Automatic Classification of Television Commercials

Tom Nijmeijer

Tutors:
Paul van der Vet
Theo Huibers
Djoerd Hiemstra
Arjen de Vries

University of Twente
Human Media Interaction

February 25, 2008

Abstract

The last years we have seen a rapid growth of the World Wide Web and vastly increasing amounts of multimedia on it. As the volume of multimedia increases it becomes increasingly important to be able to efficiently search through this multimedia. Unfortunately searching through multimedia is less straightforward than searching through text data. Most commonly search queries are expressed in words, which match text documents in format. Multimedia documents on the other hand are not inherently expressed in words. Thus techniques have to be found to that can match textual search queries to the data available in multimedia documents. In this thesis we explore the possibility of using image and video analysis techniques to classify television commercials. If we can generate proper classifications for the television commercials we can then search through them using normal textual search queries.

Classifying television commercials is a complex process. In a number of consecutive steps the raw video data has to be filtered down in order to eventually allow a machine learning algorithm to learn the proper classifications. Using the available literature an approach for this filtering was devised. The approach starts with the detection of shot boundaries, followed by the extraction of a keyframe from each detected shot. After that keypoints are selected in each keyframe and described by SIFT descriptors. A clustermodel is trained on the entire collection of SIFT descriptors to obtain a vocabulary of 'visual' words. A histogram of the 'visual' words is then constructed for each keyframe. These keyframe histogram representation are used by a machine learning algorithm to train a classifier.

In our experiments we have optimized two aspects of the classification approach in order to improve the classification results. Firstly a number of visual vocabularies of different sizes were constructed and tested in order to find an optimal size. The results showed that the optimal size for the visual vocabulary was not large but rather small. We suspect that it is easier to train the classifier with a small vocabulary, because the smaller vocabulary has already generalized the keypoints to a further degree.

The other aspect of the classification approach that was further optimized was the choice of machine learning used to train the classifier. The results showed that a support vector machine implementation managed to produce the best classification results. It significantly outperformed other machine learning algorithms based on decision trees, decision lists and Bayesian learning. This result is not uncommon, but unfortunately no proper explanation for the success of support vector machines could be found.

In the final experiments we aggregated the classifications of the individual keyframes to obtain classifications for the television commercials. As expected the classifications of the commercials as a whole were quite a bit better than the classifications of the individual keyframes. Aggregating the keyframe classifications was once done by assigning each keyframe equal weight and once by weighting the keyframes by the probability assigned to the class label generated for it. There was no notable difference in performance between these two options.

Contents

Chapter 1

Introduction

Television these days is rife with commercials. The programs and films viewers are watching are intermittently interrupted with short messages created to encourage us to buy one product or another. Most of the time we, the viewers, are not interested in watching these unwanted interruptions to our favorite television program. Unfortunately for most television channels the financial income generated by broadcasting commercials is needed to survive. In the past, work has been done on creating methods of automatically detecting television commercials and extracting them from the television stream so that we may circumvent the annoyance of watching them.

But what if we are actually interested in seeing television commercials of a certain type; if for example, we want to buy a new car. In such a case we might actually want to see car commercials, though we'd still like to avoid any other commercials. To make this possible we would need something more than mere television commercial detection. We would need to actually recognize the subject of commercials. Performing such a task is quite a bit more difficult than the detection task, but it is exactly what we aim to do in this thesis.

In more scientific terms the task of recognizing television commercials involves the analysis of video or image data in order to extract some semantic information from them. This field of image/video analysis has received quite some attention in the scientific literature. This is mostly due to the rise of the Internet and the multimedia content on it. As the amount of data on the world wide web is ever increasing, there is a growing need for efficient and effective ways to search through all this data. Searching for textual documents is relatively easy, as the search queries of the user are typically in the same textual format. It gets harder when searching for images and videos (which is effectively a sequence of images), because these media are not made up of words.

In the rest of this introduction we'll give a short preliminary overview of the obstacles faced when trying to classify or annotate an image or video. After this short overview of the field of image annotation we will explicitly define the goal of this thesis project. In this problem description we will not only discuss what we have tried to accomplished in this project, but also specifically what we have not tried to accomplish. In other words, the restrictions we have placed upon ourselves to narrow the problem to a more manageable level. Finally we will discuss the structure of the rest of this thesis.

1.1 Video and image retrieval and annotation

The field of video and image analysis is very broad and as such giving a complete overview of it is close to impossible. In short, image or video analysis involves the extraction of meaningful information from an image or video. This can range from something as relatively simple as reading a barcode to the analysis of an image in order to annotate it with meaningful words. The latter is what we aim to do in this research project.

Closely related to image annotation is the field of image retrieval. The goal in this field is to find a proper image in a large collection of images based on an information query. One way to express the query is by providing an example image or sketch. Unfortunately, for this approach to work the user has to find and supply an example picture which is not always so easy. It would be easier for the user if he could express his query in words. For this to be possible however, we need to be able to determine the proper semantic terms for images in the collection. This is where image annotation comes in.

According to Papadopoulos et al. [PMDK06] there are two basic approaches to image annotation: explicit approaches that are based on models and implicit approaches that use machine learning techniques. In model based approaches a domain model is manually constructed to support the annotation of the images. This is a reasonable option for narrow domains [SWS+00], but for a broad domain such as video commercials this is not a very desirable approach. The domain model would simply be too large to construct by hand. Therefore we would rather use a machine learning approach. Restricting the choice of annotation approach to machine learning approaches however still leaves us with a large number of approaches. A recent survey on image annotation [DLW05] refers to a number of such approaches, but unfortunately makes no attempt at comparing the different techniques.

