LEARNING VIDEO PREFERENCES USING VISUAL FEATURES AND CLOSED CAPTIONS

by

DARIN BREZEALE

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2007

ACKNOWLEDGEMENTS

I have worked with Diane Cook for a number of years now, first as a Masters student and now for a PhD. She has always gone out of her way to be helpful and supportive, providing those gentle nudges to get me working again when needed. I will always be thankful for the help and guidance that she has provided over the years.

Each of the other members of my committee—Gautam Das, Doyle Hawkins, Jean Gao, and Manfred Huber—as well as Alp Aslandogan and JungHwan Oh, who began as committee members but were unable to continue, has provided feedback that I have found helpful during this process. In particular, Dr. Huber has provided many suggestions and comments that caused me to think about things that I had neglected.

I also would like to thank the following members of the Computer Science Engineering department staff—Camille Costabile, Will Griffith, Pam McBride, and Sherri Warwick. They work behind the scenes, keeping the wheels of the bureaucracy moving smoothly, with little thanks for their efforts. Each of them has been a pleasure to work with.

November 13, 2007

ABSTRACT

LEARNING VIDEO PREFERENCES USING VISUAL FEATURES AND CLOSED CAPTIONS

Publication No. _____

Darin Brezeale, Ph.D.

The University of Texas at Arlington, 2007

Supervising Professor: Diane J. Cook

Viewers of video now have more choices than ever. As the number of choices increases, the task of searching through these choices to locate video of interest is becoming more difficult. Current methods for learning a viewer's preferences in order to automate the search process rely either on video having content descriptions or on having been rated by other viewers identified as being similar. However, much video exists that does not meet these requirements. To address this need, we use hidden Markov models to learn the preferences of a viewer by combining visual features and closed captions. We validate our approach by testing the learned models on a data set composed of features drawn from movies and user ratings obtained from publicly available data sets.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

ABSTRACT iii

LIST OF FIGURES vi

LIST OF TABLES viii

Chapter

1. INTRODUCTION 1

    1.1 Hypothesis 3

    1.2 Solution 4

2. BACKGROUND 5

    2.1 Closed Captions 5

    2.2 Visual Features 7

        2.2.1 Color-Based Features 9

        2.2.2 MPEG 10

        2.2.3 Shot-Based Features 13

        2.2.4 Object-Based Features 15

        2.2.5 Motion-Based Features 17

    2.3 Dimensionality Reduction 19

        2.3.1 Discrete Wavelet Transform 20

        2.3.2 Random Projection 22

    2.4 Clustering 23

        2.4.1 Distance Measures 23

        2.4.2 Hierarchical Clustering 25

2.4.3 K-Means 28
2.4.4 Determining the Number of Groups in the Data 28
2.5 Hidden Markov Models 31
3. RELATED WORK 35
3.1 Video Recommendation 35
3.2 Classification of Video by Genre 38
4. METHODOLOGY 44
5. EXPERIMENTS 52
5.1 Data Sets 52
5.2 Preliminary Experiments 53
5.2.1 Preliminary Experiments Using Closed Captions 54
5.2.2 Preliminary Experiments using DCT Coefficients 55
5.2.3 Results and Discussion 58
5.3 User Modeling with Hidden Markov Models and Hierarchical Clustering 59
5.3.1 Results and Discussion 65
5.4 User Modeling with Hidden Markov Models and k-Means Clustering . 69
5.4.1 Results and Discussion 72
6. CONCLUSIONS 77
Appendix
A. MOVIE DATA 79
REFERENCES 85
BIOGRAPHICAL STATEMENT 97

LIST OF FIGURES

Figure		Page
2.1	Example of a closed caption set from *Lethal Weapon*	5
2.2	A series of frames	12
2.3	Two consecutive frames showing a macroblock (left) and its new location (right)	12
2.4	A ball thrown into the air (left) and the optical flow from two of the frames for the sequence (right)	17
2.5	Frame from Clean Slate	21
2.6	One level of wavelet transformation of frame	21
2.7	One level of wavelet transformation of frame (intensity image)	22
2.8	Example of dendrogram	25
2.9	Example of complete linkage	27
2.10	Comparison of linkage methods for Exploratory Data Analysis	29
2.11	Example of hidden Markov model	32
4.1	Example of two closed caption sets from *Star Trek: First Contact*	45
4.2	Overview of the data processing phase during training	48
4.3	Example of observation symbol production	49
4.4	Overview of the training process from clustering to HMM construction	50
4.5	Overview of testing process	51
5.1	Frame from *Sliders*	57
5.2	Frame from *Sliders* in which all values in a block use the DC term	57
5.3	Hierarchies constructed for the wavelet-transformed features for a typical viewer	62

5.4 Process for combining closed captions and visual features to form observation symbols 63

LIST OF TABLES

Table		Page
2.1	Some common distance measures	24
5.1	Summary of preliminary results using Closed Captions.	55
5.2	Summary of preliminary results using DCT Coefficients.	58
5.3	Comparison of features in initial hybrid approach	65
5.4	Results per number of movies rated in initial hybrid approach	66
5.5	Comparison of DCT and Wavelet coefficients in preliminary results.	69
5.6	Comparison of the number of ratings per user for the GroupLens and Netflix data sets.	70
5.7	Comparison of features from HMM and k-means clustering	72
5.8	Results per number of movies rated in hybrid approach using k-means	74
A.1	Movie Titles and Genre	80

CHAPTER 1

INTRODUCTION

People today have access to more video than at any time in history. Sources of video include television broadcasts, movie theaters, movie rentals, video databases, and the Internet. While many video choices come from the entertainment domain, other types of video are becoming more common including medical [FLXW04] and other types of educational lectures [vid07].

As the number of video choices increases, the task of searching for video of interest is becoming more difficult. One approach that viewers take is to search for video within specific genres. In the case of entertainment video, the genre of the video is provided when the video is released. However, there is much video that is unclassified. This has led to research in automatically classifying video by genre. While knowing the genre of video is helpful, the large number of video choices within many genres still makes finding video of interest a time-consuming process. In addition, this problem is even greater for people who enjoy video from a variety of genres, which seems likely for most people. For these reasons, systems have been developed that can learn a particular person's preferences and make recommendations given these preferences.

There have traditionally been two approaches to identifying video of interest to a viewer. The first is the case-based approach, which utilizes descriptions of the video content [Eri97] [AGT+04] [ZKB+04]. In the case of entertainment video, the description might include the genre of the video, director, actors, and a brief summary of the video. The second is collaborative filtering, which attempts to identify viewers

that are considered similar by some measure. Recommendations for the current viewer will be drawn from the positively rated videos of these similar viewers.

The major strength of the case-based approach is that it relies strictly on the viewer's profile. Once a viewer's preferences are known, it is a simple task to match these up with video content descriptions. There are, however, several weaknesses to the case-based approach. One is that it takes some effort to produce content descriptions. While this is typically not a problem when dealing with entertainment video such as television or movies, there is much video in video databases and on the Internet for which there are no content descriptions. Another weakness is that the viewer must initially seed the system with some preference information. A viewer may not wish to devote the time and effort to provide enough preference details for the system to perform well. A third weakness is that recommendations will be very similar to previously rated video.

The second approach is collaborative filtering. Collaborative filtering does not require the content descriptions used by the case-based approach. Also, unlike the case-based approach, video recommendations are not restricted to video similar to that previously rated by the user if the group the viewer is assigned to has a greater variety of interests. However, it does take some effort to gather enough information about other viewers in order to determine who is similar to the current viewer. A second weakness of collaborative filtering is the latency of a new video spreading; a video can't be recommended if no one has seen and rated it yet.

There are many videos that lack the content descriptions required by the case-based approach as well as the ratings of other viewers required by collaborative filtering. The approach we have chosen to handle videos in this category is to extract visual features and closed captions from video in order to learn a viewer's preferences. While we believe our approach is most useful in those situations that preclude the use

of collaborative filtering or case-based methods, it is also applicable in the situations for which the other approaches are appropriate or could be used to supplement those approaches.

Individually, each of these types of features has limitations. Some methods for representing text or images suffer from a lack of context. The bag-of-words model, which is a common method for representing documents, does not maintain word order and as a result two documents with essentially the same words but different word order can have different meanings but appear similar when comparing their term-feature representations. Likewise, two different images may appear similar when represented as color histograms. By combining text and visual features, we believe that these limitations can be lessened.

The visual features and closed captions are combined to produce observation symbols for training hidden Markov models (HMM). A video is a collection of features in which the order that the features appear is important, which suggests that an HMM might be appropriate for classification. We believe that visual features and closed captions are complementary. Visual features represent what is being seen, but miss much of the social interaction. Video dialogue typically doesn't describe what is being seen, but represents the social interaction.

1.1 Hypothesis

Our hypothesis is that features extracted from video rated by a viewer are sufficient to learn a viewer's preferences such that recommendations can be made with an accuracy exceeding what would be expected by choosing video at random.

1.2 Solution

Our solution is to model a viewer's preferences using hidden Markov models with observation symbols generated by combining closed captions and visual features. We will test our hypothesis on a data set of entertainment video and publicly available viewer ratings. In addition, we will compare the results from the combination of visual and text features to the results obtained from using each type of feature individually.

CHAPTER 2

BACKGROUND

In this chapter, we provide a discussion of the background material needed to understand our work. Specifically, we discuss visual-based and text features derived from video, dimensionality reduction, hierarchical clustering and modeling using hidden Markov models.

2.1 Closed Captions

Closed captioning is a method of letting hearing-impaired people know what is being said in a video by displaying text of the speech on the screen. Closed captions are found in Line 21 of the vertical blanking interval of a television transmission and require a decoder to be seen on a television [Rob04]. On a DVD the closed captions are stored in sets with display times. Figure 2.1 shows the 108^{th} set of closed captions for the movie *Lethal Weapon*.

The Telecommunications Act of 1996, which took effect in 1998, placed closed captioning requirements on television shows broadcast in the United States. With some exceptions, the law required that broadcasters begin providing closed captions on their broadcasts with a goal of 100% of all broadcast hours of new (first broadcast

108
00:07:51,821 −−> 00:07:54,824
DO YOU KNOW A MAN
NAMED MICHAEL HUNSAKER?

Figure 2.1. Example of a closed caption set from *Lethal Weapon*.

in 1998 or later) television shows by 2006 and 75% of older (first broadcast prior to 1998) television shows by 2008.

In addition to representing the dialog occurring in the video, closed captioning also displays information about other types of sounds such as onomatopoeias (e.g., grrrr), sound effects (e.g., [BEAR GROWLS]), and music lyrics (enclosed in music note symbols, ♪). At times, the closed captions may also include the marks >> to indicate a change of speaker or >>> to indicate a change of topic [GGP00].

One advantage of text-based approaches is that they can utilize the large body of research conducted on document text classification [Seb02]. Another advantage is that the relationship between the features (i.e., words) and specific genre is easy for humans to understand. For example, few people would be surprised to find the words 'stadium', 'umpire', and 'shortstop' in a transcript from a baseball game.

However, using closed captions does have some disadvantages. One is that the text available in closed captions is largely dialog; there is little need to describe what is being seen. For this reason closed captions do not capture much of what is occurring in a video. A second is that not all video has closed captions. A third is that while extracting closed captions is not computationally expensive, generating the feature vectors of terms and learning from them can be computationally expensive since the feature vectors can have tens of thousands of terms.

A common method for representing text features is to construct a feature vector using the bag-of-words model [For03]. In the bag-of-words model, each feature vector has a dimensionality equal to the number of unique words present in all sample documents (or closed caption transcripts) with each term in the vector representing one of those words. Each term in a feature vector for a document will have a value equal to the number of times the word represented by that term appears in the

document. One potential drawback of the bag-of-words model is that information about word order is not kept.

Representing a transcript may require a feature vector with dimensions in the tens of thousands if every unique word is included. To reduce the dimensionality, stop lists and stemming are often applied prior to constructing a term feature vector. A stop list is a set of common words such as 'and' and 'the' [FBY92]. Such words are unlikely to have much distinguishing power and are therefore removed from the master list of words prior to constructing the term feature vectors, which also has the benefit of reducing the computational requirements. Stemming removes the suffixes from words leaving the root. For example, the words 'independence' and 'independent' both have 'indepen' as their root. The stemmed words are used to generate the feature vectors instead of the original words. One of the more common methods for stemming is using Porter's stemming algorithm [Por80].

Another common approach is to weight each term using an approach known as the term frequency-inverse document frequency (TF-IDF) approach [TI94]:

$$\text{TF-IDF} = TF(d,t) \times IDF(t)$$

where $TF(d,t)$ is the frequency of term t in document d and $IDF(t)$ is

$$IDF(t) = log\left(\frac{N}{df(t)}\right)$$

where N is the total number of documents and $df(t)$ is the number of documents containing term t [TI94].

2.2 Visual Features

A variety of features can be obtained from the visual part of a video, as demonstrated by the video retrieval and classification fields [AY99], [Bim99]. Some choices

of features are color, texture, objects, and motion, which we will describe in detail in the next few sections. Visual features may correspond to cinematic principles or concepts from film theory. For example, horror movies tend to have low light levels while comedies are often well-lit. Motion might be a useful feature for identifying action movies, sports, or music videos; low amounts of motion are often present in drama. The type of transition from one video shot to the next can affect mood [Old92].

Visual features are often extracted on a per frame or per shot basis. A video is a collection of images known as frames. All of the frames within a single camera action are called a shot. A scene is one or more shots that form a semantic unit.[1] For example, a conversation between two people may be filmed such that only one person is shown at a time. Each time the camera appears to stop and move to the other person represents a shot change, but the collection of shots that represent the entire conversation is a scene.

A shot is a natural way to segment a video and each of these segments may represent a higher-level concept to humans, such as "two people talking" or "car driving down road". Also, a shot can be represented by a single frame, known as the keyframe. Typically the keyframe is the first frame of a shot, although some authors use the term to refer to any single frame that represents a shot. Shots are also associated with some cinematic principles. For example, movies that focus on action tend to have shots of shorter duration than those that focus on character development [VL00]. One problem with using shot-based methods is that the methods for automatically identifying shot boundaries don't always perform well [Lie99]. Identifying scenes is even more difficult and there are few video classification approaches that do so.

[1] In rare cases a single shot may contain more than one scene.

2.2.1 Color-Based Features

Color-based features are simple to implement and inexpensive to process. They are useful in approaches wishing to use cinematic principles. For example, amount and distribution of light and color set mood [RSS03].

A video frame is composed of a set of dots known as pixels and the color of each pixel is represented by a set of values from a color space [Poy96]. Many color spaces exist for representing the colors in a frame. Two of the most popular are the red-green-blue (RGB) and hue-saturation-value (HSV) color spaces. In the RGB color space, the color of each pixel is represented by some combination of the individual colors red, green and blue. In the HSV color space, colors are represented by hue (i.e., the wavelength of the color percept), saturation (i.e., the amount of white light present in the color), and value (also known as the brightness, value is the intensity of the color) [Bim99].

The distribution of colors in a video frame is often represented using a color histogram, that is, a count of how many pixels in the frame exist for each possible color. Color histograms are often used for comparing two frames with the assumption that similar frames will have similar counts even though object motion or camera motion will mean that they don't match on a per pixel basis. It is impossible to determine from a color histogram the positions of pixels with specific colors, so some authors will divide a frame into regions and apply a color histogram to each region to capture some spatial information.

Another problem with color-based features is that the images represented in frames may have been produced under different lighting conditions and therefore comparisons of frames may not be correct. The solution proposed by Drew and Au [DA00] is to normalize the color channel bands of each frame and then move them into a chromaticity color space. After more processing, including the application of

both wavelet and discrete cosine transforms, each frame is now in the same lighting conditions.

2.2.2 MPEG

One of the more popular video formats is MPEG (Motion Pictures Expert Group), of which there are several versions. We provide a somewhat high-level description of MPEG-1; for more complete details, consult the MPEG-1 standard [MPE91a].

During the encoding of MPEG-1 video, each pixel in each frame is transformed from the RGB color space to the YC_bC_r color space, which consists of one luminance (Y) and two chrominance (C_b and C_r) values. The values in the new color space are then transformed in blocks of 8×8 pixels using the discrete cosine transform.

Much of the MPEG-1 encoding process deals with macroblocks (MB), which consist of four blocks of 8×8 pixels arranged in a 2×2 pattern. Because the human eye is less sensitive to the chrominance components, these are sampled less frequently than the luminance component.

The DCT used in the MPEG-1 standard is

$$F(u,v) = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) cos\ \alpha_x\ cos\ \alpha_y$$

where

$$
\begin{aligned}
f(x,y) &= \text{value of the original block at coordinates } (x,y) \\
u &= 0,1,\ldots,7 \\
v &= 0,1,\ldots,7 \\
\alpha_x &= \frac{(2x+1)u\pi}{16} \\
\alpha_y &= \frac{(2y+1)v\pi}{16} \\
C_u &= \begin{cases} \frac{1}{\sqrt{2}} & \text{, if } u = 0 \\ 1 & \text{, otherwise} \end{cases} \\
C_v &= \begin{cases} \frac{1}{\sqrt{2}} & \text{, if } v = 0 \\ 1 & \text{, otherwise} \end{cases}
\end{aligned}
$$

The upper left corner of the block of DCT coefficients has coordinates $(0,0)$ and the lower right corner has coordinates $(7,7)$. It can be seen from the equation that for coordinates $(0,0)$, the DCT produces a value that is proportional to the average value. This value is known as the DC term while the other 63 values are known as the AC terms. While each block has 64 DCT coefficients, for natural images most of the energy of the block is concentrated in a few terms in the upper left corner. That is, most of the information needed to reconstruct the block is found in these terms. One of the ways that compression is achieved in MPEG-1 video is that the DCT coefficients with little energy are discarded [Sym04].

