmax_{c_l}\{\frac{|\Phi_l|}{\sum_{C(V_i)=C_l}|S(V_i)|}\}$$

At the end of the training phase, our learned classification models is the set *VRules* R: $<r^1, r^2, ..., r^n,$ *default class*$>$, where $r^i \prec r^{i+1}$ $(1 \leq i < n)$.

4.2 Testing phase

The testing phase takes the learned classifier *VRules* and testing videos as input, it aims to predict the class label for testing videos based on the classifier.

In order to predict the classes for testing videos, we first need to transform testing videos in symbol sequences as well. The clustering results for training videos should be used as the basis for the testing video transformation. That is, the color/motion feature of each state of the testing videos should be clustered to a group whose average value is the nearest one to the feature value of the state. Then, each state of the testing videos is assigned a symbol according to the same symbol assigning scheme for training videos. After that, each testing video can be represented as a sequence of state symbols.

For each testing video V_t which is transformed to be a sequence of state symbols, we retrieve the set of *VRules* $R(V_t)$

	Basketball	Football	News	Sitcom
Training set	4	4	4	4
Testing set	5	5	6	6

Table 2: Videos of the training and testing set

which can be used to predict its class label. That is, $R(V_t)$= $\{r^i \in R |\ P^i \preceq Q \wedge Q \in S(V_t)\}$, where r^i has the form $P^i \Rightarrow C_l^i$. Obviously, if $R(V_t)= \emptyset$, we directly classify the testing video as the *default class*. Otherwise, we compute the average weighted confidence respectively for rules predicting V_t to different classes. Let $R_l(V_t)$ be the set of rules predicting the testing video V_t as class C_l, $R_l(V_t) = \{r^i \in R(V_t) | C_l^i = C_l\}$. Let $\sigma^A(r^i, V_t)$ be the absolute support of the state transitions pattern P^i of rule r^i in the testing video V_t. Then the average weighted confidence for the rule set $R_l(V_t)$, denoted as $\overline{\rho}(R_l(V_t))$, can be computed as follows,

$$\overline{\rho}(R_l(V_t)) = \frac{\sum_{r^i \in R_l}(\rho^i \times \sigma^A(r^i, V_t))}{\sum_{r^i \in R_l} \sigma^A(r^i, V_t)}$$

where ρ^i is the weighted confidence of each r^i. As a result, the class label C_l will be selected as the label for the testing video V_t if the average weighted confidence for rules in $R_l(V_t)$ has the highest value.

5. PERFORMANCE STUDY

We evaluate our association rule based classifier *VRules* by classifying videos of four categories: basketball game, football game, news and sitcom.

5.1 Setup and Data sets

We implement the classification model with Matlab and Java. Experiments are performed on a Pentium IV 2.8GHz PC with 512 MB memory. The operating system is Windows 2000 professional.

The total data sets consists of 38 video clips, which are collected from MPEG-7 test set. Each video clip lasts one minute and belongs to a class of the four categories: basketball game, football game, news and sitcom. We randomly separate the set of videos into the training set and testing set. The size of the two sets is shown by Table 2.

5.2 Results and Analysis

We use two metrics, *precision* and *recall*, to evaluate the performance of our classifiers. In the context of video classification, the precision and recall can be calculated as follows.

Precision: the ratio of the testing videos correctly classified to a class over all testing videos classified to the class.

$$precision = \frac{|\{correct\} \cap \{classified\}|}{|\{classified\}|}$$

Recall: the ratio of the testing videos correctly classified to a class over all testing videos belonging to the class.

$$recall = \frac{|\{correct\} \cap \{classified\}|}{|\{correct\}|}$$

We conducted the experiments by varying the user-defined minimum support σ^{min} for *VRules* (as discussed in Section 4, the threshold for ρ^w is set as 50%).

Table 3 shows the classification results when the σ^{min} is set as 3%. The result is quite promising, the classifier can predict all testing videos belonging to football and sitcom correctly. It only misclassified one news video to basketball. The complete set of *VRules* contains 36 association rules whose state transition pattern contains two symbols, 32 rules with their state transition pattern containing three symbols, 10 rules have 4 symbols and 2 rules have 5 symbols in their state transition patterns. Some of the *VRules* are shown in Table 4 (class C_1, C_2, C_3 and C_4 correspond to basketball, football, news and sitcom respectively). From the set of *VRules*, we can observe that,

	Basketball	Football	News	Sitcom
Precision	83.3%	100%	100%	100%
Recall	100%	100%	83.3%	100%

Table 3: Precision and Recall (σ^{min}= 3%)

Antecedence	Consequence	σ	ρ
<?, *s1*, ⋆>	C_2	16.67%	100%
<*s4*, ?>	C_2	16.67%	100%
<*s8*, ?>	C_1	13.51%	100%
<*s7*, ?, *s8*>	C_1	8.1%	100%
<*s2*, ⋆, ?, *s3*, ⋆>	C_4	6.25%	100%
<*s6*, *s8*, *s4*>	C_3	3.22%	100%
<*s7*, *s4*, *s5*, *s6*, ⋆>	C_3	3.22%	100%

Table 4: Part of the VRules (σ^{min}= 3%)

- There are frequent state transition patterns between more than two states, which cannot be captured by first-order HMM based video classification approaches.
- Videos of different categories can share some common states, however, they can be distinguished by the different transition pattern between states. For example, from Table 4 we note that both C_1 and C_3 have the states *s7*, *s8*. But their transition patterns are different.