In the search for an appropriate image annotation or retrieval approach we come across plenty of candidates. For example there is the Multiple Bernoulli relevance model [FML04], the continuous-space relevance model [LFM04] and the image inference network model [MM04]. These are but a few of the approaches to the task of image annotation, some more approaches can be found in [LW06] and [ZI06]. To make matters more complicated the process of annotating an image or video is not a single atomic process. It typically consists of a number of more or less independent parts for which there are multiple options.

One such part of the annotation process is the extraction of low level features. Although computers these days are quite powerful, they are still not powerful enough to handle an entire image as a whole as input to a machine learning algorithm. To reduce the amount of data passed to the machine learning algorithm low-level features are typically extracted from the image. There are a multitude of different low-level features that can be extracted. So besides finding a good approach to image annotation we'll also need to find a good set of low-level features to extract. Unfortunately the selection of low-level features does not always get that much attention in the literature concerning image annotation. The articles on the multiple-Bernoulli relevance model [FML04] and continuous relevance model [LFM04] merely state that a set of 18 color and 12 texture features, without explaining why these low-level features are chosen that way. The image retrieval survey article by Rui, Huang and Chang [RHC99] gives a good overview of the available types of low-level features. However the survey does not tell us which features to use.

Choosing the right set of low-level features is but one of the choices to be made. So far we have discussed image annotation, but in our case we're aiming to annotate video fragments. The papers of Feng, Manmatha and Lavrenko [LFM04] [FML04] suggest that one can effectively label videos by using image analysis techniques on the keyframes of the video. Assuming this is a good approach it brings up the question how these keyframes are to be extracted from the video fragment. All in all there are a number of decisions to be made before we can actually start building and testing our classifier.

1.2 Research goal

As the title of this report says, in this thesis we aim to automatically classify television commercials. That means we want to classify a car commercial as such. If possible we would even like the classification to identify the type of car. Before proceeding to define the research questions we wish to see answered, there are a few remarks to be made to further specify the research goal.

First of all the research goal does not include the detection of commercials. That means we will not take a raw television stream and extract the commercials from among the normal programming. This is an interesting problem to tackle in itself, but of an entirely different nature than trying to recognize commercials. Luckily the detection of commercials is not an integral part of the process of classifying commercials. As long as we have video data that constitutes commercials we can classify them. So for our experiments we will use video fragments that have already been determined to consist entirely of commercials.

Obviously television commercials are typically not silent. So besides the visual data we could also use the audio data channel to classify the commercials. However, we will restrict ourselves in this thesis to the video data only. Using the audio data of a commercial would likely be useful for the classification of the television commercials. Analyzing audio data however is quite different from the analysis of video data. Rather than spreading the attention of this thesis between audio and video analysis we prefer to focus on video analysis only. Getting the analysis of the video data right will prove to be enough of a challenge.

In summary the goal of this research project is to classify television commercial fragments based solely on their image data. In the previous section we have indicated that this will not be an easy task to accomplish. First of all we'll need to find out exactly how to accomplish the task. Which tools and/or techniques will we need to use and in what order? Only once we have answered this first question can we make our commercial classifier. Even if we construct a good classifier there is still no guarantee that it will perform well. That means we will have to test the classifier to get an idea of how well it performs. All in all there are two main questions to be answered:

1. Which techniques need to be used in which order to classify television commercials?
2. How well does a television commercial classifier perform using available techniques?

1.3 Thesis overview

The rest of this thesis will be organized as follows:
In the first chapter we will discuss which techniques we have used for the task of classifying television commercials and why we have chosen those techniques. We will see the classification is not a simple matter of feeding the video fragments to a machine learning algorithm. The amount of data present in a video fragment is simply far too large to train a classifier. Thus the raw video data in a video fragment will have to be reduced to a more manageable level before training the classifier. Obviously this means the elimination of a large part of the data. If this is not done in an intelligent manner it will greatly influence the performance of the classifier.

After discussing our classification approach we'll talk about the collection of video fragments used for the experiments. This includes the acquisition of a usable collection of raw video fragments to be classified. As mentioned in the research goal these video fragments consist solely of commercials, because we are not interested in the detection of commercials as such. Besides acquiring the video fragments, we'll discuss the process of annotating the commercials. Without these annotations it would not have been possible to train or test the classifier.

In chapter 4 we will present the results of our attempts at constructing a television commercial classifier. In order to optimize the performance of the classifier to a degree we have constructed several classifiers by varying certain parameters. The results of all these attempts will be presented for further comparison.