Each frame in the MPEG-1 format is classified as either an I-frame, a P-frame, or a B-frame [2] depending on how it is encoded. I-frames contain all of the information needed to decode the frame. Consecutive frames within the same shot are often very similar and this temporal redundancy can be exploited as another means of

[2] The standard also supports a D-frame in which only the DC coefficients are stored.

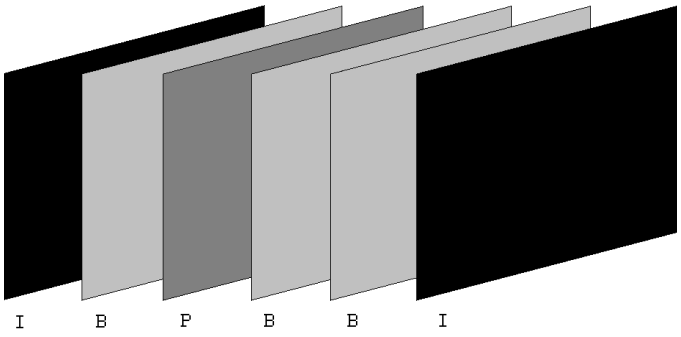

Figure 2.2. A series of frames.

compressing the video. P-frames are encoded with respect to the previous I-frame or P-frame. B-frames can make use of of information from previous or future frames, or a combination of the two [Gha03]. See Fig. 2.2 for an example of an arrangement of frames. Macroblocks from the frame(s) being referenced may appear in the P-frame or B-frame being encoded, although not necessarily at the same location. After a referenced macroblock is located in the frame being encoded, a motion vector is calculated for projecting the referenced macroblock to its new location. Fig. 2.3 shows a macroblock in a frame and its new location in the following frame.

Much research has been conducted on extracting features directly from MPEG video, primarily for the purpose of indexing video [KDLF97] [WDV+03] [SPN+05]. The primary features extracted from MPEG videos are the DCT coefficients and motion vectors. These can improve the performance of the classification system because

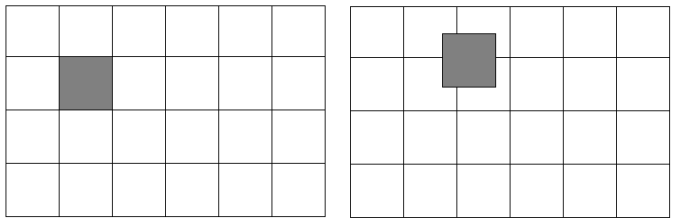

Figure 2.3. Two consecutive frames showing a macroblock (left) and its new location (right).

the features have already been calculated and can be extracted without decoding the video.

2.2.3 Shot-Based Features

In order to make use of shots, they first must be detected. This has proven to be a difficult task to automate, in part because of the various ways of making transitions from one shot to the next. Lienhart [Lie99] states that some video editing systems provide more than 100 different types of edits and no current method can correctly identify all types. Most types of shot transitions fall into one of the following categories: hard cuts, fades, and dissolves. Hard cuts are those in which one shot abruptly stops and another begins [AECI00]. Fades are of two types: a fade-out consists of a shot gradually fading out of existence to a monochrome frame while a fade-in occurs when a shot gradually fades into existence from a monochrome frame. A dissolve consists of one shot fading out while another shot fades in; features from both shots can be seen during this process. While it is important to understand shot transition types in order to correctly identify shot changes, the shot transition types themselves can be useful features for categorization [WAD00].

One of the simplest methods for detecting shots is to take the difference of the color histograms of consecutive frames, with the assumption that the difference in color histograms of frames within the same shot will be smaller than the difference between frames of different shots [ZKS93]. This approach, while easy to implement, has a number of potential problems. One is deciding what threshold differences must exceed in order to declare a change in shots. Shots that contain a lot of motion require a higher threshold value than those with little motion. Also, the threshold value is likely to be different for different videos and even within the same video no particular value may correctly identify all shot changes [JCB01]. A threshold value that is too

low will identify shot changes that don't exist while a threshold value that is too high will miss some shot changes.

Iyengar and Lippman [IL97] detect shot changes using the Kullback-Leibler distance between histograms of consecutive frames that have been transformed to the rgb color space. The rgb values are calculated using

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}, \quad b = \frac{B}{R+G+B}$$

The Kullback-Leibler distance is calculated using

$$KL(p||q) = -\sum_{i=1}^{N} p(x_i) log \frac{q(x_i)}{p(x_i)}$$

where N is the number of bins in the histograms, $p(x_i)$ is the probability of color x_i for one frame and $q(x_i)$ is the probability of color x_i for the other frame.

Truong et al. [TDV00a] detect shot changes with shot transitions of the types hard cut, fade-in, fade-out, and dissolve [TDV00b]. Hard cuts are detected by using a global threshold to identify potential cuts, then a sliding window is applied to these frames using an adaptive threshold. Fade-ins and fade-outs are detected by first identifying monochrome frames and then checking if the first derivative of the luminance mean is relatively constant. Dissolves are identified when the first order difference of the luminance variance curve falls within a range calculated from the luminance variances of the shots preceeding and succeeding the dissolve.

Rasheed and Shah [RS02] detect shot changes using the intersection of histograms in the HSV color space. This method works best for hard cuts [RSS03].

Jadon et al. [JCB02] detect shot changes as well as shot transition types using a fuzzy logic based approach [JCB01]. Abrupt changes (i.e., hard cuts) are detected using the intersection of frame histograms in the RGB color space. Gradual changes are detected using pixel differences and intersection of color histograms. The pixel

difference between two consecutive frames is calculated using the Euclidean distance between corresponding pixels in the RGB color space. Gradual changes are further divided into fade-ins and fade-outs, which are detected using pixel differences, intersection of color histograms, and edge-pixel counts. After detecting edges using a Sobel edge detector, the difference in the number of pixels of edges between consecutive frames is used to identify fade-in and fade-out transitions. Each feature is fuzzified, that is, values are assigned to qualitative categories (e.g., categorize as negligible change, small change, or large change). Fuzzy rules are constructed from these features.

Lu et al. [LDA01] avoid detecting shots altogether, instead identifying keyframes using clustering after first transforming frames to a chromatic color space to put all frames under the same lighting conditions.

2.2.4 Object-Based Features

Object-based features are uncommon due to their computational requirements. When they are used, they tend to focus on identifying specific types of objects, such as faces [YLM+06] [WCY03]. Once objects are detected, features derived from them include dominant color, texture, size, and trajectory.

Object-based features can be costly and difficult to derive. Wei et al. [WAD00] report that detecting text objects is efficient enough to be applied to all video frames but that detecting faces is so expensive that they limited it to the first few frames of each shot. Most methods require that the objects be somewhat homogenous in color or texture in order to segment them correctly, which may also require confirmation from humans [HFF05]. Objects that changed shape, such as clouds, would also prove difficult to handle.

Dimitrova et al. [DAW00] and Wei et al. [WAD00] use an approach described in [WS99] for detecting faces. Using images in which the skin-tone pixels have been labeled, a model is learned for the distribution of skin-tones in the YIQ color space. The YIQ color space, a transform of gamma-corrected RGB values, is used in broadcast video [Jac05]. The skin-tone distribution model is used for identifying regions of skin-tone pixels, which are processed with morphological operations to smooth and combine isolated regions that are related. Finally, shape analysis is applied to identify faces.

Dimitrova et al. [DAW00] and Wei et al. [WAD00] both use an approach described in Agnihotri and Dimitrova [AD99] for identifying text objects within video frames. Using the luminance components of a frame, a process for enhancing the edges is performed followed by edge detection and the filtering of areas unlikely to contain text. Connected component analysis, which identifies pixels that are connected, is performed on the remaining areas to identify text boxes. Text boxes from the same line of text are merged. These text boxes can become objects to be tracked or passed to character recognition software.

Fan et al. [FLXW04] detect objects representing a high-level concept, such as gastrointestinal regions [FLE04]. The frames from sample clips containing examples of the high-level concept are segmented by identifying regions with homogeneous color or texture. Afterward, a medical consultant annotates those regions matching the high-level concept. Low-level features, such as dominant color and texture, are extracted for these regions and passed to a support vector machine for learning the relationship between features and concept.

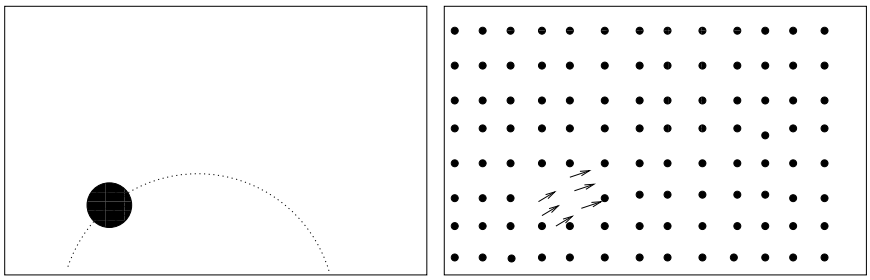

Figure 2.4. A ball thrown into the air (left) and the optical flow from two of the frames for the sequence (right).

2.2.5 Motion-Based Features

Motion within a video is primarily of two types: movement on the part of the objects being filmed and movement due to camera actions. In some specific types of videos, there might also be other types of movement, such as text scrolling at the bottom of a news program. Motion-based methods largely consist of the use of MPEG motion vectors or the calculation of optical flow.

Optical flow is an estimate of motion in a sequence of images calculated from the velocities of pixel brightness patterns, which could be due to object motion or camera motion, and is calculated from differences between two consecutive video frames. Figure 2.4 shows a ball thrown into the air from left to right (with a dotted line representing the motion of the ball that the viewer would observe in the preceding and following frames) and the corresponding plot of optical flow values derived from two of the video frames in the sequence.

There are many ways to measure optical flow [BFB94]. The method described by Horn and Schunck [HS81], which is used by several of the papers that we reviewed, determines the optical flow by solving two constraint equations. The gradient constraint equation finds the component of movement in the direction of the brightness

gradient. The second constraint is known as the smoothness constraint and is used to determine the component perpendicular to the brightness gradient.

Fischer et al. [FLE95] detect total motion in a shot by comparing the histograms of blocks of consecutive frames. In order to detect object motion, they first calculate optical flow as described in [HS81]. Motion due to camera movement (e.g., panning) would result in all blocks having motion. Using this, camera motion can be subtracted, leaving only the motion of objects. These objects are identified by segmenting pixels with parallel motion.

Nam et al. [NAT98] measure the motion density within a shot. A 2D wavelet transform is applied to each frame within a shot. A 1D wavelet transform is then applied to the intensity of each pixel in the sequence of 2D-wavelet transformed frames to produce a motion sequence [NT98], which is followed by calculating the dynamic activity within the shot.

Roach et al. [RMX02] detect the motion of foreground objects using a frame-differencing approach. Pixel-wise frame differencing of consecutive frames is performed using the Euclidean distance between pixels in the RGB color space. These values are thresholded to better represent the motion and doing so for the sequence of pixels produces a 1D signal in the time dimension. To reduce this signal's sensitivity to camera motions, it is differentiated to produce a final motion signal.

The quantity of motion in a video is useful in a broad sense, but it is not sufficient by itself in distinguishing between the types of video that typically have large quantities of motion, such as action movies, sports, and music videos [NAT98]. Calculating the quantity of motion in a shot includes using optical flow, MPEG motion vectors, or frame differencing. Optical flow is costly to calculate and may not match the direction of the real motion, if that is also required. The optical flow algorithm of Horn and Schunck has problems with occluded edges since they cause a

discontinuity in reflectance [HS81]. Extracting motion vectors from MPEG-encoded video is not costly, but of course requires that the video be in this video format in the first place. However, motion as indicated by motion vectors may be less accurate than motion as measured using optical flow [WDV+03]. Iyengar and Lippman [IL97] found that measuring motion using frame differencing produced results similar to those that measured motion using optical flow, yet frame differencing is simpler to implement and less computationally expensive. However, region-based features such as frame differencing are non-specific as to the direction of the motion [RMP01].

Measuring specific types of motion, such as object motion or camera motion, is also a difficult problem because of the difficulty in separating the two. Many approaches for measuring the motion of objects require that the object be segmented, which is a difficult task itself [RMP01]. Identifying object motion can be made easier if global motion can be detected and adjusted for. However, calculating global motion is costly and therefore some applications only apply calculations to regions [HFF05].

2.3 Dimensionality Reduction

Samples in a data set are often represented by a very large number of features, which may make learning difficult or be computationally infeasible to process [Pyl99]. This has led to methods for reducing the dimensionality of the data, either by eliminating variables thought to be irrelevant or by transforming each sample such that it is represented by less information while retaining the relationships between the samples [PHL04]. While many methods have been proposed for reducing the dimensionality of a data set, we only describe those utilized in this work [Fod02].

2.3.1 Discrete Wavelet Transform

The traditional approach to analyzing signals has been to use Fourier analysis, which consists of finding a representation of the signal as a sum of sines and cosines. An assumption of Fourier analysis is that the signal is periodic, which makes Fourier analysis inappropriate for signals that are not periodic [Gra95]. Like Fourier analysis, the purpose of wavelet analysis is to find a new representation of a signal; however, this representation is a sum of wavelets, which are waves that are short in duration [War00]. From the linear algebra perspective, wavelets are a set of basis functions; we wish to represent our signal in terms of these basis functions. The wavelet transform decomposes a signal into two signals: a trend (or weighted average) signal and a details signal, each having half the terms of the original signal [Wal99]. The wavelet transform can be applied recursively to each new trend signal.

Wavelet analysis can be applied to two-dimensional data, such as an image, by first applying the wavelet transform to each row (or column) of the image and then then transformed columns (or rows). Figure 2.5 shows a video frame from the movie *Clean Slate*. Figure 2.6 shows the same frame after applying 1-level of a Daub4 wavelet. The vertical details of the image are highlighted in the upper right quadrant of the image. The horizontal details of the image are highlighted in the lower left quadrant. The lower right quadrant represents the diagonal details of the image. The upper left quadrant represents the trend values. To make this area recognizable, the wavelet-transformed values were converted to an intensity image, which can be seen in Figure 2.7.

Wavelet analysis has a number of applications in image processing. By keeping only the trend signal values, the dimensionality of the original signal can be reduced. By comparing Figures 2.5 and 2.7, we can see that it is easy to recognize the reduced version of the original image while using 75% fewer values. In addition to reducing

Figure 2.5. Frame from Clean Slate.

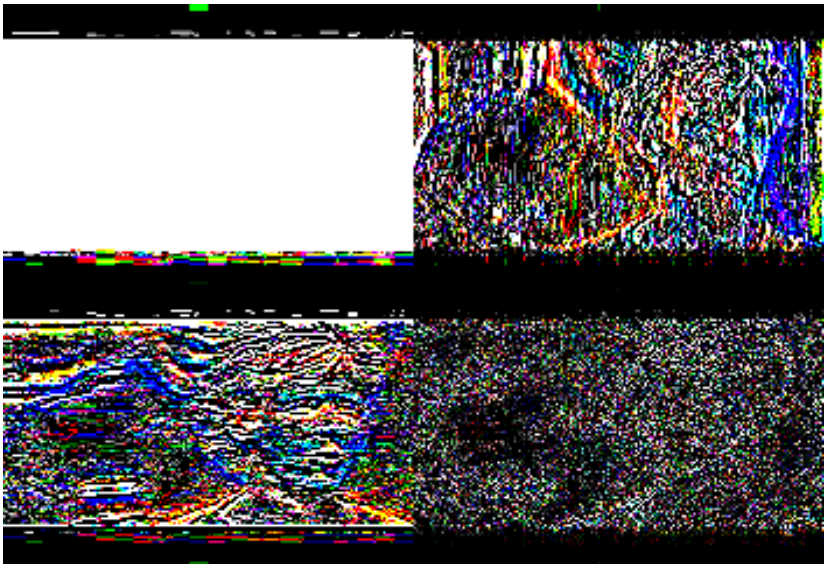

Figure 2.6. One level of wavelet transformation of frame.

Figure 2.7. One level of wavelet transformation of frame (intensity image).

the dimensionality, applications of wavelets to images have been shown to improve matching in image retrieval [JFS95]. The previously mentioned ability of wavelet analysis to separate the horizontal and vertical components of images has made it popular for edge detection [Wal99].

2.3.2 Random Projection

The idea of random projection is to project a set of points in a high-dimensional space to a randomly selected lower-dimensional subspace [Das00]. The application of random projection is simple: Given an input matrix X with dimensions $N \times d$ where N is the number of samples and d is the dimensionality of each sample, we can transform this matrix to a new matrix X' with dimensions $N \times k$ by multiplying X by a random matrix R with dimensions $d \times k$ such that $X' = XR$. Papadimitriou et

al. [PTRV97] show that there is a high probability that pairwise Euclidean distances are kept in the projected subspace.

Several ways of generating the transformation matrix R have been proposed. One way is to generate a matrix in which each element is drawn from a standard normal distribution, $N(0, 1)$. Then each column of this matrix is normalized such that the sum of the column values is one [FB03].

An advantage of random projection over principal component analysis (PCA), a popular dimensionality reduction method, is that PCA is very computationally expensive while generating and applying random projections is not [BM01]. Also, should the original matrix X be too large to work with in memory, it can be partitioned and the matrix R applied to the individual partitions with the results combined.