When the σ^{min} is set as 2%, the results are same. Table 5 shows the results when the σ^{min} is set to be 4%. The performance of classifying football game and news videos degrades because of the following reasons. There are matching rules which predicate these videos correctly. However, these rules are ordered lower. When the threshold for support is set higher, they are excluded from the classifier. Hence, some testing football/news videos cannot be correctly classified.

We compared our approach with an HMM-based video classification method [5] by conducting experiments with their approach on the same set of training and testing videos. Table 6 shows the results of their approach when the number of hidden states is selected as 2. We observe that our approach can work better than their approach when the number of hidden states is selected as 2 or 4. Our performance is comparable to theirs when the number of hidden states is set as 3. However, their approach needs some learning process before deciding the appropriate number of states for the set of training videos while we mine the knowledge directly from the videos.

6. RELATED WORK

We briefly review some related work in the section.

Video classification is a necessary tool for efficient access, understanding and retrieval of video. Different methods

	Basketball	Football	News	Sitcom
Precision	83.3%	100%	66.7%	100%
Recall	100%	60%	66.7%	100%

Table 5: Precision and Recall (σ^{min}= 4%)

	Basketball	Football	News	Sitcom
Precision	100%	83.3%	85.7%	100%
Recall	60%	100%	100%	100%

Table 6: Performance of HMM-based Classification

have been proposed in the literature for video program classification into predefined categories [5]. For example, an inductive decision tree is learned from low-level features in [10], and then a set of if-then rules are generated as classifiers. However, as observed by [5], video data contains not only spatial features, but also temporal information. The temporal dynamic characteristics should be taken into account in understanding video content. Therefore, recent work on video classification applied Hidden Markov Model (HMM) since it has the good capability to grasp temporal statistical properties of stochastic processes. For example, in [6], face and text trajectories are used to form the HMM and the learned model is used to classify videos from four categories of TV programs. Rather than considering visual features only, [2] constructs the HMM framework with both audio and image features. The combination of audio and visual feature really improves the classification accuracy. However, it suffers from the system complication and computational intractability. To address the efficiency of classification, [5] forms the HMM on summarized videos where only the chromaticity of key frames are considered.

Association rule mining have been proved to be useful in multimedia area by many proposals before. [1] argued that the semantic of images can be captured by mining hidden association between basic features. Then the authors proposed to use the associations between features to discriminate image repositories. Their observation on the association between features can be utilized in video classification as well. However, the association rules they discovered cannot indicate directly how probable an association is related to an image class. While our classifier address the problem by relating state transition patterns with video class labels. Recently, an association-based video summarization scheme is proposed in [11]. Frequent sequential patterns are mined from a video and they are selected as representative frames of the video summary. Since we have different objectives, our frameworks are different. Furthermore, their sequential patterns do not require the frames be consecutive because they are interested in representative frames. Whereas we capture the temporal information of videos in terms of the state transitions, which requires the states to be consecutive.

7. CONCLUSIONS

In this paper, we proposed an association-based classifier, which is called *VRules*, for performing classification on videos. This classification model mines frequent *state transition patterns* to discriminate video categories. The state transition patterns capture not only the temporal association between video states, but also the semantic association between features. Rules relating state transition patterns to class labels are prioritized and selected as the classifier according to their predictive power. We implemented the classification model and conducted experiments to verify its performance. The experimental results show that the association-based classifier has comparable performance on video classification.

Acknowledgements. Thanks to Mr Yi Haoran at Nanyang Technological University, Singapore for providing the source codes of their HMM-based video classification approach and offering a lot of help.

8. REFERENCES

[1] C. Djeraba. Association and content-based retrieval. In *IEEE Transactions on Knowledge and Data Engineering, vol.15, No.1*, 2003.

[2] J. Huang, Z. Liu, and Y. Wang. Joint video scene segmentation and classification based on hidden markov model. In *Proceedings of Advances in Digital Libraries Conference, Santa Barbara, CA*, 1998.

[3] W. Lin and A. Hauptmann. News video classification using svm-based multimodal classifiers and combination stragegies. In *Proceedings of SIGMM*, 2002.

[4] B. Liu, W. Hsu, and Y.Ma. Integrating classification and association rule mining. In *Proceedings of SIGKDD*, 1998.

[5] C. Lu, M. Drew, and J. Au. An automatic video classification system based on combination of hmm and video summarization. In *International Journal of Smart Engineering System Design, vol.5, no.1*, 2003.

[6] G. Wei, L. Agnihotri, and N. Dimitrova. Tv program classification based on face and text processing. In *Proceedings of IEEE multimedia and Expo, New York*, 2000.

[7] H. Yi, D. Rajan, and L. Chia. An efficient video classification system based on hmm in compressed domain. In *Proceedings of 4th PCM*, 2003.

[8] H. Yi, D. Rajan, and L. Chia. A unified approach to detection of shot boundaries and subshots in compressed video. In *Proceedings of IEEE ICIP*, 2003.

[9] M. J. Zaki and C. C. Aggarwal. Xrules: an effective structural classifier for xml data. In *Proceedings of SIG KDD, Wasington, DC, USA*, 2003.

[10] W. Zhou, A. Vellaikal, and C. JayKuo. Rule-based video classification system for basketball video indexing. In *Proceedings of ACM multimedia workshop*, 2000.

[11] X. Zhu and X. Wu. Sequential association mining for video summarization. In *Proceedings of IEEE ICME*, 2003.