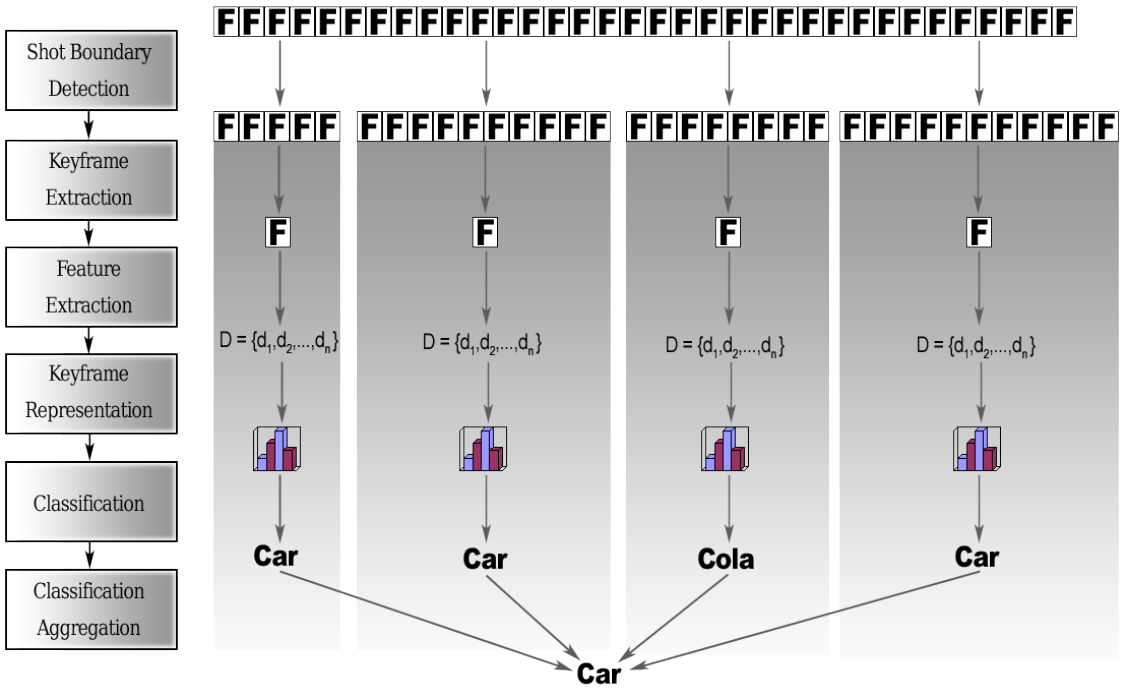
In the final chapter of this thesis we will evaluate the results we have achieved. We will assess the usefulness of the automatic classification of television commercials in a practical sense. Have we actually been successful in constructing a classifier that can classify commercials with a decent reliability? Besides this purely practical evaluation we will also compare our results to the results achieved in other relevant scientific literature. Are we achieving better or worse performance than comparable tasks in the literature, and to what can the differences be attributed? To close the report we will discuss in which direction the research can be taken to further the progress of automatically analyzing television commercials and images in general.

Chapter 2

The classification approach

Video annotation is a hard task to accomplish successfully, requiring a number of different operations in succession to achieve the final goal of providing a video fragment with an appropriate annotation. The initial input for the classification process consists of raw video data. Such video data is far too large and complex to be processed by a machine learning algorithm in its raw form. Therefore the data will have to be filtered and transformed through a number of preprocessing steps to reduce the complexity to a more manageable level. Fortunately it is reasonably safe to assume that the raw data contains a lot of redundant and noisy data which can safely be eliminated. The problem however lies in determining which data to eliminate. The task of processing videos and images in order to annotate or classify them has received quite some attention in research literature. Using this literature we have determined a usable approach to the classification task.

Figure 2.1: *Overview diagram of the experimental approach.*



2.1 Shot detection

The first step towards reducing the raw video to a more manageable level is generally accepted to be temporal video segmentation. This divides the video stream into smaller parts that can be analyzed separately. For video annotation the video stream is usually segmented into shots. A shot can be defined as a sequence of video frames recorded by a single camera in one stretch. Segmenting a video stream into shots means we have to detect where the transitions between shots take place. This problem of determining where the transitions between the shots are is called shot boundary detection.

Another form of temporal video segmentation is scene segmentation. A scene is a part of a video taking place in a single setting or otherwise belonging together based on the content of the video. Scenes are greater units than shots, one scene usually consists of several shots. The task of distinguishing scenes is harder than detecting shots as it requires knowledge of the semantics of a video. We actually need to know the subject of a scene to identify transitions from one scene to another. Shot transitions on the other hand are of a technical nature, meaning we can detect them without needing to know the subject of the video. In our case segmenting scenes would most probably not be very useful as commercials are typically quite short, usually consisting of just one scene. Considering this we can afford to limit ourselves to the more solvable problem of detecting shot boundaries.

Although the task of detection shot boundaries is easier than detecting scene boundaries, it is still no trivial task. Like so many computer vision tasks, though the task of distinguishing between shots is easy for a human being it is a lot more difficult to solve for a computer. Because of this shot boundary detection has received ample attention in computer vision research. Nevertheless the shot boundary detection problem is still an open problem to this day. The main obstacle towards successful shot boundary detection lies in the distinction between abrupt transitions (cuts) and gradual transitions. An example of a gradual transition would be where one shot fades gradually into another shot. Abrupt transitions can be detected quite well, because the image will quickly change from one frame to the next. Detecting gradual transitions however is far harder as they have to be distinguished from camera and object movements which cause frame differences of similar order.

Various approaches have been proposed to deal with shot boundary detection. In a survey article by Korinska and Carrato [KC01] a large number of these approaches are reviewed. Some of these approaches operate on uncompressed video data while others directly use compressed data (MPEG). The approaches operating on raw compressed video data unfortunately exhibit a number of shortcomings. The problem stems from the fact that MPEG encoded videos consist of three types of frames. Intra (I) frames are encoded using only the data present in that frame. Predicted (P) and bi-directional (B) frames on the other hand, use the information encoded in neighboring frames to construct their own image. To be able to execute shot detection on the P and B frames a part of their image has to be reconstructed from the neighboring frames which costs valuable computation time. The alternative is to ignore P and B frames and only analyze the I frames, but this results in an unwanted loss of resolution. Other shortcomings of these algorithms include:

- They can only handle very short gradual transitions and are unable to classify them.
- They do not distinguish between quick camera movements and real shot boundaries.
- They use a lot of adjustable threshold values.