2.4 Clustering

Clustering is an unsupervised method of learning, that is, the class that each training sample belongs to is unknown to the system [HK06]. This contrasts with supervised learning, in which class labels are associated with the training samples. The difference is that supervised learning methods can take advantage of the class being known to find the relevant features for assigning a new object to a class. Unsupervised learning methods attempt to group similar objects without knowing which features are relevant for class membership and typically, without even knowing the number of classes.

2.4.1 Distance Measures

Each clustering method requires determining the similarity of objects, typically viewed as the distance between them. Some of the standard distance measures are the Manhattan, Euclidean, Minkowski, and Mahalanobis distances. The formulae for

Table 2.1. Some common distance measures

Name	Equation
Manhattan	$d_{ij} = \sum_{k=1}^{p} \lvert x_{ik} - x_{jk} \rvert$
Euclidean	$d_{ij} = (\sum_{k=1}^{p} (x_{ik} - x_{jk})^2)^{1/2}$
Minkowski	$d_{ij} = (\sum_{k=1}^{p} \lvert x_{ik} - x_{jk} \rvert^r)^{1/r}$
Mahalanobis	$D_{ij}^2 = (X_i - X_j)\Sigma^{-1}(X_i - X_j)$

each of these can be found in Table 2.1 where each individual is represented by a p-dimensional vector $X = \{x_1, x_2, \ldots, x_p\}$ and x_{ik} is the kth element of individual i and x_{jk} is the kth element of individual j.

The Manhattan (also known as the city block or l_1 norm) and Euclidean (or l_2 norm) distances are special cases of the Minkowski (l_r norm) distance with $r = 1$ and $r = 2$, respectively. If we think in terms of two-dimensional geometry, although each of the l_r norms can be calculated in n dimensions, then given right triangle ABC with the right angle being at point B, the distance between points A and C using the Euclidean distance as the metric is the hypotenuse of the triangle. The distance between these same two points with the Manhattan distance as the metric is the sum of the lengths of sides AB and BC.

The Mahalanobis distance, which adjusts for the covariance, calculates the distance between objects X_i and X_j by

$$D_{ij}^2 = (X_i - X_j)\Sigma^{-1}(X_i - X_j)$$

where Σ is the pooled within-group covariance matrix. The Mahalanobis distance becomes the Euclidean distance when Σ is the identity matrix [Bis06]. A major disadvantage of using the Mahalanobis distance is that it is very computationally expensive.

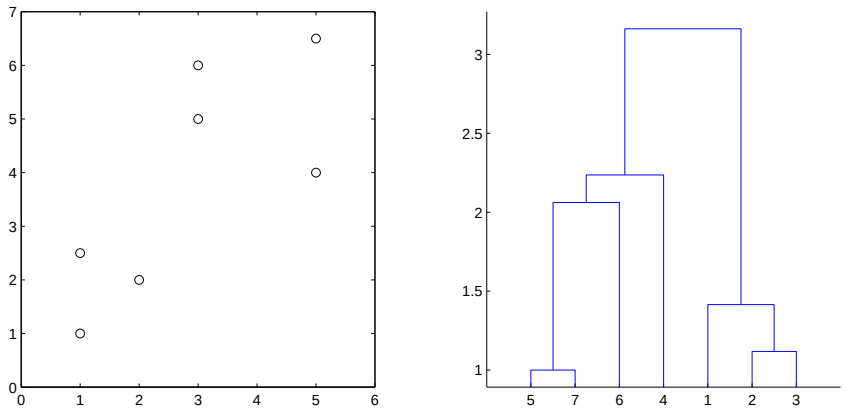

Figure 2.8. Example of dendrogram.

2.4.2 Hierarchical Clustering

Hierarchical clustering methods are of two types: agglomerative or divisive. Agglomerative clustering begins with all individuals separate. The two nearest individuals are joined into a group. Then the next two nearest objects (i.e., individual or group) are joined and this process is repeated until all individuals are members of the same group. Divisive clustering begins with a single group and divides it into smaller groups until eventually each member has been separated out. In both cases it is possible to represent the relationships as a tree, or dendrogram, which makes hierarchical clustering particularly popular for exploring relationships in the data [MM04]. Figure 2.8 shows a set of points in two dimensions and the corresponding hierarchical relationship.

We can see in the X-Y plot of the data that the points with coordinates (4, 5) and (4, 6) are the closest (at least according to the Euclidean distance, to be discussed below) and these two points are the first to be connected in the dendrogram, shown by the lowest horizontal connection. For this particular example, we can determine the order in which the objects were grouped by following the order of the horizontal

connections from bottom to top. The highest horizontal connection represents the combination of all of the points into a single group. The length of the vertical lines is an indication of the distance between two connected objects. Shorter vertical lengths represent objects that are closer.

Once a hierarchy has been constructed, a decision must be made as to where to prune (i.e., partition or cut) the tree in order to form the groups. For example, if the tree in Figure 2.8 is pruned by drawing a horizontal line a quarter of the way down from the top, then the two branches that are cut would each form a group. The membership of each group consists of all of the objects connected to the cut branch.

Regardless of whether the agglomerative or divisive method is chosen, to determine the similarity of two objects (an individual or group) consideration must be given to how to measure the distance between objects and how to use the distances. Of the methods that have been explored for determining how to use the distance between objects, some of the most common are single linkage, complete linkage, and average linkage [ELL01]. The single linkage method determines the similarity of two objects by calculating the distance between the nearest members of each object; the most similar object is the one for which this distance is the smallest. Because single linkage only relies on the distance to the nearest member of a group, a new object may not be near the group as a whole. Single linkage is computationally efficient but is subject to chaining.

The complete linkage method determines the similarity of two objects by calculating the distance between the farthest members of each object, with the most similar object being the one for which this distance is the smallest. This ensures that a new object will be close to all members of a group. However, a potential problem is that complete linkage is sensitive to outliers and in fact constructing a hierarchy using complete linkage can be used to find outliers. An example of complete linkage

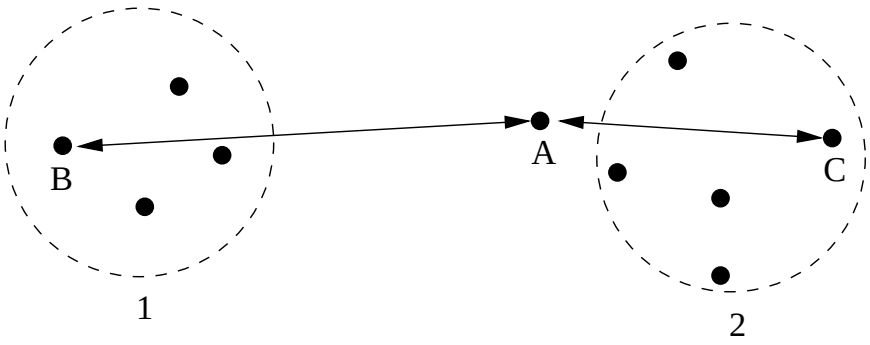

Figure 2.9. Example of complete linkage.

can be seen in Figure 2.9. In this example, there are two existing groups and point A is to be assigned to one of these groups. By the Euclidean distance, the member of group 1 with the maximum distance to A is B. The member of group 2 with the maximum distance to A is C. The minimum of these maximums is the distance from A to C, so A would be assigned to group 2.

Average linkage determines the similarity of two groups by calculating the average distance from each member of one group to each member of the other group. Average linkage is more robust to outliers than single or complete linkage [ELL01].

An advantage of hierarchical clustering over some other forms of clustering, such as k-means, is that it is unnecessary to know the number of clusters in advance. Once a hierarchy has been constructed of n objects, it can be partitioned into anywhere from 2 to n clusters. This is also an advantage in terms of computational performance. A disadvantage of hierarchical clustering is that once an individual has been merged into a group (agglomerative method) or separated from a group (divisive method), it can not be undone. Another disadvantage is that the data may not be hierarchical in nature and therefore hierarchical clustering imposes a structure that doesn't exist in the data.

2.4.3 K-Means

The k-means algorithm assigns objects to the group whose center it is closest to, with closest in this case typically being determined using the Euclidean distance [HK06]. k of the objects to be clustered are chosen as the initial group centers (some implementations may choose k random points in space instead). Then each of the remaining objects is assigned to the group whose center it is closest to. Afterwards, the mean value of each group is calculated as the new group center. Now each object is assigned to the group whose mean value it is closest to and a new group center is calculated. This process is repeated until no objects change group membership or until some stopping criterion is met. Different initial values for the group centers can result in some differences in the final group memberships.

k-means is a popular clustering method that is easy to understand and to implement. It attempts to produce clusters in which the intracluster distance is low but the intercluster distance is high. k-means is most suitable when the groups in the data are well-separated. One disadvantage of k-means is that it is sensitive to outliers. Another disadvantage of k-means clustering as compared to hierarchical clustering is that changes in the value of k require the entire clustering process to be repeated, which can make a search for the performance-maximizing value of k very computationally expensive.

2.4.4 Determining the Number of Groups in the Data

It is often unclear how many groups exist in a data set or what relationship exists between the groups. This is particularly true when looking at the raw data. For example, do the groups form nicely separated spheres or is there much overlap that will make discrimination difficult? When hierarchical clustering is used, which combination of linkage method and distance metric is best for a given data set in

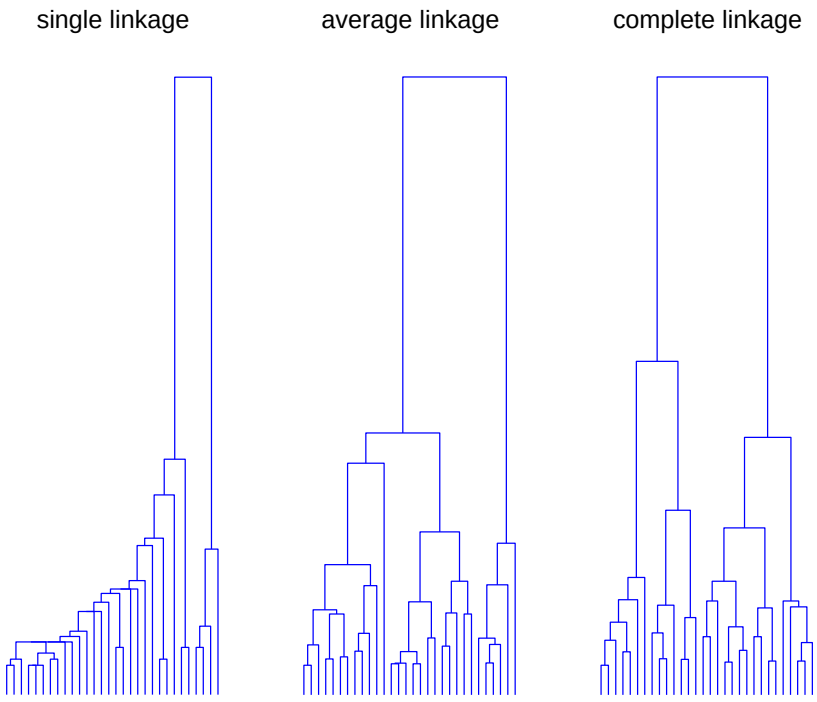

Figure 2.10. Comparison of linkage methods for Exploratory Data Analysis.

the sense that it clearly identifies the groups present in the data? One method for attempting to answer these questions is to use exploratory data analysis (EDA), which typically consists of making a visual inspection of the data in some form [MM04]. An example can be seen in Figure 2.10. In this example, generated from the famous *Iris* data set, dendrograms are generated for the hierarchies produced using the single, complete, and average linkage methods with the Euclidean distance as the distance metric.

As stated earlier, the length of the vertical bars in the dendrogram represents the distance between the two objects (i.e., individual or group) connected by a horizontal bar. Each of the dendrograms in Figure 2.10 have the greatest height in the last horizontal connection, which suggests that at least according to each of these

combinations of linkage method and distance metric, there are two groups present in the data set.

While visual inspection methods do serve a purpose, they are subjective in nature and also make it difficult to automate the process of identifying the number of groups present in the data set. Many methods have been proposed for automatically determining the number of groups within a data set, which also allow the choices to be more quantitative. Milligan and Cooper [MC85] compared 30 different methods using Monte Carlo simulation and concluded that the "best" metric is dependent on the data. We briefly mention the two methods used in this work for determining the number of groups in the data.

The method of Krzanowski and Lai [KL88] uses the following equation:

$$\mathrm{DIFF}(g) = (g-1)^{2/p}\ \mathrm{trace}(W_{g-1}) - g^{2/p}\ \mathrm{tr}(W_g)$$

where g is the number of groups, p is the number of dimensions, and W_g is the within-group-sum-of-squares covariance matrix for group g. By this measure, the number of groups in the data is the value of g that maximizes

$$\mathrm{KL}(g) = \left| \frac{\mathrm{DIFF}(g)}{\mathrm{DIFF}(g+1)} \right|$$

One disadvantage of this method is that it is unable to determine if the data only represents a single cluster [TWH01].

Tibshirani et al. [TWH01] proposed the Gap statistic. First, the sum of the pairwise distances between all group members of each of the k clusters is calculated:

$$D_r = \sum_{i,i' \in C_r} d_{ii'}$$

where D_r is the sum of the distances for cluster r. This is followed by calculating

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r$$

where k is the number of clusters, n_r is the number of members of cluster r, and W_k is the pooled within-group-sum-of-squares if the squared Euclidean distance is used for measuring the distance between group members. This is compared to a null reference distribution [MM04] to determine the number of groups present in the data.

2.5 Hidden Markov Models

In this section, we provide an overview of hidden Markov models for classification. For a much more detailed discussion of hidden Markov models, see Rabiner [Rab89].

The hidden Markov model (HMM) is widely used for classifying sequential data, in particular temporal processes, and for modeling probabilistic processes. An HMM represents a set of states and the probabilities of making a transition from one state to another state [RJ86]. While in each state, an observation symbol can be generated with some probability. The model is 'hidden' because the true number of states and which state the model is in are unknown; only the observation symbols being generated are known with certainty.

Formally, an HMM is represented by $Q = 1, \ldots, N$ states, each generating $V = 1, \ldots, M$ observation symbols, an $N \times N$ matrix A of transition probabilities where a_{ij} is the probability of moving to state j while in state i, an $N \times M$ matrix B of observation (or emission) probabilities where b_{ik} is the probability of generating symbol v_k while in state i, and a $1 \times N$ vector π of starting probabilities where i is the probability of beginning in state q_i.

These concepts can be demonstrated by the example of the Occasionally Dishonest Casino from Durbin et. al. [DEKM98]. In a dice game in a casino, the casino uses a fair die 95% of the time, but 5% of the time switches it for a loaded die. Once the casino has begun using the loaded die, the casino will continue to use it 90% of

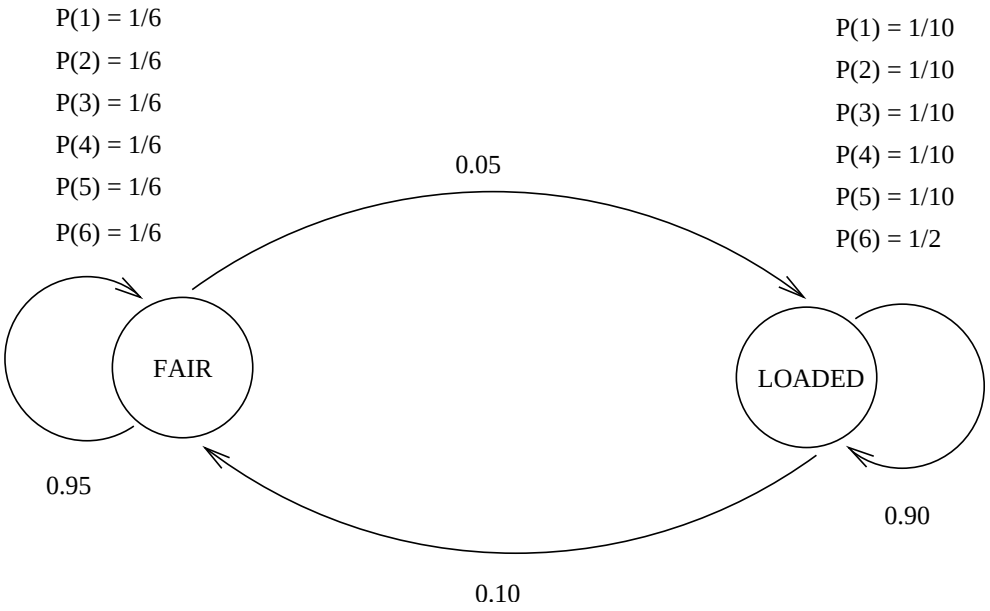

Figure 2.11. Example of hidden Markov model.

the time but 10% of the time will switch back to the fair die. The probability for each side of the fair die is 1/6. When using the loaded die, the probability of getting a 6 is 50% while the remaining five sides of the die each have a probability of 10%. We'll assume that the probabilities of the casino beginning with either die are equal. Figure 2.11 is a graphical representation of the HMM representing the Occasionally Dishonest Casino.

In more formal terms, this example has two hidden states $Q = \{\text{Fair Die, Loaded Die}\}$ and observation symbols $V = \{1, 2, 3, 4, 5, 6\}$. The starting, transition, and observation probabilities are

$$\pi = \left[\frac{1}{2}, \frac{1}{2}\right], \qquad A = \begin{bmatrix} 0.95, & 0.05 \\ 0.10, & 0.90 \end{bmatrix}, \qquad B = \begin{bmatrix} \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6}, & \frac{1}{6} \\ \frac{1}{10}, & \frac{1}{10}, & \frac{1}{10}, & \frac{1}{10}, & \frac{1}{10}, & \frac{1}{2} \end{bmatrix}$$

respectively.