Considering this there is no clear advantage to using methods operating on compresses data above methods using uncompressed data. Restricting ourselves to uncompressed data approaches still leaves us with a number of possibilities. We will briefly describe the main types of approaches. For a more complete overview we refer to [KC01].

- **Pair-wise pixel comparison**
  These approaches calculate the sum of the differences between each pair of pixels between consecutive frames. If this sum exceeds a preset threshold a shot boundary is detected. This is the simplest approach to shot boundary detection but suffers from many limitations. By simply summing the pixel differences these methods cannot distinguish between changes across the entire image and more local yet larger changes in just a part of the image. As a result fast object movements in a shot can be mistaken for shot boundaries.

- **Block based comparison**
  In these more complex approaches a frame is divided into a number of blocks. For each of these blocks a dissimilarity score is calculated. If enough blocks exceed a threshold dissimilarity score a shot boundary is detected. These approaches are more robust than pair-wise comparison approaches with regard to small object moving on the screen. Such small objects will only cause a limited number of blocks to exceed the threshold, which will not result in the detection of a cut.

- **Histogram comparison**
  Histogram comparison approaches construct histograms of the gray-levels or colors in the frames. Such histograms will not change too much across frames of the same shot even if there is a lot of movement in the shot. On the other hand the histograms should differ greatly when crossing a shot boundary as a new background and objects are introduced. To account for gradual transitions a secondary lower threshold can be used. When a frame transition exceeds this lower threshold it is marked. As long as the following frame differences keep exceeding the low threshold the differences are summed. If this sum exceeds the higher threshold a gradual transition is detected.

- **Clustering based segmentation**
  Another way of approaching shot boundary detection is by viewing it as a two-class clustering problem: transitions and non-transitions. A clustering algorithm such as K-means is employed to cluster the frame dissimilarity measures. The main advantage of this approach is that it does not use threshold values that have to be determined or estimated beforehand. Clustering however is quite a costly process.

- **Feature based segmentation**
  One approach falling in this category detects intensity edges in frames and analyzes their appearance and disappearance across frames. Shot boundaries are detecting when intensity edges appear or disappear far from each other.

- **Model driven segmentation**
  The previously mentioned approaches all approached the problem from a data analytic viewpoint. It is also possible to approach the problem in a top-down model driven manner. In these approaches models are constructed and trained for the various transition types. Following that, the transitions are detected using these models. Like the clustering based approaches, these model based approaches generally do not need preset threshold values.

Looking at the list of shot boundary detection approaches we are still faced with the question as to which approach would be appropriate for our task. Unfortunately there are few comparative studies available that compare the various shot boundary detection methods [BR96]. Obviously we would prefer a simple yet effective approach. The histogram based type of approach seems to fit the bill quite nicely. It does not require machine learning techniques like the clustering and model based approaches yet is reported to perform quite well. We have found a working implementation of a histogram based shot boundary detection algorithm using twin thresholds, histogram smoothing and flash compensation that is sufficient for our purposes. Twin thresholds are employed to be able to detect gradual transitions as described in the paragraph on histogram-based algorithms. Histogram smoothing reduces the differences between frames due to noise.

Finally flash compensation prevents single frames that are very different from their neighboring frames from being detected as shot boundaries.

2.2 Keyframe extraction

By segmenting the video fragment into shots we have not yet reduced the amount of information. After shot detection the next step is usually to extract one or more keyframes from each shot. These keyframes are taken to represent the content of the entire shot. By reducing a shot to a limited number of frames we are greatly reducing the amount of information that has to be analyzed further which (as said) is necessary if we want to successfully train a classifier.

Before proceeding to review the different methods of extracting keyframes there is another question that begs answering. Is there a viable alternative to extracting keyframes? It is a given that the data has to be reduced to a more manageable level. But rather than extracting a single frame (or just a limited number) from one shot could we not analyze all frames of a shot and choose a set of representative features afterwards? Although such an approach seems possible there are no immediately apparent advantages. No matter which approach we choose, at some point we need to eliminate the majority of the data. Intuitively it is a lot easier to think about which information should be eliminated at the level of frames rather than low-level features. In addition we have not been able to find any literature using such an alternative approach. Given that there is a lot more to be found about keyframe extraction we will not explore the alternative any further.

By extracting keyframes the raw video data is reduced from several hundreds of frames per shot to a mere handful of keyframes. Obviously this means we lose the majority of frames and data in this step. Not only do we lose the information captured in the individual frames, the motion present in the sequence of frames is also lost. This motion can include both camera motion and objects motion on the screen. The question is if we need this motion information to successfully classify television commercials. When thinking about different commercials we most likely think about certain objects occurring in the various commercials. In other words, to classify a commercial we are mostly interested in the objects that appear during a commercial. The movement of these objects is not so obviously relevant to distinguish between different commercials. That does not necessarily mean the motion information is useless for the classification of commercials. It could conceivably happen that a classifier trained on the motion information would be quite successful Unfortunately there is no way to be certain beforehand and incorporating motion information in our classification approach would have made things overly complex. Assuming the motion information is not a necessity for the successful classification of commercials we have chosen to accept the loss of motion information during the keyframe extraction.

The other information we are losing due to keyframe extraction is the image data present in the frames not chosen as keyframes. Whether this loss of information is significant depends on the effectiveness of the keyframe extraction algorithm. If the keyframe extraction algorithm works well, the keyframes extracted will provide a faithful representation of all frames in a shot. In principle we want the keyframes to be as representative as possible, though we will discuss the importance of this further on. There are a number of different approaches to keyframe extraction ranging from very simple to rather complex. The simplest approach is to pick one or more predetermined frames from the shot such as the first, middle or last frame. This will work decently for static shots, but not for more dynamic shots. Besides this very naive approach there are a number of other more complex approaches. A good overview of the different approaches can be found in [ZT01].

- **Histogram based**
  In these methods the first frame of a shot is selected as the first keyframe. After that the histogram difference of the subsequent frames compared to the latest selected keyframe is calculated. If this difference exceeds a certain threshold a new keyframe is determined and the process is repeated. This method is very similar to the shot boundary detection method we have used. If we were to use this method, the threshold would have to be set lower than the thresholds used for the shot boundary detection, otherwise the result would simply be that the first frame of each shot is selected as the keyframe. Furthermore this method always selects the first frame as the keyframe, which is not a good idea if we are dealing with gradual transitions.

- **Cluster based**
  These methods cluster the frames of a single shot. A keyframe is selected for each of the major clusters by selecting the frame closest to the cluster centroid. Clustering the frames for each and every shot is quite a costly procedure. Furthermore the number of clusters to cluster into is often a parameter of the clustering algorithm. If we would choose the number of clusters too low we would not get as many keyframes as appropriate. On the other hand if we set the number of clusters too high, the clustering algorithm would take that much longer. Thus finding a good number of clusters would be another step to optimize.

- **Motion based**
  In these methods the motion during a shot is analyzed. Keyframes are selected at the points where the motion experiences a local minimum. The method of determining the motion during the shot can range from simple pixel comparisons to more complex methods.

- **Mosaic approaches**
  These methods try to construct a single image representing the entire shot including its motion. Basically the resulting mosaic image summarizes the image information during the entire shot. The resulting mosaic image will provide a good representation of the entire shot, but the mosaic image can take on any dimension.

Even though there are more intelligent methods of choosing a keyframe than the naive approach of selecting a predetermined frame from a shot, there are still plenty of cases in which the naive approach is employed. The more complex approaches aim to result in more representative keyframes. Just how much this is the case in practice however will depend on the actual structure of the video data. In the case of a short or stationary shot, choosing the middle frame as a keyframe is not that far from optimal. The naive keyframe extraction approach only starts degrading in performance when we have to deal with longer and more dynamic shots. In television commercials we will most likely encounter quite dynamic shots, but not very long ones. This is just an assumption however.

Even if a more complex feature extraction method would lead to more representative keyframes there is still the question whether this would also reflect in the classification results. Like with most other parts of the classification approach this can only be verified by performing the classification experiment with better keyframes. To perform such an experiment we would first need better keyframes, if possible the most representative keyframes. Unfortunately we cannot really expect any of the automated feature extraction algorithms to provide the best results in all cases. Only if we extract the keyframes manually can we guarantee a set of keyframes of a certain quality. This would however be a very time consuming task. Secondly there is the problem of deciding what exactly constitutes a representative keyframe. The term representative cannot be immediately be translated into a clear set of constraints which the keyframes should meet. Taking these practical concerns into account we can conclude there is no quick and easy way to answer this question. It would be an interesting question to answer in a separate research project, but for this project we simply do not have the time to adequately answer it.

Finally there is the consideration that keyframe extraction immediately follows shot boundary detection. The purpose of shot boundary detection is to detect the boundaries of actual shots. However we have also indicated that shot boundary detection often generates false positives in the case of large motions during a shot. In a way we can say that shot boundary detection does not just segment a video stream in actual shots but also segments the video stream based on the motion. This is very similar to the keyframe extraction methods we have mentioned. Indeed we can draw parallels between the histogram and motion based keyframe extraction methods and the previously discussed shot boundary detection methods.

Taking all of this into account we decided to restrict the keyframe extraction to the naive approach, relying on the shot boundary detection method to produce shots that can be reasonably represented in a single keyframe.

2.3 Feature Extraction

The amount of data has been greatly reduced by extracting a limited number of keyframes from the video data. However keyframes are still too complex to train a classifier. Thus the data present in the keyframe images has to be filtered even further. This is usually done by calculating certain low-level features for an image. There is a broad range of low level features that can be extracted from images. Features can encode the color, texture, shape or localization of objects in an image. That does not mean there are four possible low-level features. Each type of low-level feature can be encoded in numerous different ways. Color features for example can be defined using one of many color space models. The same can be said for texture, shape and localization features. Two nice surveys concerning the various low-level features that can be extracted from images are [RHC99] and [LZLM07].

Besides the different kinds of information that can be encoded, we can distinguish between global and local features. The features in the former category encode information about an image in its entirety or a region of the image. Another way to encode features however is to determine a set of interest points from an image and construct a local descriptor for each interest point. This strategy of determining image features has proven to be useful for a lot of different image analysis tasks [MS05]. The advantages of local features are that they are distinctive (more so than global features), robust to occlusion and do not require segmentation of the image. Furthermore the local descriptors can be constructed to be invariant to scale changes, rotation and illumination changes.