To use hidden Markov models, one or more of the following problems must be dealt with [Rab89] [DHS01]:

- Evaluation Problem – Given an HMM with known transition probabilities, what is the probability that this model would have generated a particular sequence of observation symbols?
- Decoding Problem – Given an HMM and a set of observation symbols, what is the most likely sequence of hidden states for producing these observation symbols?
- Learning Problem – Given a set of observation symbols, what are the parameters for constructing the best model that generates these symbols?

In order to construct an HMM, we must deal with the Learning Problem. Figure 2.11 represents the true underlying model, but in many situations the actual model is unknown and only the observation symbols are known. To construct a model, we must guess the number of states (assuming they are unknown) and provide samples of sequences of observation symbols. We can test models with different numbers of states to determine which one best classifies our test samples. The choice of the number of states is important because too few will result in a model that doesn't accurately represent reality, but too many will result in a model that fits the particular data instead of generalizing (i.e., overfit) [Moo07]. Using the assumed number of states and the sequences of observation symbols, the model estimates the transition and observation probabilities through an iterative process. This is often accomplished using the Baum-Welch algorithm, which is a version of the Expectation Maximization (EM) algorithm. There is no guarantee that the learned parameters are optimal [DHS01].

It is typical to construct one HMM for each class to be predicted. The Evaluation Problem is the situation we are dealing with when using constructed models to classify test sequences (technically this assumes that $P(model_1) = \cdots = P(model_n)$). The observation sequence representing the sample to be classified is provided to each

HMM. The one that produces the highest likelihood (i.e., probability) for generating this sequence is the class to which the sample will be assigned.

CHAPTER 3

RELATED WORK

Research related to this work falls primarily into two categories. The first is the previously mentioned video recommendation. The second is automatic classification of video by genre because of the similarity in feature selection and classification.

While the application area is video recommendation, the general approach that we have taken in feature selection and classification is drawn from efforts to automatically classify video by genre.

3.1 Video Recommendation

One of the earliest approaches to video recommendation is the work of Karunanithi and Alspector [KA95] in which they compare a case-based approach (which they refer to as a feature-based approach) to a collaborative filtering approach (which they refer to as a clique-based approach). The case-based approach trains a neural network to predict movie ratings on a 1–10 scale. The features provided to the neural network are category (i.e., genre), MPAA rating, the rating provided by movie critic Leonard Maltin, Academy Award (i.e., won, lost, or not considered), length in minutes, and country of origin (i.e., USA, made in USA with foreign collaboration, or foreign) [AKK97]. The collaborative filter approach determines the similarity of two viewers by calculating Pearson's correlation between the movie ratings of each viewer. Results are reported as the correlation between the predicted and the actual ratings. For most of the viewers used in the study, the collaborative filtering approach produced the best results.

Basu et al. [BHC98] investigate a collaborative filtering approach, a case-based approach, and a hybrid approach. All classification was performed using inductive learning and the results were evaluated using precision and recall, with the emphasis placed on achieving high precision at the expense of recall. The collaborative filtering approach finds similar viewers strictly by which movies they like and dislike. The case-based approach uses 26 features (e.g., actors, genre, titles, and so forth) obtained from the Internet Movie Database (IMDb). To create a hybrid approach, the genre comedies, dramas, and action are isolated and the users who liked them are found. The proportion of specific genre of movies that a user has rated as liked is used to determine if that user likes a specific genre in general. The precision and recall are 77% and 27% for the collaborative filtering approach, 73% and 33% for the case-based approach, and 83% and 34% for the hybrid approach.

Smyth and Cotter [SC99] combine case-based and collaborative filtering approaches in order to offset the weaknesses of each. Their hybrid recommender requires that a user initially provide information about the types of television programs that they like and dislike. This information is used to find viewers with similar interests. Some recommendations are derived from the content information provided at registration while other recommendations are based on the preferences of the similar viewers. Smyth and Cotter measure performance by the percentage of users who receive N or more good recommendations per day, where $N = 1, 2, 3$. The collaborative filtering recommendations produce one or more good recommendations per day for 96% of users while the case-based recommendations produce one or more good recommendations for 78% of users.

Kurapati et al. [KGS+01] use a case-based approach to make television program recommendations by constructing three agents. An implicit preferences agent analyzes the viewer's viewing habits. An explicit preferences agent learns preferences

by asking the viewer questions about general interests, such as preferred genre, channels, and time-of-day for watching television. A feedback agent allows the viewer to provide specific information, such as favorite actors and television programs. Both Bayesian and decision tree classifiers were investigated for constructing the implicit preference agent.

Ardissono et al. [AGT+04] combine three user models. The Explicit user model is constructed from a form the viewer fills out requesting demographic information, general interest in topics such as books and politics, and TV program preferences. The Stereotypical user model uses information obtained in constructing the Explicit user model, which in turn is used to determine how well the viewer matches a number of pre-existing categories, or stereotypes, of TV viewers. Finally, the Dynamic user model is constructed by observing the viewer's TV viewing habits. In particular, the day and time of TV programs is monitored as well as the types of TV programs watched. The preferences derived from these three user models are combined using a weighted sum. A precision of 0.8 and a mean absolute error rate of 0.3 were achieved.

The fusion of these three user models has several strengths. The input of the user who constructed the Explicit user model allows the system to begin making recommendations immediately as well as to match the user to existing viewer profiles (the Stereotypical user model). The Dynamic user model allows the system to learn new preferences over time. However, this approach has several weaknesses as well. The user may not wish to spend the time necessary to provide the initial preference information. Without much initial preference values, the Explicit and Stereotypical user models will be limited in their usefulness. The Dynamic user model allows the system to learn over time, but it considers the time and day of viewing as important features. We hypothesize that digital video recorders, which make recording television

programs for later viewing easy, will reduce the importance of time and day as features since a viewer's choices will not be limited to what is on at a certain time of the day.

Zimmerman et al. [ZKB+04] extend the work of Kurapati el al. [KGS+01] by using a radial basis function neural network to fuse five individual recommenders. The individual recommenders consisted of two that learned implicitly from individual viewing history (one each using Bayesian or decision tree classifiers), two that learned implicitly from household viewing history (one each using Bayesian or decision tree classifiers), and one that learned explicitly.

We believe that our approach has several advantages over existing methods. It does not require that a viewer provide any information about his preferences other than a rating for a video that he has viewed. This saves time as well as avoids poor recommendations that might occur due to omissions in the preference description. Another benefit is that it is unnecessary to identify similar viewers. A third is that, as mentioned earlier, there are situations in which neither case-based nor collaborative filtering approaches are applicable and the only choice is to analyze the video itself.

Our approach does have some known disadvantages, which are not trivial. One is that some video may not have closed captions nor may it be possible to automatically generate a transcript using speech recognition. If this is the case, preferences could be learned using the visual features alone. Another is that initially the system would not know anything about the viewer's preferences and would require that the viewer locate enough video of interest to learn preferences.

3.2 Classification of Video by Genre

Approaches to classification of video by genre use features from three modalities: audio, visual, or text. We only discuss classification methods that utilize visual or

text features because of their relationship to this work. For a survey of the video classification literature, see Brezeale and Cook [BCar].

Zhu et al. [ZTL01] classify news stories using features obtained from closed captions. News video is segmented into stories using the topic change marks (explained in Chapter 2) inserted by the closed caption annotator. A natural language parser is used to identify keywords within a news segment and the first 20 unique keywords are kept. A weighted voting scheme involving the conditional probabilities of classes and keywords is used to classify the news segment.

Brezeale and Cook [BC06] perform classification using text and visual features separately; we describe the use of text features here. The text features are the closed captions from DVDs. After processing the closed caption text with a stop list and stemming, classification was performed using a support vector machine. There were fifteen genres of movies.

Lin and Hauptmann [LH02] combine classifiers of visual and text features. A video is divided into shots and a keyframe is extracted from each. Each keyframe is represented by a vector of the color histogram values in the RGB color space. A support vector machine (SVM) is trained on these features. For each shot, the closed captions are extracted and represented as a vector. For these vectors, another SVM is trained. Two methods for combining classifiers are investigated. The first method is based on Bayes' theorem and uses the product of the posterior probabilities of all classifiers. Performance is improved by assuming equal prior probabilities. The second method uses an SVM as a meta-classifier for combining the results of the other two SVMs. Both methods had similar recall, but the SVM meta-classifier had statistically significant higher precision.

Lin and Hauptmann combine both visual and text-based features for classification and in that regard their approach is very similar to our approach described in

this work. Our approach differs from that of Lin and Hauptmann in the choice of machine learning algorithms employed as well as our consideration of the temporal relationship between features. We also use clustering to reduce the feature space from which to produce observation symbols for the HMMs.

Wang et al. [WCY03] classify news video into one of ten categories. Classification is performed primarily using text features. The spoken text from news stories is extracted using speech recognition. Audio features are extracted from one second clips. Forty-nine audio features are produced including mel-frequency cepstral coefficients, high zero crossing rate ratio, and bandwidth. From each video shot, 14 features are produced including the number of faces, display of closed captions, shot duration, and motion energy.

Using the text features, an SVM produces a confidence vector for each news story. A confidence vector is produced by one Gaussian mixture model (GMM) per class using the audio features. A confidence vector is also produced by one GMM per class using the visual features. If the confidence vector produced by the SVM using text features exceeds some threshold, then it is used to classify the news story. If not, then an SVM-based meta classifier is used with the input being the concatenation of the text, audio, and visual confidence vectors.

Qi et al. [QGJ+01] classify a stream of news video into types of news stories. Audio and visual features are first used to detect video shots and then these shots are grouped into scenes if necessary. The closed captions and any scene text detected using optical character recognition (OCR) are the features used by a support vector machine for classifying the news stories.

Rasheed et al. [RSS03] used low-level visual features to classify movie previews by genre. Features are chosen with specific cinematic principles in mind. The features are average shot length to measure the tempo of the scene, shot motion content to

determine amount of action, lighting key to measure how well light is distributed (i.e., are there shadows) and color variance, which is useful for distinguishing between such genre as comedies (which tend to have bright colors) and horror movies (which tend to be dark).

Clustering is performed using mean-shift clustering. This method is chosen because it can automatically detect the number of clusters and it is non-parametric, so it is unnecessary to make assumptions about the underlying structure. The genre studied are action, comedy, drama, and horror, but the authors kept in mind that some movies may fall into several genre categories. The mean-shift clustering approach produced six clusters, which are labeled action+drama, drama, comedy+drama, comedy, action+comedy, and horror.

Nam et al. [NAT98] focus specifically on identifying violent video shots. First, the motion within a shot is measured by applying a 2D wavelet transform to each frame within a shot. A 1D wavelet transform is then applied to the intensity of each pixel in the sequence of 2D-wavelet transformed frames. Those shots whose motion exceeds some predetermined threshold are identified as action shots. Violent action shots are differentiated from non-violent action shots by identifying audio and visual signatures associated with gunfire and explosions. Specifically, violence is identified using the colors associated with flames and blood as well as the energy entropy of bursts of sounds produced by gunfire and explosions.

Dimitrova et al. [DAW00] classify four types of TV programs: news, commercials, sitcoms, and soap operas. Faces and text are detected and tracked. Counts of the number of faces and text are used for labeling each frame of a video clip. An HMM is trained for each class using the frame labels as the observations.

Lu et al. [LDA01] classify a video by first summarizing it. The color channel bands of each frame are normalized and then moved into a chromaticity color space.

After more processing including both wavelet and discrete cosine transforms, each frame is now in the same lighting conditions [DA00]. A set of twelve basis vectors determined from training data can now be used to represent each frame. A hierarchical clustering algorithm segments the video into scenes; the keyframes from the scenes represent the summarized video. One HMM is trained for each video genre with the keyframes as the observation symbols.

Huang et al. [HLW+99] combine audio and visual features for classifying video from the following classes: news reports, weather forecasts, commercials, basketball games, and football games. The audio features produced are as described in [LHW98]. The visual features are dominant color, dominant motion vectors, and the mean and variance of the motion vectors.

Four ways of using these features are investigated. In the first method, the audio and visual features are determined for each clip and concatenated into a single vector. The feature vectors for sequences of 20 clips are the input to HMMs, one for each video class. In the second method, audio, color, and motion features are produced for each video frame and a separate HMM is trained for each. The product of the observation probabilities for each these three types of features is used for classification. The third method uses two stages of HMMs. In the first stage, audio features are used to train HMMs for distinguishing between commercials, football or basketball games, and news reports or weather forecasts. In the second stage, visual features are used to train HMMs to distinguish football games from basketball games and news reports from weather forecasts. For the fourth method, for each of the three types of features (audio, color, motion), an HMM is trained for each class. The output from these HMMs becomes the input to a 3 layer perceptron neural network. The product HMM gave the best average classification accuracy.

Gibert et al. [GLD03] use motion and color to classify sports video into one of four classes: ice hockey, basketball, football, and soccer. Motion vectors from MPEG video clips are used to assign a motion direction symbol to each video frame. Color symbols are assigned to each pixel of each frame. A symbol for the most prevalent color is assigned to the entire frame. Unlike most other applications of HMMs for video classification, the authors train two HMMs for each video class: one for the frame color symbols and the other for the motion direction symbols. The output probability for each class is calculated by taking the product of the color and motion output probabilities for that class.

CHAPTER 4

METHODOLOGY

In this chapter we discuss conceptually the overall methodology of our final approach as well as the rationale for our choices. Specific implementation details are discussed in Chapter 5.

As stated in the Introduction, our intent is to learn a user's video preferences by constructing a model whose input is the set of features from the videos that this user has viewed and rated. We intend to represent each video by a combination of text features, specifically closed captions, and visual features while maintaining the temporal relationship of the features. Were we uninterested in the temporal relationship, there would still be a variety of ways to combine the text and visual features but an easy way to represent the video would be to create a vector for all of the closed captions and a vector for the visual features (most likely a subset of all of the visual features for computational reasons) and concatenate the two. However, the order in which text and visual features appear in a video are important to its meaning and therefore we believe that this temporal relationship should not be ignored. Therefore, in order to represent a video, we have two basic issues to address: 1) how to combine the text and visual features and 2) how to capture the temporal relationship of features.

A common approach to dealing with the first issue is to segment a video into shots and then represent each shot by the text and visual features that occur during this shot. As discussed in Section 2.2, a shot is a natural way of segmenting a video, but as was also discussed, attempts to automate this process with a high degree of

1573
01:34:21,963 −−> 01:34:23,765
RELAX, DOCTOR. I'M
SURE THEY'RE JUST HERE
1574
01:34:23,765 −−> 01:34:25,767
TO GIVE US A SENDOFF.

Figure 4.1. Example of two closed caption sets from *Star Trek: First Contact*.

accuracy have proven difficult. In our own attempts to automatically segment video into shots we found the process to be highly unreliable.

Because the text features that we are utilizing are closed captions, specifically those available on DVDs of movies, we have an alternative method for segmenting a video. Closed captions that are displayed on-screen at the same time, which we have termed *closed caption sets*, are stored on a DVD along with the time period for which the closed caption will be displayed. For other video sources that contain closed captions, such as television broadcasts, the time in which the closed captions appear could be ascertained from the elapsed time of the video.

By using these closed caption set display times, we can know exactly when during the course of a video certain text and visual features occur together. Therefore, our proposed methodology for combining text and visual features begins by extracting the closed caption sets from the training videos that the viewer has rated with the intent of using the times that the closed caption sets are displayed as the mechanism for segmenting the video. The display times are used only for segmentation and for finding a corresponding video frame from which to extract visual features. For example, Figure 4.1 shows the 1573^{rd} and 1574^{th} closed caption sets from the movie *Star Trek: Close Contact* with their timestamps. The timestamps for the 1573^{rd} closed caption set show that this set of closed captions will be displayed from the time period beginning at 1 hour, 34 minutes, 21 seconds, and 963 milliseconds of

the movie until 1 hour, 34 minutes, 23 seconds, and 765 milliseconds. For a movie encoded at 29.97 frames per second, we can calculate that this two second period begins at frame 169,689 and ends at frame 169,743. Visual features can be extracted from one or more of the video frames in this range.

One potential problem that can occur from creating a segment for each individual closed caption set is that the video may be over-segmented. The number of unique terms that occur in a closed caption set is typically less than twenty while the dimensionality of the feature vectors representing the text may have thousands of terms, resulting in term feature vectors in which the vast majority of terms have values of zero. The consequence of this is that there may be little overlap between the text feature vectors and therefore it is difficult to determine which feature vectors are similar. A solution to this problem is to combine several closed caption sets into a single segment, which is represented by a single vector. Specifically, M consecutive closed caption sets are combined to form a single feature vector (Section 2.1), with a new feature vector being constructed at every N^{th} closed caption set. Once the segmentation times have been determined, visual features are extracted from a single video frame (Section 2.2) that occurs during this time period to represent the entire time period. If $M = 2$, for example, then all of the words found in both of the closed caption sets shown in Figure 4.1 would be combined into a single segment. The visual features representing this segment would be obtained in some manner from the range of frames that begins with the display of the 1573^{rd} closed caption set (frame 169,689) and ends when the 1574^{th} closed caption set stops being displayed (frame 169,743). In this work we would represent this entire range of video frames by a single frame in this range.