Unfortunately there is no best set of features that can be used effectively for each task. Which kind of features will work depends largely on the type of application. Predicting which features will be suitable however is very hard if not impossible. This is mostly due to the so called semantic gap [SWS+00]. The semantic gap is the discrepancy in information represented by low-level features and the information expressed in semantic terms. Determining which feature values correspond to which semantic terms is a task reserved for the machine learning algorithm used later in the chain of operations. The effectiveness of this machine learning however depends on the features used. Choosing features that are closer to the semantic level of information will make the machine learning task that much easier.

In most literature we have found on the subject of image or video annotation the choice of low-level features is not explained. We are however not satisfied in choosing a set of features without some reasoning behind it. In our experimental approach we have used the SIFT features as our low-level feature of choice [Low04]. The SIFT approach is a form of local feature descriptor. As stated before, local descriptors have proven to be effective in multiple image analysis tasks. Amongst other local features descriptors the SIFT descriptor is reported to perform very well [MS05]. Considering this, the SIFT feature descriptor seems like a good choice of low-level features.

The SIFT approach consists of determining a number of interesting keypoints in the image and constructing a feature vector for each keypoint. This feature vector is called the descriptor. The construction of SIFT features is a four step process:

1. **Scale space extrema detection:**
   In this step the entire image is searched across all scales for candidate keypoints. To do this the image is smoothed using a 2D Gaussian kernel of varying size. This results in a set of smoothed images. Objects in the image will be smoothed out at the most appropriate scale. The various smoothed images are then subtracted from each other. Extrema are then detected in these difference of Gaussian images to identify the candidate keypoints.

2. **Keypoint localization:**
   In this step each candidate keypoint is fitted to the nearby data for localization and scale. Candidate keypoints that are poorly localized along an edge or with insufficient contrast are eliminated. This is necessary because in the previous step not all pixels of the image are sampled. By performing this step the most appropriate specific position for the keypoint is found.

3. **Orientation assignment:**
   After selecting the appropriate keypoints and locating them properly, the orientation of each keypoint is determined. This is done by constructing a histogram of the gradient orientations of sample points around the keypoints. The keypoint is then assigned a dominant gradient orientation in the histogram. In case of multiple dominant gradients, a keypoint is identified for each orientation.

4. **Keypoint descriptor:**
   Finally a feature vector is constructed for each keypoint. The feature vector represents a histogram of the gradient orientations in an area of sample points around the keypoint. The orientations are aligned with the keypoint orientation to achieve rotation invariance. In total the descriptor consists of 4 times 4 histograms that each contain 8 bins for the orientation.

The SIFT feature vector for each keypoint represents a histogram of the gradients around that keypoint. That means the information in the SIFT features concerns the shape as well as the local texture around the interest point. The basic SIFT descriptor however lacks any information about color. The entire process of locating interest points and constructing descriptors for them is based on a gray scale version of the keyframe. Color features are commonly used in image annotation or retrieval tasks. While shapes and texture are probably a good way to distinguish commercials, colors intuitively play an important role as well. Certain products have a specific dominant color associated to them. Orange drinks for example are usually associated with the color orange.

There are several ways imaginable to add color information to the set of features used to characterize an image. For starters we could simply use a global color histogram next to the SIFT features extracted. This however would defeat the purpose of the local SIFT descriptors which are designed to emphasize the distinctive points in an image. The only proper way to add color information to the feature set would be to add the color information to the local descriptors. Such an approach has been proposed in [VS06] and is reported to outperform the purely shape based SIFT descriptors. Unfortunately this article is too recent to have been evaluated further in literature and the results given in the article itself are based on a small test corpus.

The question is whether the information contained in the SIFT features will be sufficient to classify a keyframe and a commercial even without the color information. This depends on how we actually want to recognize a keyframe. SIFT features are specifically designed to deal with object recognition. Detecting objects in a keyframe should be a good method of identifying different commercials. Car commercials will most certainly include a car at some point during the commercial. Similar reasonings can be used for other products.

2.4 Keyframe Representation

With the extraction of the low-level (SIFT) features we come close to the desired level of data complexity for training a classifier. We are however still left with two problems. Firstly the amount of features is still slightly too large. Secondly the number of features varies from one keyframe to another. Because almost all machine-learning algorithms require the classification instances to have a fixed dimension of features, we need to transform the low-level features to a static and limited number of features per keyframe.

In theory we could choose to classify all keypoints individually and somehow determine the classification of a keyframe and commercial by combining the results. This would circumvent the problem of reducing the variable number of keypoints per keyframe to a fixed size feature vector. On the other hand we would be faced with the problem of determining a proper classification based on the classification of a lot of keypoints with no way to determine which keypoints provide a good indication of the right classification. That is assuming that a single keypoint can even be accurately classified. Considering this it is a better idea to construct a good feature representation of the keyframe and let the machine learning algorithm do the job of finding the most indicative features.