Another potential problem with creating segments from individual closed caption sets or groups of closed captions in which no groups overlap is that possibly

important visual features can be missed. Since closed captions are typically not displayed constantly, there is a gap between closed caption sets that is lost. For example, if one set of closed captions is displayed from frame 100 until frame 200 and the next set of closed captions is displayed from frame 251 to frame 400, then the 50 frames that occur between these closed caption sets will not be represented under these conditions. This potential problem can be avoided by having the groups of closed caption sets overlap.

In order to address the second issue–capturing the temporal relationship of features–we chose to use hidden Markov models. Two HMMs can be constructed, one for 'liked' movies and the other for 'disliked' movies. To construct the 'liked' HMM, sequences from each of the movies the user has rated as 'liked' would be provided while the 'disliked' HMM would be constructed using sequences from the movies the user has rated as 'disliked'.

A simple approach for generating the sequences of observation symbols for each movie would be to concatenate the text and visual feature vectors from each closed caption set time period into a single vector and provide the sequence of these vectors as they occur chronologically to the HMM. There is a major problem with this approach, however. Because different combinations of closed caption sets and visual features are unlikely to have the same term feature representation, each combination would essentially represent a different observation symbol which would result in nearly every observation symbol being unique. Another consequence is that it is unlikely that the observation symbols produced from test samples will match any of the existing observation symbols and therefore HMMs are not suitable for classifying the test sequences. Our goal is to identify similar movies this viewer would like, not recognize the specific movies they have rated.

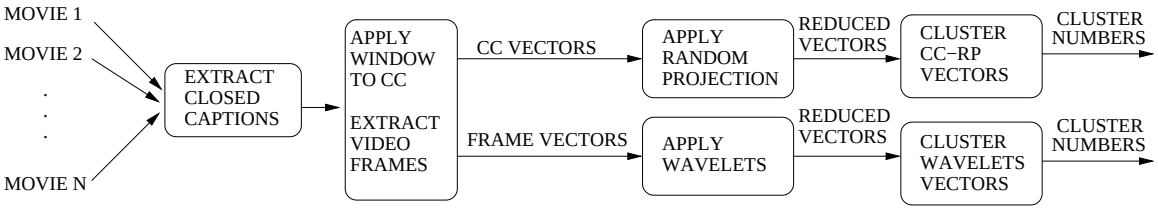

Figure 4.2. Overview of the data processing phase during training.

Our solution to this problem is to use clustering so that similar feature vectors will be represented by the same observation symbol. With this approach, similar movies should have similar sequences of observation symbols. In order to combine the visual features and closed captions, clustering (e.g., k-means, hierarchical, and so forth) is applied to the feature vectors for the closed captions for all of the movies a viewer has rated. This process is repeated for the feature vectors representing the visual features of these same movies. Automated methods (Section 2.4.4) determine how many clusters should be produced if using a method such as k-means or where to prune each hierarchy in the case of hierarchical clustering. The hypothesis is that vectors from movies the user liked and disliked will tend be in different groups. An overview of the data processing phase during training is shown in Figure 4.2.

Observation symbols are generated by combining the cluster number of a closed caption set and the cluster number of its corresponding video frame (i.e., the video frame chosen for the time period that the closed caption set was displayed) in the form (`closed caption set cluster number`, `video frame cluster number`). This process of generating observation symbols is repeated for each pair of closed caption set and video frame to produce the sequence of observation symbols for a specific movie, with a separate sequence being generated for each movie. Individually, the closed captions and the corresponding visual features that occur during a video may have

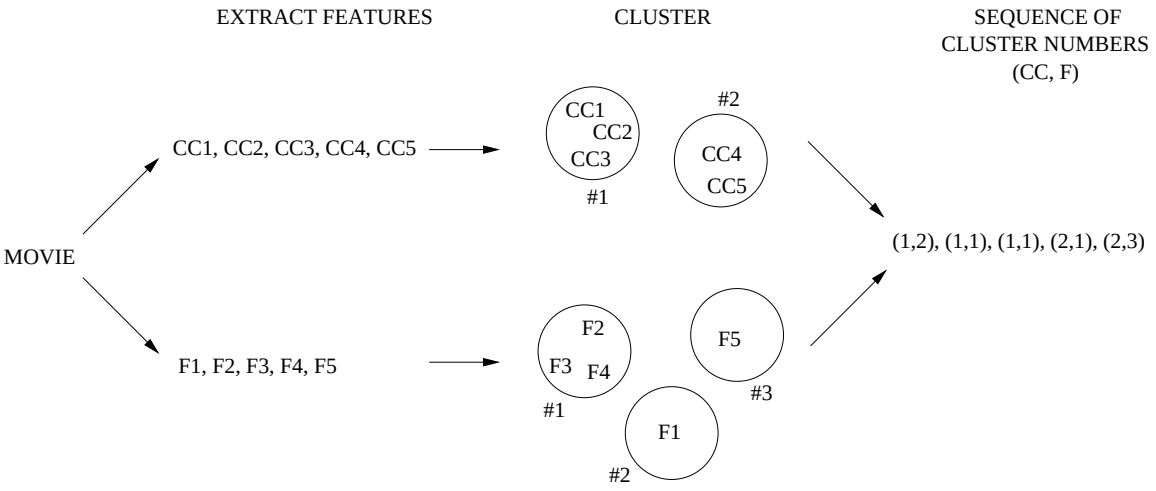

Figure 4.3. Example of observation symbol production.

multiple possible meanings. By generating observation symbols that combine these types of features, we believe that some element of context is captured.

An example is shown in Figure 4.3. In this example, k-means clustering is used to produce two clusters for the closed caption data and three clusters for the data derived from visual features. Closed captions set #1 (the one that occurred first in the movie) is in cluster 1 while its corresponding frame is in cluster 2, so the observation symbol will be (1,2). Closed captions set #2 (the second to occur in the movie) is in cluster 1 while its corresponding frame is cluster 1, so it is represented by the observation symbol (1,1). This is repeated for all of the closed caption sets and frames extracted from a video clip to generate a sequence of observation symbols that represent a video.

All of the sequences of observation symbols for the training videos rated as 'liked' by the viewer are used to construct an HMM. A second HMM is constructed using the sequences of observation symbols for the training videos rated as 'disliked'

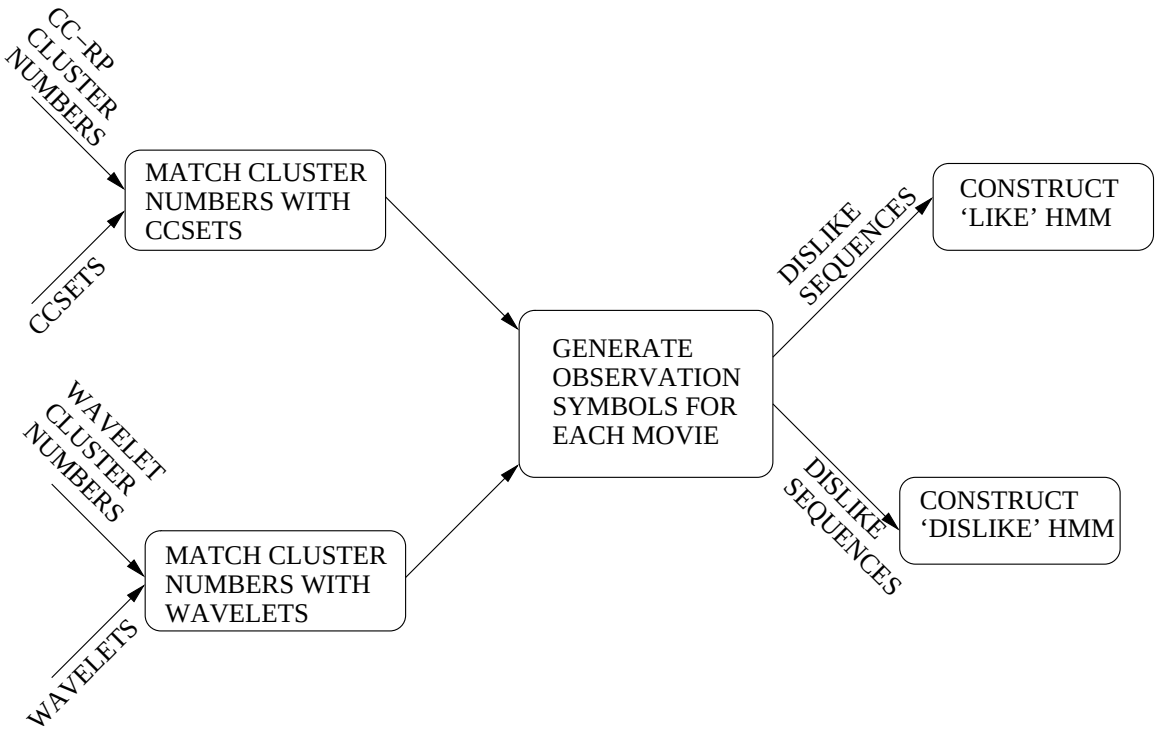

Figure 4.4. Overview of the training process from clustering to HMM construction.

by the viewer. Figure 4.4 shows the training process from the clustering to the construction of the HMMs.

To classify a new video, the text and visual features are extracted and represented as term-feature vectors as described for training videos. Instead of clustering the vectors, they are assigned to one of the clusters produced during the training phase to determine their cluster numbers. This is followed by generating a sequence of observation symbols as described earlier for training videos. This sequence is supplied to both HMMs and the video is assigned the classification of the HMM that generates the sequence with the highest probability. An overview of the testing process is shown in Figure 4.5.

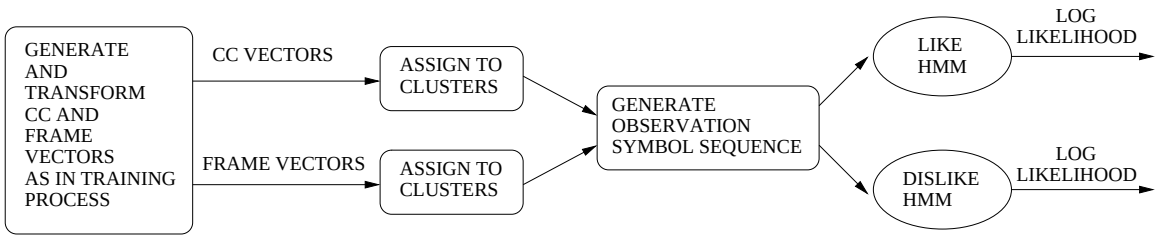

Figure 4.5. Overview of testing process.

The states of the HMM represent the high-level concepts that occur in the movies that the user has rated. In theory, these concepts could be things like "car chases" or "two people talking". However, it is difficult to look at the extracted features to discern the actual concepts being represented, especially in light of the fact that the constructed HMMs are not unique. Also, different viewers will have different preferences and therefore different models will be constructed.

This approach does not require either us or the viewer to enumerate the concepts found in the movies that the viewer prefers. The entire model building process is intended to find the necessary relationships between the features of rated movies and what the viewer prefers.

CHAPTER 5

EXPERIMENTS

5.1 Data Sets

In order to validate our approach, we chose to obtain real-world data. Video from the entertainment domain was chosen due to the fact that it has the closed captions required by our approach and the availability of data sets of user ratings. The user ratings for the experiments described in this document were obtained from the following two publicly available data sets: One Million Ratings MovieLens Dataset and Netflix Prize Dataset.

The MovieLens data set is the result of an online collaborative filtering movie recommender project [Gro05]. The data set was produced by 6,040 viewers who had rated movies from a set of 3,883 possible movies for a total of more than 1 million ratings. The range of rating values is 1–5, with one representing a strongly disliked movie while five represents a movie the user strongly liked. Users provided demographic information including gender, age (grouped into 7 age ranges), occupation (consisting of 21 choices), and zip code. The movie data includes the title and one or more genre labels.

The Netflix Prize data set was released by the DVD rental company Netflix as part of a contest to determine if participants could produce movie recommendations that surpassed Netflix's own recommender by at least 10% [Net07]. The data set consists of over 100 million ratings from 480,189 users from a set of 17,770 movies. The range of rating values is 1–5.

We acquired the DVD version of 90 of the movies represented in the MovieLens data set; 88 of these movies are also in the Netflix data set. These movies were selected from 18 entertainment genres (e.g., sci-fi, drama, and so forth) with many having multiple genre labels. The list of movies can be found in Appendix A.

5.2 Preliminary Experiments

When we began this work, we wished to determine the viability of using closed captions and visual features for learning preferences. To do so, we conducted experiments on the related task of classifying video by genre, for which there is already a large body of research [BCar]. We hypothesized that this would be an easier task than learning preferences. If we were unable to produce positive results using these features for classifying movies by genre, then it was unlikely that these features would be useful for learning video preferences.

For these initial experiments, closed captions and visual features were used separately for representing video and for each we performed three types of experiments: classification by genre, classification by user rating (i.e., attempt to learn the specific rating a user would assign to a video), and classification by grouped user ratings (i.e., the individual ratings are grouped to form 'Like' and 'Dislike' groups).

All tests were performed using the support vector machine classifier available in the Weka data mining software [WF00] with the default linear kernel. Support vector machines are well-suited to classification problems in which there are few training examples but the feature vectors have many terms [BC00]. We chose 81 movies represented in the MovieLens project that had been rated by at least 20 users. There were 1,116 users who had rated at least 10 of these 81 movies. For each type of experiment the mean classification accuracy was calculated.

To classify by genre, we created a separate test file for each genre with each movie being marked as either being in that genre or not. To classify by user rating, we created a test file for each of the 1,116 users with the movies that user had rated. The class for each movie was the rating that user had given the movie on a 1–5 scale. To classify by grouped user ratings, we created a test file for each of the 1,116 users with the movies that user had rated. The ratings were grouped: a movie with a rating of 4 or 5 was labeled as 'liked' while a movie with a rating of 1–3 was labeled as 'disliked'.

5.2.1 Preliminary Experiments Using Closed Captions

The closed captions were extracted from the DVDs in their entirety including any sound effects (e.g., [DOOR CREAKS]). The words found in sound effects could possibly be used to gain understanding of what is happening at that point in a video, but we did not pursue this in these experiments. Because the processing and storage requirements for text is significantly smaller than for visual features, we were able to use the entire set of closed captions from each movie. Each movie's closed captions were converted to a feature vector using the bag-of-words model (Section 2.1) after applying a standard stop list and Porter's stemming algorithm [Por80].

When classifying by genre using closed captions, feature vectors for all 81 movies were used. These feature vectors each had 15,254 terms. When classifying using the individual ratings each user had assigned to the movies, the feature vectors ranged in size from 4,401 to 13,350 terms depending on the movies rated. The results from these experiments are summarized in Table 5.1. These results are discussed in Section 5.2.3 along with the results from the experiments in which movies were represented using visual features.

Table 5.1. Summary of preliminary results using Closed Captions.

Experiment	Classification Accuracy	95% Confidence Interval
CC, by Genre	89.71%	(84.34, 95.09)
CC, Individual Ratings	38.45%	(37.40, 39.50)
CC, Grouped Ratings	64.04%	(63.02, 65.05)

5.2.2 Preliminary Experiments using DCT Coefficients

Many representations of video using visual features have been developed over the years, some of which are described in Sections 2.2 and 3.2. Our hypothesis was that movies with similar types of shots should have similar feature vectors and therefore we chose to represent each movie as a collection of shots with each shot being represented by video features found within the shot.

Processing visual features requires significantly more computational and storage requirements than closed captions, so we chose to extract features from the first five minutes of each video. To extract the DCT coefficients, we modified `mpeg_java`, an MPEG-1 video player whose source code is available [And05]. Since this software only supports MPEG-1 encoded video, we had to convert each DVD to an MPEG-1 clip. The resolution of the frames in our video was 240×352.

A color histogram was generated for each I-frame (Section 2.2.2). Shots were detected by comparing the color histograms of consecutive I-frames; if the differences between two of these frames exceeded some threshold, we assumed a shot change had occurred [AY99].

The DCT coefficients were extracted from the first frame of each shot with the assumption that the first frame is representative of the entire shot. In many cases the frames within a single shot will be similar enough for this assumption to hold true. If two consecutive frames within a single shot are significantly different, then it is likely

that the shot detection method will falsely identify a shot at this point anyway and the DCT coefficients for this frame will be included in the collection of shots.

The next step was to combine the DCT coefficients of the frame in order to represent the frame in some manner. One way in which this could be accomplished would be to simply concatenate all of the DCT coefficients in the order in which they occur. This approach makes it difficult to find similar frames since a common method for representing this type of information is as a vector with comparisons of vectors typically performed on a term-by-term basis. For example, many filmed scenes involve the camera rotating left or right. Any two consecutive frames filmed during this process are likely to appear to be dissimilar when compared on a term-by-term basis. While it still might be possible to find frames that are similar in content by some type of sliding window approach, such methods are very computationally expensive. An alternative approach, which we chose to use, is to represent the frames as histograms of the DCT coefficients. A term-by-term comparison of histograms determines that two frames are similar if they have similar color distributions, even if the exact locations of the colors in each frame differ as in the example of the last paragraph.

After deciding to represent a frame with a histogram of its DCT coefficients, we still wished to reduce the overall amount of information utilized and therefore we chose to use only the DC term from each block. To see how much information is contained just in the DC term of each block, see Figures 5.1 and 5.2. Figure 5.1 shows a frame from the TV show *Sliders* that was reconstructed from DCT coefficients. Figure 5.2 represents the same frame, but the 63 AC terms were set to zero before applying the inverse DCT. Although Figure 5.2 is blocky, it is still possible to recognize it as representing the frame shown in Figure 5.1. The histograms for each of the three color components were concatenated to form a vector representing the shot in a

Figure 5.1. Frame from *Sliders*.

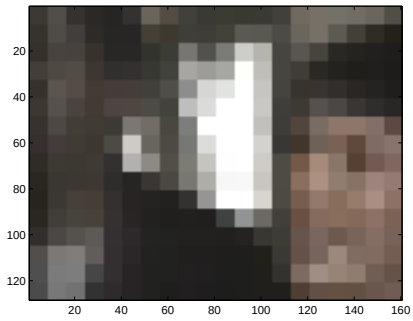

Figure 5.2. Frame from *Sliders* in which all values in a block use the DC term.

manner similar to that described in [WDV+03]. The resulting vector had $3 \times 2041 = 6123$ terms since DC coefficients can range in value from 0 to 2040 [MPE91b].

Once all of the shots had been represented as a histogram of DC terms, these histograms were clustered in order to group similar shots together. We chose k-means clustering (Section 2.4.3) with the Euclidean distance as the similarity measure. k-means is well known and generally applicable, which made it suitable as an initial choice. After the clustering was complete, each movie was represented by a feature vector with a term for each of the k clusters. We varied the number of clusters of shots to use for representing a video, first using $k = 20$ and then $k = 40$. For example, let us say that $k = 5$ and the vector of shots for a particular movie is [1, 0, 5, 50, 0]. This means that this movie contains a total of 56 shots, one of which was in cluster 1, 5 of which are in cluster 3, and 50 of which are in cluster 4. We don't know what high-level concept might be represented by each cluster; we just know that this movie contains this distribution of these concepts.

During the extraction of the DCT coefficients, the software failed prior to reaching the end of each movie due to a lack of robustness for dealing with differences in MPEG video encoding. This prevented us from using the entire five minutes of video

Table 5.2. Summary of preliminary results using DCT Coefficients.

Experiment	Classification Accuracy	95% Confidence Interval
DC Terms, by Genre (20)	88.48%	(82.66, 94.30)
DC Terms, Individual Ratings (20)	33.26%	(32.33, 34.19)
DC Terms, Grouped Ratings (20)	59.23%	(58.28, 60.19)
DC Terms, by Genre (40)	87.24%	(81.17, 93.31)
DC Terms, Individual Ratings (40)	32.54%	(31.63, 33.45)
DC Terms, Grouped Ratings (40)	58.76%	(57.83, 59.69)

from some of the movies and resulted in an inconsistent number of minutes processed for each movie. While the total number of shots for all 81 movies was 46,311, we were only able to obtain a few shots for some movies while for others we obtained hundreds. The results for the experiments that used DCT coefficients are summarized in Table 5.2 with 95% confidence intervals [BC06].

5.2.3 Results and Discussion

Comparing Tables 5.1 and 5.2, we can see that the results were virtually the same regardless of whether closed captions or DCT coefficients were used. In each case classification by genre had the best results while classification by individual ratings had the worst. We expected classification by genre of a movie to be easier than learning an individual's preferences and so were not surprised by these results. We were surprised to find that when using DCT coefficients as the feature the results were very similar regardless of the number of clusters. The previously mentioned problem in obtaining consistent data may have contributed to this. Another possible reason was that the threshold value that we used for shot detection may have been too conservative, that is, the amount of difference between two frames necessary to

indicate that a shot had occurred may have been too high. This would result in some shots being undetected.

The results when learning preferences using individual ratings with the features being visual were 32.5% (for 40 clusters) and 33.3% (for 20 clusters); when the features were closed captions the results were 38.4%. These values are better than the 20% accuracy one would expect to get if the ratings were chosen at random from a 1–5 scale, but there is still much room for improvement. It seems unlikely that users would be satisfied with a recommender system with classification accuracies this low. One reason for this poor performance could be that the number of training examples for each user was too small to learn a user's rating preferences.

5.3 User Modeling with Hidden Markov Models and Hierarchical Clustering

In our preliminary experiments, we were able to classify video by genre with results that were much better than would be expected by choosing genre at random [BC06]. This suggests that text and visual features are viable for learning preferences. However, the results were much less favorable when we used each of these types of features for predicting that a viewer would like or dislike a movie, or in predicting the specific viewer rating of a movie.

To address the limitations of the approach taken in our preliminary experiments, we wished to combine the text and visual features as well as represent the temporal relationship of the features. The approach that we chose to accomplish this can be described as follows: extract the closed captions and visual features and cluster each separately, generate observation symbols for a hidden Markov model by combining the cluster assignments of the features, and construct a hidden Markov model for each of the two classes (i.e., Like, Dislike) that we are interested in predicting.

For these experiments, we again used the viewer ratings from the MovieLens data set. In this case the data set consisted of 357 viewers who had rated at least 20 of these movies, for a total of 9,708 ratings. The number of movies rated by each viewer ranged from 20 to 69 with a mean of 27. For each viewer we split the ratings into two groups: movies with ratings of 4 or 5 were considered 'liked' while the remainder were considered 'disliked'. For the entire data set, there were 4,771 (49%) disliked movies and 4,937 (51%) liked movies although this is not guaranteed for any particular viewer. Two-thirds of the liked and disliked sets were used for training.

It was not computationally feasible for us to extract and work with features from the full length of each movie, therefore we chose to extract features from only a five minute portion of each movie. In particular, we extracted features from minutes 5 to 10 of each movie. This differs from our preliminary experiments that attempted to extract visual features from the first five minutes of each video. The reason for using minutes 5 to 10 as opposed to the first five minutes of each movie is that the very beginning of a movie is often used for displaying credits and therefore may not be representative of the movie as a whole. The visual features might differ as well as there are fewer closed captions. Limiting our method to just a five minute period does have the potential drawback that this time period may not capture what is important to a user. For example, a viewer may prefer movies in which story and character development is drawn out over a long period. Another viewer may enjoy action scenes, which typically don't occur at the very beginning of movies.

In order to capture any temporal relationship existing in the features, we first needed to segment the video and then extract the text and visual features present in each segment. A common way to do this would be to segment the video into shots (Section 2.2). We found this very difficult to accomplish in an automated fashion. Since the closed captions that we were working with were extracted from DVDs

of movies, we had timestamps available to us for when the closed captions would be displayed. Therefore we decided to use the time that these closed caption sets (Section 2.1) were displayed as the segmentation mechanism as described in Chapter 4 with $M = 1$ and $N = 1$, that is, a segment consisted of one closed caption set and a new segment began every one closed caption set.

For the five minutes of each video that we were using, we extracted the closed caption sets. Each of these closed caption sets was represented using the bag-of-words model after applying a stop list and Porter's stemming algorithm. The number of closed caption sets for this five minute period ranged from 32–162, with a mean value of 86. The term-feature vectors representing the closed captions had 4,003 terms. Random projection (Section 2.3.2) was applied to reduce the dimensions of the vectors from 4,003 terms to 400 terms. The transformation matrix R was generated using the method of Fern and Brodley [FB03]. Random projection was chosen due to the success demonstrated by Bingham and Mannila [BM01] in applying random projection to sparse matrices representing documents. We found the need to reduce the dimensionality of the closed caption vectors at a later stage of the process that involved clustering the vectors. Clustering the original 4,003 term features vectors of closed captions would take approximately 24 days in Matlab running on a computer with a 2.8 GHz Pentium D processor and 1 Gb of memory. After applying random projections, the clustering process took approximately two days.

In the preliminary experiments, the visual features were derived from the DCT coefficients produced during the MPEG encoding process. Since these DCT coefficients only represent local features, we decided to instead use wavelet-transformed RGB values since they capture global features [Gha03]. We also changed our method for extracting visual features which allowed us to overcome the technical problems from which our preliminary experiments suffered.

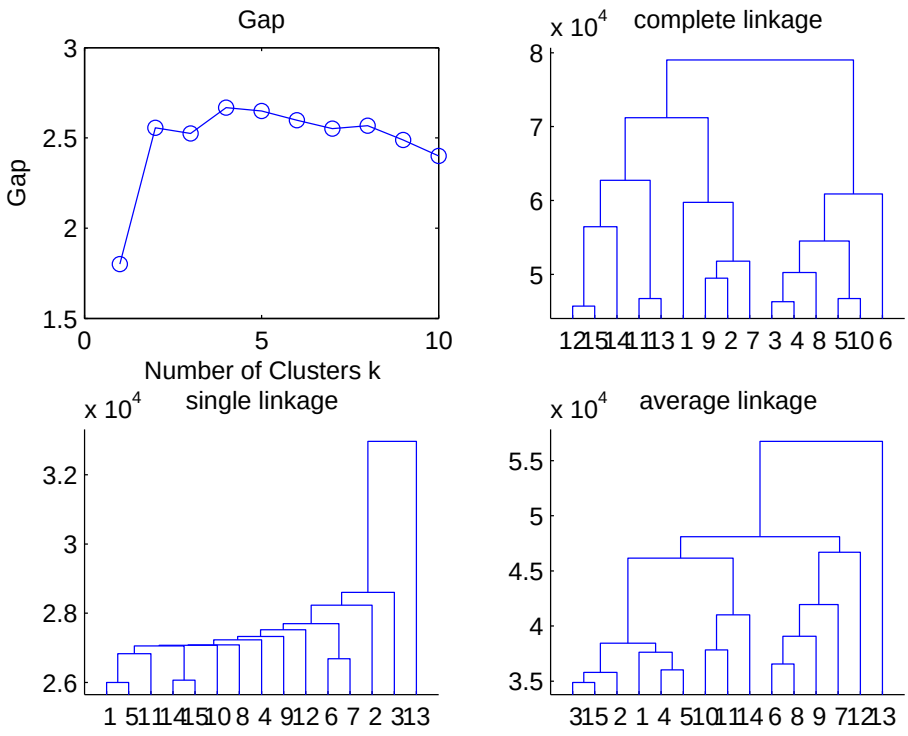

Figure 5.3. Hierarchies constructed for the wavelet-transformed features for a typical viewer.

To produce the visual features, the first frame from each of the time periods that the closed caption sets were displayed was extracted. Representing each frame by concatenating the RGB values of the pixels would have produced vectors with 253,440 terms. To reduce the dimensionality, five levels of a 2D Daubechies 4 wavelet [Wal99] were applied separately to the R, G, and B components. The final trend values (Section 2.3.1) were concatenated to form vectors of 363 terms.

The visual features and closed captions were clustered separately using agglomerative hierarchical clustering. We performed exploratory data analysis (Section 2.4.4) on a subset of our training data to determine which linkage method (Section 2.4.2) and distance measure produced the best hierarchy, with 'best' in this case

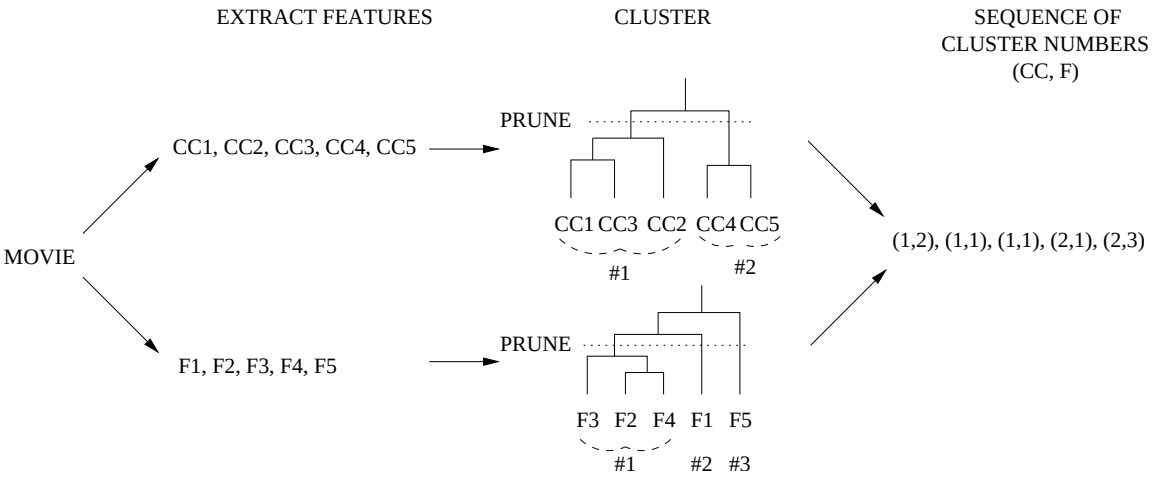

Figure 5.4. Process for combining closed captions and visual features to form observation symbols.

being a combination of balance and height between groups. We found that in general complete linkage with the Euclidean distance produced balanced hierarchies for the wavelet-transformed pixel values. An example can be seen in Figure 5.3. For the closed captions transformed by random projections, none of the combinations of linkage methods and distance measures produced well-balanced hierarchies which suggests that hierarchical clustering may not be the most appropriate form of clustering for closed captions. For both types of features, we performed clustering using complete linkage with the Euclidean distance.

We chose to partition the hierarchy into clusters using the method proposed by Krzanowski and Lai (Section 2.4.4). Calculating the within-group-sum-of-squares is very computationally intensive and therefore we limited our search for the correct number of clusters to cluster sizes of 2–10. This is also what prompted the previously mentioned decision to use random projection to reduce the dimensionality of the vectors of closed caption terms.

Once the clustering was complete, the cluster numbers for the closed caption sets and the corresponding visual features were combined to form the observation symbols for training the HMMs. To create an observation symbol, the cluster number that a closed caption set is assigned to is combined with the cluster number of the corresponding visual feature. An example can be seen in Figure 5.4. In this example, the pruning of the hierarchy of closed captions produces two clusters, numbered from left to right as #1 and #2. The pruning of the hierarchy of video frames produces three clusters, numbered from left to right as #1, #2, and #3. We can see that `CC1`, the first closed caption set to occur in the video, is in cluster #1. The video frame that was displayed at the time this closed caption set was displayed is `F1` and it is in cluster #2. Therefore, the observation symbol produced from this combination is `(1, 2)`. This process is repeated for all of the closed caption and visual features in a video.

After the sequences of observation symbols were generated for the training samples, two HMMs were constructed: one from the training samples of the movies the viewer rated as 'liked' and one from the movies the viewer rated as 'disliked'. The 'liked' and 'disliked' HMMs each had two states; we made an arbitrary decision to set the start probabilities equal for both states. The initial elements of the transition matrix were all set to have equal likelihood as were the elements of the observation matrix.

It is necessary when constructing models to make a decision as to how many states to use. A choice of too few will result in models that don't accurately represent the underlying relationships being modeled while too many will overfit the data (Section 2.5). In constructing the 'liked' and 'disliked' HMMs, we investigated models with 2–10 states with the same number of states in each model. The optimal number of states will be that which maximizes the performance.

Table 5.3. Comparison of features in initial hybrid approach

Features	Accuracy	95% CI
CC + Visual	54.6%	(52.7, 56.6)
CC only	51.4%	(49.4, 53.5)
Visual only	52.7%	(50.7, 54.7)

To test the constructed models, we needed to generate observation symbols for the test samples. This was accomplished by finding the training cluster that the test sample would have been assigned to by the complete linkage method using the Euclidean distance as the distance measure. The test samples produced observation symbols that were not present in the training samples, which made it impossible to calculate the log-likelihood of the test sequences. To overcome this, the emission probabilities with values of zero were changed to an arbitrarily low value of 10^{-6}.

5.3.1 Results and Discussion

Our results from combining closed captions and visual features are shown in Table 5.3 [BC07]. We found that the accuracy, precision, and recall were essentially the same regardless of the number of states; the reported results are for models with two states. We also generated observation symbols from each type of feature alone in order to determine if the combination of features was an improvement. Our combination approach had an average classification accuracy over the 357 viewers of 54.6%, which is only slightly better than what would be expected if the movies were picked at random. The average classification accuracy when using closed captions or visual features alone was 51.4% and 52.7%, respectively. While the mean value of our combination approach is larger than the mean values of using either type of feature separately, the confidence intervals overlap and we therefore can't state that the results are significantly different.

Table 5.4. Results per number of movies rated in initial hybrid approach

Number Rated	# users	mean	95% CI
$20 \leq$ movies rated < 30	253	54.2	(51.8, 56.5)
$30 \leq$ movies rated < 40	78	54.0	(50.1, 57.8)
$40 \leq$ movies rated ≤ 69	26	61.5	(56.3, 66.6)

We believe there are several possible reasons for the poor performance of this approach. The first is that out of the 357 users for which we had preference information, 253 of them had rated less than thirty movies (Table 5.4). In fact, forty-eight had only rated twenty movies. This is unlikely to be a sufficient amount of data for this approach to effectively learn preferences. When we look at the average results for the viewers who had rated forty or more movies, the mean classification accuracy improves to 61.5%. However, the confidence intervals overlap for each of the three ranges of number of ratings that we looked at and therefore we can't state that the differences are statistically significant.

Another possible problem is that when generating the test observation symbols by combining the cluster numbers for the visual features and closed captions, it's possible to generate observation symbols that never occurred in the training data. Many test sequences contained symbols that never occurred in the training, so even though we gave these symbols an emission probability of 10^{-6} to make it possible to calculate the log-likelihood for the sequence, each of these symbols essentially contributed nothing to that calculation. The remaining symbols may not have been enough to effectively learn preferences. A possible solution to this that we can investigate in a future work is to assign observation symbols to test samples by finding the nearest training combination of closed caption and visual features. This will avoid generating observation symbols that never occurred in the training data. It is not possible to

generate unseen observation symbols for the test samples when using only a single type of feature, so this can't account for the poor performance of using either type of feature alone.