There are a number of possibilities to address the problem:

- **Bag-of-features**
  There is an approach similar to the bag-of-words model used in text retrieval [NLM99]. This approach can be described using four steps [LA07]. The first two steps are comprised of finding the points of interest and constructing descriptors for them. Following that the descriptors are quantized or clustered to a fixed number of visual 'words' which can be called a visual vocabulary. Finally the representation of the image is constructed by constructing a histogram of the visual 'words'. This bag-of-features approach [DWF+04] has been used in a number of research projects with reasonable success [ZMLS07] [LA07]. The resulting keyframe representation, represents the distribution of visual 'words' in a keyframe. The distribution is represented in the form of a histogram. What the visual 'words' mean is generally not known, but the idea is that they are predictive and discriminative for the class of the keyframe.
- **Image signatures**
  Another approach to the problem is to cluster the keypoints of a single keyframe into a set number of clusters. After this an image signature can be constructed by concatenating the cluster centroids and relative cluster sizes one after another in a feature vector [ZMLS07] [LSP05]. This image signature approach actually provides information quite similar to the bag-of-words approach. The difference is that in the bag-of-words approach the keypoints are clustered across the entire collection, while in the signature approach the keypoints are clustered for each keyframe individually. So instead of a distribution of preset visual 'words' the signature approach represents the distribution of keypoints specific to each keyframe [RTG00].
- **Pyramid match kernel**
  A final option we would like to mention is the pyramid match kernel [GD05]. This kernel function can measure the similarity between two unordered feature sets of variable length. Of course this approach is restricted for use with kernel-based machine learning algorithms. A kernel does not provide a representation of each keyframe individually but rather measures the similarity between a pair of keyframes. To compute the pyramid match kernel, multiple histograms are constructed with multidimensional bins increasing in size from one histogram to another. The keypoints of the keyframes are then assigned to their proper bins. This leads to a pyramid of histograms with an increasing number of bins. The final kernel value is a weighted sum of the newly matching keypoint pairs at the different levels of the two

histogram pyramids being compared. Effectively the kernel function is a kind of histogram intersection and the keyframe representation is formed by the histogram pyramid. As such this approach is not that different from the previous two approaches which are also essentially histograms.

As can be seen in the previous section each approach for the representation of a keyframe can eventually be seen as a histogram distribution of the keypoints in that keyframe. In that sense there is no real choice to be made. The real choice lies with the choice of bin partitioning. How we can best partition the keypoints into bins cannot really be determined beforehand. The only way to compare the different approaches would be to test them in an experiment. Unfortunately we did not have the time to execute such a comparison, so we have restricted ourselves to the bag-of-features approach.

2.5 Classification

The near final phase of the classification approach consists of training the classifier using a machine learning algorithm. As is well known there are a lot of machine learning algorithms available. Providing a full overview of all possible machine learning algorithms would be quite a challenge. Instead we will restrict ourselves to algorithms that have been used for video and or image classification before. A survey article by Liu et al [LZLM07] provides us with a overview of different algorithms that have been used to learn high level semantic concepts from low-level features.

- **Support vector machines**
  The goal of a support vector machine is to find the optimal separating hyperplane given a set of multidimensional data. This optimal separating hyperplane maximizes the distance of the plane to the nearest points in the dataset. To find this hyperplane the algorithm uses so-called support vectors which are essentially those training samples closest to the hyperplane. SVM's have a strong theoretical background. On the downside a SVM can only be trained for two classes. That means a separate SVM has to be trained for each class that has to be identified.

- **Bayesian classifiers**
  Bayesian classifiers use Bayes' rule to train a belief network that can predict the classification of an example based on the low-level features. A belief network is a directed acyclic graph in which the nodes represent variables and the edges indicate conditional dependencies between variables. In the case of a Bayesian learning algorithm the different features take on the role of nodes and the end node is the variable indicating the classification of the training example.

- **Neural networks**
  Neural networks are said to be modeled after the human brain. The network consists of nodes called neurons that are interconnected. Each neuron takes a weighted value of its incoming edges and then determines its own activation level based on this value. Like belief networks, neural networks can be acyclic in which case they are usually called perceptrons. A neural network classifier trains such a neural network using the feature variables as the initial inputs. The final output unit (neuron) of the network then indicates the classification to be assigned.

- **Decision trees**
  As the name implies, decision tree learning algorithms construct a decision tree based on the input data. Generally this is done by choosing the feature variables that are most indicative of the final classification result. A tree node is then constructed that tests the input on this variable. Based on the result of this test the training set is split. For each subset the process is repeated until all the examples left in a subset have the same classification or there are no more variables left to test on.

The input to the classifier will consist of a histogram of the distribution of keypoints in the keyframe. Semantically speaking the keypoints have no clear meaning nor do the visual 'words' in which the keypoints have been categorized. Considering this there is little point to gaining a clear picture of the decision structure the machine learning algorithm creates.

Unfortunately there is no clear theoretical case to make for the use of one algorithm or the other. Decision trees are more intuitive to use for predicate variables, but it is not impossible to use them for the continuous variables we use. The decision tree will simply be less informative in such a case, but might still be effective. Support vector machines can in principle only distinguish between 2 classes, but we can apply them for all of our classes by simply training multiple SVM's. Bayesian and neural networks both seem like good candidates for the job. Rather than choosing one algorithm or the other beforehand we have tested several algorithms and compared their performance.

2.6 Aggregating the keyframe classifications

After the machine learning algorithm has been trained the keyframes can be classified. However we are not truly interested in the classification of the keyframes. What we really want is to classify the commercials as a whole. That means we have to determine the classification of a commercial based on the classification of its constituent keyframes. The simplest way to do this is implement a simple majority vote, the classification that occurs most in the keyframes is the classification of the commercial.