In our preliminary experiments [BC06], discussed in Section 5.2, we investigated the use of closed captions and DCT coefficients separately. We felt that this work suffered from two problems. First, it did not attempt to combine the text and visual features with the resultant gain in performance that one would expect from such a combination. Second, no consideration was given to the order in which features appear. However, the preliminary experiments do help us to establish a baseline for comparison with the current method in order to determine whether the use of HMMs has improved performance.

The use of closed captions alone in the preliminary experiments had a classification accuracy of 64.04% with a 95% confidence interval of (63.02, 65.05), as seen in Table 5.1. This exceeds the classification accuracy of 51.4% that we achieved when using closed captions alone in our initial approach using HMMs. In our preliminary experiments, all of the closed captions for an entire movie were represented in a single feature vector. In these initial experiments using HMMs, prior to dimensionality reduction each closed caption set is represented by a feature vector with 4,003 terms for the 4,003 unique words present in the entire data set. However, the maximum number of words that any closed caption set had was eleven which means that more than 99% of the terms were zero. The unlikely overlap of many of the feature vectors makes learning difficult. Using closed caption sets as the segmentation mechanism may not be appropriate since it appears to be over-segmenting the video, both when closed captions are used alone or combined with visual features. A better approach may be to segment by combining several closed caption sets, such as considering every ten sets of closed captions a segment.

The visual features in the preliminary experiments were the DC terms of the discrete cosine transform coefficients and the classification accuracy using these features alone was 59.23% with a 95% confidence interval of (58.28, 60.19); these are found in Table 5.2. Because both the visual features and methodology were different in the preliminary experiments, we could not directly compare our current approach when using visual features alone to our preliminary experiments. Therefore, we wished to know how well the preliminary experiments would have performed if wavelet-transformed RGB values had been used instead of DCT coefficients.

As stated previously, technical problems prevented us from extracting visual features from the entire five minute period for some of the videos. They also make it impossible for us to completely replicate the earlier results. Therefore, instead of extracting the DCT coefficients from the MPEG-encoded video, we simulated the MPEG encoding process by applying the discrete cosine transform to blocks of the bitmaps of the individual video frames from the five minute period for each video.

To identify the video shots, we first computed the color histogram for each video frame. The color histograms for consecutive frames were compared by taking the difference between each. Because the magnitude of the differences varied so much for different movies, the differences of these differences were then calculated to identify significant differences. An analysis of a sample of our movies indicated that those 2^{nd} order differences that exceeded a threshold of 30,000 indicated a shot change in most cases, which allowed us to automate the shot detection process.

After identifying the video shots, either DCT coefficients or wavelet coefficients were used to represent each shot as described in Section 5.2. The classification results, shown in Table 5.5, indicate that in the preliminary experiments wavelet coefficients would have performed as well as DCT coefficients. Therefore, the decreased perfor-

Table 5.5. Comparison of DCT and Wavelet coefficients in preliminary results.

Feature Type	Accuracy	95% C.I.
DCT coefficients	60.6%	(59.2, 62.0)
Wavelets	60.4%	(59.0, 61.7)

mance of these current experiments when using only visual features appears to be due to problems with the methodology.

5.4 User Modeling with Hidden Markov Models and k-Means Clustering

The results from our initial attempt to learn video preferences by combining text and visual features were only slightly better than what would be expected if the movies were assigned to the Like and Dislike classes at random. We decided that the most likely sources of the poor performance were insufficient number of training examples for most users, over-segmentation of the video that resulted in closed caption feature vectors that were very sparse, and observation symbols from test samples that were unseen in the training data (i.e., zero probability).

To address the perceived problems with our initial efforts to combine features, we made the following changes to our methodology [BC07]. We increased the average number of movies rated per user by switching to the Netflix data set. Table 5.6 shows the number of ratings for our set of DVDs for the MovieLens and Netflix data sets. By using the ratings from the Netflix data set, we can restrict our tests to those users who had rated at least 45 movies. In our earlier work using the MovieLens data set, there were 357 users who matched our criteria. There are 334 users in the Netflix data set that we can use.

Instead of using a single closed caption set as a segmentation mechanism, we created a window of consecutive closed caption sets with a length of twenty, with a

Table 5.6. Comparison of the number of ratings per user for the GroupLens and Netflix data sets.

# ratings	# MovieLens users	# Netflix users
$20 \leq ratings < 30$	253	10,967
$30 \leq ratings < 40$	78	2,934
$40 \leq ratings < 50$	20	631
$50 \leq ratings < 60$	5	142
$60 \leq ratings < 70$	1	30
$70 \leq ratings < 80$	0	9
$80 \leq ratings < 90$	0	5

new window beginning every tenth closed caption set (i.e., $M = 20$ and $N = 10$) as described in Chapter 4. That is, we combine closed caption sets 1–20, 10–30, 20–40, and so forth. All of the words from all closed caption sets within the window are combined to represent that entire time period that these closed captions are displayed. Visual features were derived from a single frame within this period to represent the entire time frame. The frame chosen is the first (available) frame from the time period of the 10^{th} closed caption set in each window of 20.

By combining the closed caption sets within a window into a single feature vector, we were able to reduce the number of feature vectors per minute of video. This allowed us to increase the total number of minutes of video processed from five to twenty, with the specific twenty minute segment of each video being minutes five to twenty-five. Random projection was used to reduce the dimensionality of the feature vectors produced from the closed captions from 4,003 terms to 363 terms. The visual features were wavelet-transformed RGB values as described in Section 5.3. Hidden Markov models were constructed as described in Section 5.3 except that the initial values used in the starting, transition, and observation matrices were chosen randomly. With two of the perceived problems with the earlier hidden Markov approach

addressed, the results were similar to those described in Section 5.3 (summarized in Table 5.3).

We still hypothesized that the performance was being influenced by observation symbols generated for test sequences that were unseen when constructing the models. To generate observation symbols for the test sequences, we matched closed caption set/video frame pairs in the test data to closed caption set/video frame pairs in the training data instead of finding similar closed caption sets and video frames individually. To do this, we began by concatenating the closed caption and visual feature vectors from the training samples and did the same to the test samples. The original vectors of the closed captions and visual features had substantially different ranges of values, so we normalized each column of these row vectors using [Pyl99]

$$\nu_{norm} = \frac{\nu_i - min(\nu_1 \ldots \nu_n)}{max(\nu_1 \ldots \nu_n) - min(\nu_1 \ldots \nu_n)}$$

for both the training and test samples where v_i is the specific term value and $v_1 \ldots v_n$ is the range of values in that column. Also, the decision to reduce the dimensionality of the closed caption vectors to 363 terms was so that the length would match the length of the vectors of wavelet coefficients; the concern was that unequal vector lengths would give more influence to one type of feature. After normalizing the feature vectors, the test samples were assigned the observation symbol from the nearest training sample, with nearest being determined using the Euclidean distance. In spite of these various changes to the approach described in Section 5.3, the classification results remained essentially the same.

During our investigation of these results to determine the cause of the poor performance, we noticed that the hierarchical clustering produced groups in which a single group would contain the majority of the clustered objects. The consequence of this is that many of the observation symbols will be the same and therefore it will

Table 5.7. Comparison of features from HMM and k-means clustering

Features	Accuracy	95% CI	Prec.	95% CI	Recall	95% CI
CC + Visual	61.7%	(60.2, 63.2)	51.2%	(48.0, 54.4)	53.4%	(49.1, 57.6)
CC only	60.9%	(59.4, 62.4)	49.0%	(46.0, 52.0)	50.8%	(46.9, 54.7)
Visual only	61.5%	(60.1, 63.0)	50.9%	(47.8, 54.0)	50.1%	(46.3, 54.0)

be difficult to distinguish between the sequences of 'liked' and 'disliked' movies. This prompted us to investigate another clustering method, and for this we chose to use k-means clustering (Section 2.4.3).

We applied k-means clustering to the feature vectors that were processed as described earlier in this section. The value of k to use for each type of feature for each user was determined using the method of Krzanowski and Lai, with the value of k limited to 2–10 to keep the computational requirements reasonable. When we looked at the cluster assignments produced by k-means clustering, we found that the objects were much better distributed than what had been produced using hierarchical clustering. When producing observation symbols for the test samples, we did not restrict the values to those that had occurred when generating the observation symbols for the training examples.

5.4.1 Results and Discussion

We investigated HMMs with 10–70 states (in increments of 10) and found that from states 30–70 the average classification accuracies were all in the range of 61–62% with the highest classification accuracy occurring at 60 states. Table 5.7 shows the results that were achieved when k-means clustering was used with HMM models with 60 states. Results are reported in terms of accuracy, precision, and recall. Throughout this work our efforts have been focused on improving classification accuracy. However, the results of much of the other work in video recommendation is

reported in terms of precision and recall and therefore we report our results using those metrics as well. Accuracy is the ratio of the number of correct predictions to the total number of test samples. Precision is the ratio of the true positives to the sum of the true positives and false positives. Recall is the ratio of the true positives to the sum of the true positives and the false negatives. Precision and recall are more commonly found in information retrieval. In the case of a movie recommender system, the recommender might recommend a subset of the total movies available. Precision is a measure of how many of the recommended movies the user actually prefers. Recall is a measure of how many of the movies in the total data set that the user would prefer end up in the recommendations made to the user. When only a subset of movies is recommended to a viewer, it is possible to improve the precision without improving the overall accuracy of classification.

We can see that the results from combining features were approximately the same as were achieved when generating observation symbols for either type of feature alone. The correlation between the model constructed using only closed captions and the model constructed using only visual features was 41%. This suggests that both types of features contribute differently to the classification. The correlations between the results produced by the model constructed using a combination of closed captions and visual features and the models constructed using only closed captions or only visual features had mean values of 46% and 47%, respectively.

When we compare these results to the initial effort to combine features that used hierarchical clustering (Table 5.3), we can see that the switch to k-means clustering improved the results. An analysis of the models shows that for many users one of the models would perform very well while the other model would perform very poorly. In particular this seemed to be the case when the user's ratings were not close to

Table 5.8. Results per number of movies rated in hybrid approach using k-means

Number Rated	# Users	Accuracy	95% CI
$45 \leq$ movies rated < 50	167	60.0	(57.8, 62.3)
$50 \leq$ movies rated < 60	131	62.5	(60.3, 64.8)
$60 \leq$ movies rated < 70	26	64.6	(59.7, 69.5)
$70 \leq$ movies rated ≤ 88	10	72.3	(59.8, 84.8)

being evenly distributed between 'liked' and 'disliked'. This could be a consequence of having too few training examples to learn from for one of the classes.

Table 5.8 shows the classification accuracy by number of movies rated. We can see here that as would be expected, the classification accuracy improved as the number of ratings increased although the 95% confidence intervals overlap for each case and therefore we can't state that the differences are statistically significant. However, an analysis of the individual predictions shows that even for the users for which there were a large number of rated movies, in most cases only one of the HMMs performed well. Since this was the HMM constructed from the majority of the user's ratings, this resulted in a measured improvement in performance.

As stated previously, much of the other work in video recommendation reports results in terms of precision and recall. In video recommender systems that focus on improving precision, typically this means that the viewer is provided with a subset of the total number of movies that the viewer is predicted to like. It is possible then to improve the precision (since it is calculated from this subset) while the overall accuracy of the system may not improve. While the authors of the work that we describe here for comparison have not stated so explicitly, we assume that they maximized precision by providing the viewer with a subset of the video that they may potentially like. This should be kept in mind when comparing the results by other in this field with our own work.

Ardissono et al. [AGT+04] achieved a precision of 80% and a mean absolute error rate of 30% in their case-based approach to recommendation. Basu et al. [BHC98] achieved precision and recall values of 83% and 34%, respectively, in their system that combined the case-based and collaborative filtering approaches. Their approach focused on achieving high precision at the expense of the recall. By comparison, we achieved precision and recall values of 51.2% and 53.4%, respectively, over the entire set of movies for each viewer. The average precision and recall for our approach were 45% and 50%, respectively. Therefore, the precision that we achieved was statistically different than what we would expect by choosing movies at random, but the recall was not statistically different from random. Neither of the other approaches reports overall classification accuracy, so we can not compare our results to theirs using that metric. While both of their approaches achieve much higher precision than our approach, they are still restricted to those situations for which these two approaches can be applied as described in Chapter 1.

While our final approach was able to produce results that exceeded what would be expected if movies were classified as 'like' or 'dislike' at random, there is still much room for improvement. There are several possible areas that could be investigated in a future work. One is to increase the number of training examples for each user. Table 5.8 suggests that the results improve with the number of training examples, but the small number of users for which we had large numbers of training examples resulted in large confidence intervals. More viewers with large numbers of training examples would allow us to be more confident in the results.

It is possible that our choice of features, in particular the decision to rely on transformed color values, may not have sufficiently captured the characteristics that are important to viewers. As was discussed in Chapter 2, a large number of visual

features are available and one or more may better represent what is of interest to a viewer.

A third area that could be investigated further is the use of HMMs for modeling the viewer's preferences. It is possible that the training and test sets of data were insufficiently similar for the models to accurately classify the test sequences. This could also be due to the models overfitting the data. We also did not investigate potential bias in the models in this work. If there is bias in the model predictions, it may be possible to identify and correct for this.

CHAPTER 6

CONCLUSIONS

Traditional approaches to video recommendation have been shown to have relatively good performance. However, for reasons described previously, these approaches are not always applicable. To address this need and to provide an alternative, we have explored the use of visual features and closed captions extracted from video for learning a viewer's preferences. Our final approach achieved results that were better than what would be expected if the video was randomly assigned to either the 'like' or 'dislike' classes and therefore we believe this approach to be a viable alternative to traditional approaches to learning video preferences.

We took two types of features commonly found in entertainment video and combined them to learn models representing a user's likes and dislikes. We found that in many cases one of the learned models tended to not perform well. In an experiment conducted under ideal conditions, the movies that a user would watch and rate would be chosen at random. However, the user ratings that we used were not obtained under such conditions and for most viewers the number of liked and disliked movies were far from even. This likely resulted in an insufficient number of training examples for one of the classes.

While our experiments focused on entertainment video, we believe that this work can be extended in a number of ways. It can be applied to the task of classifying video by genre. While there has already been much research in this area, there is still room for improvement.

It could also be applied at the shot or scene level. Applications include content filtering, such as identifying violent scenes in movies, or the identification of scenes important to the user. Video summarization can also be performed by finding scenes important to many users. Of the research that has been performed in automatically classifying video by genre, very little has attempted to subdivide genre, such as finding action movies that include car chases or separating romantic comedies from dark comedies.

Much of the research in learning video preferences and classifying video by genre has focused primarily on the entertainment video domain. Other domains, such as education, should be explored more to determine how well our approach would apply to them.

Another area where our work could be applied is video learning. As more and more educational video becomes available, students will have a variety of video choices for learning a particular topic. If the student's performance on a test covering a specific topic is tracked, which should be possible with online courses, then video can be recommended that is similar to those that resulted in the best performance by the student.

APPENDIX A

MOVIE DATA

These movies were chosen from the MovieLens [Gro05] and Netflix [Net07] data sets as the sources of ratings information.

Table A.1: Movie Titles and Genre

Movie	**Genre**
13th Warrior, The (1999)	Action, Horror, Thriller
Adventures of Buckaroo Bonzai Across the 8th Dimension, The (1984)	Adventure, Comedy, Scifi
Affair of Love, An (Une Liaison Pornographique) (1999)	Drama, Romance
Apocalypse Now (1979)	Drama, War
Awfully Big Adventure, An (1995)	Drama
Babes in Toyland (1961) [1]	Childrens, Fantasy, Musical
Betrayed (1988)	Drama, Thriller
Black Beauty (1994)	Adventure, Childrens
Blue Lagoon, The (1980)	Adventure, Drama, Romance
Boogie Nights (1997)	Drama
Boys and Girls (2000)	Comedy, Romance
Boy Who Could Fly, The (1986)	Drama, Fantasy
Bull Durham (1988)	Comedy
Chicken Run (2000)	Animation, Childrens, Comedy
Child's Play (1988)	Horror
Chinatown (1974)	FilmNoir, Mystery, Thriller
Circle of Friends (1995)	Drama, Romance

[1] not in Netflix data set

Table A.1 Continued

Clean Slate (1994)	Comedy
Cool Runnings (1993)	Comedy
D3: The Mighty Ducks (1996)	Childrens, Comedy
Days of Thunder (1990)	Action, Romance
Dial M for Murder (1954)	Mystery, Thriller
Dolores Claiborne (1994)	Drama, Thriller
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)	Scifi, War
Even Cowgirls Get the Blues (1993)	Comedy, Romance
Following (1998)	Drama
French Kiss (1995)	Comedy, Romance
Gloria (1999)	Drama, Thriller
Glory (1989)	Action, Drama, War
Godzilla (1998)	Action, Scifi
Gone Fishin' (1997)	Comedy
Hard 8 (a.k.a. Sydney, a.k.a. Hard Eight) (1996)	Crime, Thriller
Haunted Honeymoon (1986)	Comedy
Hoosiers (1986)	Drama
House Party (1990)	Comedy
I Dreamed of Africa (2000)	Drama
It Takes Two (1995)	Comedy
Lawn Dogs (1997)	Drama
Lethal Weapon (1987)	Action, Comedy, Crime, Drama
Life and Times of Hank Greenberg, The (1998)	Documentary

Table A.1 Continued

Live Flesh (1997)	Drama
Mary Shelley's Frankenstein (1994)	Drama, Horror
Minus Man, The (1999)	Drama, Mystery
Mod Squad, The (1999)	Action, Crime
Moll Flanders (1996)	Drama
My Crazy Life (Mi vida loca) (1993)	Drama
My Son the Fanatic (1998)	Comedy, Drama, Romance
Naked Gun 33 1/3: The Final Insult (1994)	Comedy
Natural, The (1984)	Drama
Night Falls on Manhattan (1997)	Crime, Drama
Nobody's Fool (1994)	Drama
Papillon (1973)	Drama
Party Girl (1995)	Comedy
Pork Chop Hill (1959)	War
Radio Days (1987)	Comedy, Drama
Rambo: First Blood Part II (1985)	Action, War
Rapture, The (1991)	Drama, Mystery
Remember the Titans (2000)	Drama
Rock, The (1996)	Action, Adventure, Thriller
Roseanna's Grave (For Roseanna) (1997)	Comedy, Romance
Rounders (1998)	Crime, Drama
Santa Claus: The Movie (1985) [2]	Adventure, Childrens, Fantasy
Selena (1997)	Drama, Musical

[2]not in Netflix data set

Table A.1 Continued

Singin' in the Rain (1952)	Musical, Romance
Sliding Doors (1998)	Drama, Romance
Soldier (1998)	Action, Adventure, Scifi, Thriller, War
Space Cowboys (2000)	Action, Scifi
Sphere (1998)	Adventure, Scifi, Thriller
Spy Hard (1996)	Comedy
Stage Fright (1950)	Mystery, Thriller
Stargate (1994)	Action, Adventure, Scifi
Star Trek: First Contact (1996)	Action, Adventure, Scifi
Star Trek VI: The Undiscovered Country (1991)	Action, Adventure, Scifi
Stigmata (1999)	Thriller
Strange Days (1995)	Action, Crime, Scifi
Strangers on a Train (1951)	FilmNoir, Thriller
Strictly Ballroom (1992)	Comedy, Romance
Striking Distance (1993)	Action
Sweet Hereafter, The (1997)	Drama
Swimming with Sharks (1995)	Comedy, Drama
They Shoot Horses, Don't They? (1969)	Drama
Thin Blue Line, The (1988)	Documentary
Three Kings (1999)	Drama, War
Three to Tango (1999)	Comedy, Romance
Total Eclipse (1995)	Drama, Romance
True Grit (1969)	Adventure, Western

Table A.1 Continued

Untouchables, The (1987)	Action, Crime, Drama
Walk in the Clouds, A (1995)	Drama, Romance
What Dreams May Come (1998)	Drama, Romance
White Men Can't Jump (1992)	Comedy

REFERENCES

[AD99] Lalitha Agnihotri and Nevenka Dimitrova. Text detection for video analysis. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, 1999. (CBAIVL '99)*, pages 109–113, 1999.

[AECI00] Yousri Abdeljaoued, Touradj Ebrahimi, Charilaos Christopoulos, and Ignacio Mas Ivars. A new algorithm for shot boundary detection. In *Proceedings of the 10th European Signal Processing Conference*, pages 151–154, 2000.

[AGT+04] Liliana Ardissono, Cristina Gena, Pietro Torasso, Fabio Bellifemine, Angelo Difino, and Barbara Negro. User modeling and recommendation techniques for personalized electronic program guides. In Liliana Ardissono, Alfred Kobsa, and Mark Maybury, editors, *Personalized Digital Television: Targeting Programs to Individual Viewers*. Kluwer, 2004.

[AKK97] Joshua Alspector, Aleksander Kołcz, and Nachimuthu Karunanithi. Feature-based and clique-based user models for movie selection: A comparative study. *User Modeling and User-Adapted Interaction*, 7(4):279–304, 1997.

[And05] Joerg Anders. URL: http://rnvs.informatik.tu-chemnitz.de/ jan/MPEG/MPEG_Play.html, retrieved 2005.

[AY99] Y. Alp Aslandogan and Clement T. Yu. Techniques and systems for image and video retrieval. In *IEEE TKDE Special Issue on Multimedia Retrieval*, 1999.

[BC00] Kristin P. Bennett and Colin Campbell. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13, 2000.

[BC06] Darin Brezeale and Diane J. Cook. Using closed captions and visual features to classify movies by genre. In *Poster session of the Seventh International Workshop on Multimedia Data Mining (MDM/KDD2006)*, 2006.

[BC07] Darin Brezeale and Diane J. Cook. Learning video preferences from video content. In *Eighth International Workshop on Multimedia Data Mining (MDM/KDD2007)*, 2007.

[BCar] Darin Brezeale and Diane J. Cook. Automatic video classification: A survey of the literature. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, to appear.

[BFB94] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[BHC98] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI98)*, pages 714–720, 1998.

[Bim99] Alberto Del Bimbo. *Visual Information Retrieval*. Morgan Kaufman, San Francisco, CA, 1999.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

[BM01] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. *KDD '01: Proceedings of*

the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pages 245–250, 2001.

[DA00] Mark S. Drew and James Au. Video keyframe production by efficient clustering of compressed chromaticity signatures. In *Poster session of the eighth ACM international conference on Multimedia (MULTIMEDIA '00)*, pages 365–367, 2000.

[Das00] Sanjoy Dasgupta. Experiments with random projection. In *Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[DAW00] Nevenka Dimitrova, Lalitha Agnihotri, and Gang Wei. Video classification based on HMM using text and faces. In *European Signal Processing Conference (EUSIPCO2000)*, 2000.

[DEKM98] R Durbin, S Eddy, A Krogh, and G Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK, 1998.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, New York, NY, 2nd edition, 2001.

[ELL01] Brian S. Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Oxford University Press, New York, NY, 4th edition, 2001.

[Eri97] Stefan Eriksson. Personal tv program schedules: an introduction to usability and experiments with neural networks as a proposed solution, 1997.

[FB03] Xiaoli Zhang Fern and Carla E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of 20th International Conference on Machine learning (ICML2003)*, 2003.

[FBY92] William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[FLE95] Stephan Fischer, Rainer Lienhart, and Wolfgang Effelsberg. Automatic recognition of film genres. In *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*, pages 295–304, 1995.

[FLE04] Jianping Fan, Hangzai Luo, and A.K. Elmagarmid. Concept-oriented indexing of video databases: toward semantic sensitive retrieval and browsing. *IEEE Transactions on Image Processing*, 13(7):974–992, 2004.

[FLXW04] Jianping Fan, Hangzai Luo, Jing Xiao, and Lide Wu. Semantic video classification and feature subset selection under context and concept uncertainty. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pages 192–201, 2004.

[Fod02] Imola Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[For03] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[GGP00] Susan Gauch, John M. Gauch, and Kok Meng Pua. The VISION Digital Video Library Project. In Allen Kent, editor, *The Encyclopedia of Library and Information Science, Vol. 68, Supplement 31*. Marcel Dekker, August 2000.

[Gha03] Mohammed Ghanbari. *Standard Codecs: Image Compression to Advanced Video Coding*. The Institution of Electrical Engineers, London, United Kingdom, 2003.

[GLD03] Xavier Gibert, Huiping Li, and David Doermann. Sports video classification using HMMs. In *International Conference on Multimedia and Expo (ICME '03)*, volume 2, pages II–345–348, 2003.

[Gra95] Amara Graps. An introduction to wavelets. *IEEE Computational Sciences and Engineering*, 2(2):50–61, 1995.

[Gro05] GroupLens Research, University of Minnesota. One Million Ratings MovieLens Dataset, URL: http://www.cs.umn.edu/Research/GroupLens, retrieved 2005.

[HFF05] G. Y. Hong, B. Fong, and A.C.M. Fong. An intelligent video categorization engine. *Kybernetes*, 34(6):784–802, 2005.

[HK06] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman, San Francisco, CA, 2nd edition, 2006.

[HLW+99] Jincheng Huang, Zhu Liu, Yao Wang, Yu Chen, and Edward K. Wong. Integration of multimodal features for video scene classification based on HMM. In *Third IEEE Workshop on Multimedia Signal Processing*, pages 53–58, 1999.

[HS81] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *AI*, 17(1-3):185–203, August 1981.

[IL97] Giridharan Iyengar and Andrew Lippman. Models for automatic classification of video sequences. In Ishwar K. Sethi and Ramesh C. Jain, editors, *Proceedings of SPIE Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 216–227, 1997.

[Jac05] Keith Jack. *Video Demystified*. Newnes, Burlington, MA, fourth edition, 2005.

[JCB01] R.S. Jadon, Santanu Chaudhury, and K.K. Biswas. A fuzzy theoretic approach for video segmentation using syntactic features. *Pattern Recognition Letters*, 22(13):1359–1369, 2001.

[JCB02] R.S. Jadon, Santanu Chaudhury, and K.K. Biswas. Generic video classification: An evolutionary learning based fuzzy theoretic approach. In *Indian Conference on Computer Vision, Graphics, and Image Processing (ICVGIP)*, 2002.

[JFS95] Charles E. Jacobs, Adam Finkelstein, and David H. Salesin. Fast multiresolution image querying. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 277–286, 1995.

[KA95] Nachimuthu Karunanithi and Joshua Alspector. A feature-based neural network movie selection approach. In *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications, (IWANNT'95)*, pages 162–169, 1995.

[KDLF97] Vikrant Kobla, David S. Doermann, King-Ip Lin, and Christos Faloutsos. Compressed-domain video indexing techniques using DCT and motion vector information in MPEG video. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 200–211, 1997.

[KGS+01] Kaushal Kurapati, Srinivas Gutta, David Schaffer, Jacquelyn Martino, and John Zimmerman. A multi-agent TV recommender. In *Proceedings of the UM2001 Workshop on Personalization in Future TV (TV01)*, 2001.

[KL88] W.J. Krzanowski and Y.T. Lai. A criterion for determining the number of groups in a dataset using sum of squares clustering. *Biometrics*, 44:23–34, 1988.

[LDA01] Cheng Lu, Mark S. Drew, and James Au. Classification of summarized videos using hidden markov models on compressed chromaticity signatures. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 479–482, 2001.

[LH02] Wei-Hao Lin and Alexander Hauptmann. News video classification using SVM-based multimodal classifiers and combination strategies. In *ACM Multimedia*, 2002.

[LHW98] Zhu Liu, Jincheng Huang, and Yao Wang. Classification of TV programs based on audio information using hidden markov model. In *IEEE Signal Processing Society Workshop on Multimedia Signal Processing*, pages 27–32, 1998.

[Lie99] Rainer Lienhart. Comparison of automatic shot boundary detection algorithms. In *In SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 290–301, 1999.

[MC85] Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[MM04] Wendy L. Martinez and Angel R. Martinez. *Exploratory Data Analysis with MATLAB*. Chapman & Hall/CRC, Boca Raton, FL, 2004.

[Moo07] Andrew W. Moore. Statistical Data Mining Tutorials, URL: http://www.autonlab.org/tutorials/, retrieved 2007.

[MPE91a] MPEG-1. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s. ISO/IEC 1117-1: Systems, November 1991.

[MPE91b] MPEG-1. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s. ISO/IEC 11117-2: Video, November 1991.

[NAT98] Jeho Nam, Masoud Alghoniemy, and Ahmed H. Tewfik. Audio-visual content-based violent scene characterization. In *International Conference on Image Processing (ICIP '98)*, volume 1, pages 353–357, 1998.

[Net07] Netflix. Netflix Prize Dataset, URL: http://www.netflixprize.com, retrieved 2007.

[NT98] Jeho Nam and Ahmed H. Tewfik. Progressive resolution motion indexing of video object. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98)*, volume 6, pages 3701–3704, 1998.

[Old92] Gabriella Oldham. *First Cut: Conversations with Film Editors*. University of California Press, Berkeley, CA, 1992.

[PHL04] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. volume 6, pages 90–105, 2004.

[Por80] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Poy96] Charles Poynton. *A Technical Introduction to Digital Video*. John Wiley & Sons, New York, NY, 1996.

[PTRV97] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: a probabilistic analysis. In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168, 1997.

[Pyl99] Dorian Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann, San Francisco, CA, 1999.

[QGJ+01] Wei Qi, Lie Gu, Hao Jiang, Xiang-Rong Chen, and Hong-Jiang Zhang. Integrating visual, audio and text analysis for news video. In *Seventh IEEE International Conference on Image Processing (ICIP 2000)*, 2001.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RJ86] Lawrence R. Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[RMP01] Matthew Roach, John S. Mason, and Mark Pawlewski. Motion-based classification of cartoons. In *Proceedings of the International Symposium on Intelligent Multimedia*, pages 146–149, 2001.

[RMX02] Matt Roach, John Mason, and Li-Qun Xu. Video genre verification using both acoustic and visual modes. In *International Workshop on Multimedia Signal Processing*, 2002.

[Rob04] Gary D. Robson. *The Closed Captioning Handbook.* Focal Press, Burlington, MA, 2004.

[RS02] Zeeshan Rasheed and Mubarak Shah. Movie genre classification by exploiting audio-visual features of previews. In *IEEE International Conference on Pattern Recognition*, 2002.

[RSS03] Zeeshan Rasheed, Yaser Sheikh, and Mubarak Shah. Semantic film preview classification using low-level computable features. In *3rd International Workshop on Multimedia Data and Document Engineering (MDDE-2003)*, 2003.

[SC99] Barry Smyth and Paul Cotter. Surfing the digital wave: Generating personalised television guides using collaborative, case-based recommendation. In *Proceedings of the Third International Conference on Case-based Reasoning*, 1999.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[SPN+05] Uma Srinivasan, Silvia Pfeiffer, Surya Nepal, Michael Lee, Lifang Gu, and Stephen Barrass. A survey of mpeg-1 audio, video and semantic analysis techniques. *Multimedia Tools and Applications*, 27(1):105–141, 2005.

[Sym04] Peter Symes. *Digital Video Compression*. McGraw-Hill, New York, NY, 2004.

[TDV00a] Ba Tu Truong, Chitra Dorai, and Svetha Venkatesh. Automatic genre identification for content-based video categorization. *Proc. 15th International Conference on Pattern Recognition*, IV:230–233, 2000.

[TDV00b] Ba Tu Truong, Chitra Dorai, and Svetha Venkatesh. New enhancements to cut, fade, and dissolve detection processes in video segmentation. In *Proceedings of the eighth ACM international conference on Multimedia (MULTIMEDIA '00)*, pages 219–227, 2000.

[TI94] Takenobu Tokunaga and Makoto Iwayama. Text categorization based on weighted inverse document frequency. Technical report 94-TR00001, Department of Computer Science, Tokyo Institute of Technology, 1994.

[TWH01] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society, Series B*, 63(2):411–423, 2001.

[vid07] videolectures.net. URL: http://www.videolectures.net, retrieved 2007.

[VL00] Nuno Vasconcelos and Andrew Lippman. Statistical models of video structure for content analysis and characterization. *IEEE Transactions on Image Processing*, 9(1), 2000.

[WAD00] Gang Wei, Lalitha Agnihotri, and Nevenka Dimitrova. TV program classification based on face and text processing. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages 1345–1348, 2000.

[Wal99] James S. Walker. *A Primer on Wavelets and their Scientific Applications*. CRC Press, Boca Raton, FL, 1999.

[War00] Leslie Ward. Introduction to wavelets and their applications. Course notes, 2000.

[WCY03] Peng Wang, Rui Cai, and Shi-Qiang Yang. A hybrid approach to news video classification multimodal features. In *Proceedings of the Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing and the Fourth Pacific Rim Conference on Multimedia*, volume 2, pages 787–791, 2003.

[WDV+03] Hualu Wang, Ajay Divakaran, Anthony Vetro, Shih-Fu Chang, and Huifang Sun. Survey of compressed-domain features used in audio-visual indexing and analysis. *Journal of Visual Communication and Image Representation*, 14(2):150–183, June 2003.

[WF00] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, San Francisco, 2000.

[WS99] Gang Wei and Ishwar K. Sethi. Face detection for image annotation. *Pattern Recognition Letters*, 20(11-13):1313–1321, 1999.

[YLM+06] Xun Yuan, Wei Lai, Tao Mei, Xian-Sheng Hua, Xiu-Qing Wu, and Shipeng Li. Automatic video genre categorization using hierarchical

SVM. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 2905–2908, 2006.

[ZKB+04] John Zimmerman, Kaushal Kurapati, Anna L. Buczak, Dave Schaffer, Jacquelyn Martino, and Srinivas Gutta. TV personalization system: Design of a TV show recommender engine and interface. In Liliana Ardissono, Alfred Kobsa, and Mark Maybury, editors, *Personalized Digital Television: Targeting Programs to Individual Viewers*. Kluwer, 2004.

[ZKS93] HongJiang Zhang, Atreyi Kankanhalli, and Stephen W. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1:10–28, 1993.

[ZTL01] Weiyu Zhu, Candemir Toklu, and Shih-Ping Liou. Automatic news video segmentation and categorization based on closed-captioned text. In *IEEE International Conference on Multimedia and Expo (ICME 2001)*, pages 829–832, 2001.

BIOGRAPHICAL STATEMENT

Darin Brezeale was born in Dallas, Texas, in 1966. He received a B.S. degree in electrical engineering in 1992, an M.S. degree in computer science engineering in 1999, an M.A. degree in economics in 2002, and a Ph.D. in Computer Science Engineering in 2007, all from the University of Texas at Arlington. His research interests are in artificial intelligence and user modeling.