A more intelligent approach would somehow weight the classification of each keyframe based on some measure of the predictiveness of the keyframe. Obviously this is only possible if we have such a measure available. One possibility for such a measure would be to use the degree of certainty the classifier has of its result if the chosen machine learning algorithm provides it. Another possibility is to weight the keyframes based on a theory of the importance of keyframes in relation to their temporal occurrence during the commercial. One such theory could be that keyframes occurring towards the end of a commercial are more predictive of the class of the commercial. In our experiments we will not use such a theory to weight the keyframes. We will however test both the aggregation based on equal weights and aggregation based on using the degree of certainty of the class labels.

Chapter 3

Data Collection

For training and testing the classifier a test collection of video commercials is required. A test collection however does not simply consist of a set of television commercial fragments. Besides the video fragments we'll also need annotations to go with the fragments. Without these annotations it would not be possible to train the classifier or to evaluate its performance afterwards. A collection of video fragments that has been annotated previously would have been ideal, but unfortunately such a complete test collection could not be found. Therefore in this chapter both the process of acquiring video fragments and annotating them will be discussed.

3.1 Acquiring the video fragments

To start with, a large amount of commercial fragments was needed for the test collection. Such a collection was found in the data gathered from the MediaCampaign project [1]. This data consists of video commercials recorded from Dutch television in 2006. In total the collection consists of 1123 video fragments solely containing television commercials during 5 consecutive days on a single TV channel. Each fragment contains in principle only a single commercial. However errors in this segmentation do exist. This includes both undersegmentation (multiple commercials per fragment) as well as oversegmentation (one commercial spanning multiple fragments). Undersegmentation turned out to be quite common in the collection. Some fragments consisted of 5 separate commercials or more. Oversegmentation fortunately was a lot less common. Because of this we could ignore these commercials rather than trying to reconstruct them. The fragments are of the AVI format using a DivX encoding.

3.2 Annotation of the fragments

While the acquisition of the necessary video fragments was relatively easy, annotating the fragments was quite a bit more time consuming. The process of annotating the video fragments was necessarily done manually. Automatically annotating the fragments is after all the goal of our research. Besides that we also have to take into account the segmentation errors of the video fragments.

We have supplied the annotations to the fragments in a separate XML file. Each XML file lists the keyframes extracted from its corresponding video fragment (more on that in the next section)

[1] http://www.media-campaign.eu/

identifying them by their frame number. In case of under-segmented fragments the XML file defines multiple commercial elements each of which has its own annotation. The keyframe elements are then grouped under the appropriate commercial element. A small number of fragments turned out to be very short clips with no real discernible images. These fragments have been marked with a special annotation tag `<unusable_fragment/>` to indicate they needed to be ignored for the rest of the experiments.

An example of an annotation XML file can be found below. This XML file describes a single fragment that contains two different commercials. The commercials are delimited by the `<commercial>` begin and ending tag. Each commercial description consists of an annotation followed by a listing of the keyframes associated to that commercial. The annotation of the first commercial (lines 3-8) in the example indicates that the first commercial concerns a product called Regenerist which is of the brand Olaz and falls in the category of personal care products. Following the annotation comes the keyframe listing (lines 9-15), in this containing five keyframes.

```
1  <fragment id="20060809-224527-234531-023">
2    <commercial>
3      <annotation>
4        <product name="Regenerist">
5          <brand name="Olaz"/>
6          <category name="Personal care"/>
7        </product>
8      </annotation>
9      <keyframes>
10       <keyframe frameno="33" text="true" numkeypoints="2080"/>
11       <keyframe frameno="127" text="true" numkeypoints="2069"/>
12       <keyframe frameno="222" text="true" numkeypoints="3005"/>
13       <keyframe frameno="299" text="true" numkeypoints="1759"/>
14       <keyframe frameno="361" text="true" numkeypoints="748"/>
15     </keyframes>
16   </commercial>
17   <commercial>
18     <annotation>
19       <storechain name="Albert Heyn">
20         <type name="Supermarket"/>
21       </storechain>
22     </annotation>
23     <keyframes>
24       <keyframe frameno="412" text="false" numkeypoints="608"/>
25       <keyframe frameno="463" text="true" numkeypoints="4040"/>
26       <keyframe frameno="486" text="true" numkeypoints="8425"/>
27     </keyframes>
28   </commercial>
29 </fragment>
```

The annotation scheme used for the video fragments has been developed ad-hoc to suit the test collection used. As far as we know there is no standard annotation scheme available for television commercials or anything similar. We can make no claims as to the appropriateness and completeness of our annotation scheme for television commercials in general. However, it has been sufficient for our purposes during the thesis.

Our annotation scheme distinguishes between six different types of television commercials. Each type of commercials has its own annotation structure associated to it. By using an annotation

structure rather than a single annotation tag we can classify the commercials at multiple levels of distinctness. This allows for different levels of classification. The types of commercials that have been distinguished during the annotation are as follows:

1. **Product:**
   This is the type of commercial most people will think about first. It advertises a specific product, which can be bought as such. Most products are of a certain brand, so whenever possible we have included this brand in the annotation. Furthermore we can group products together in a number of categories such as food wares, cars and drinks. During the annotation of our test collection we have distinguished the following categories: food, beverage, financial, detergent, cleaner, cosmetic, personal care, pharmaceutic and automotive. These three attributes are combined in an XML structure as follows:

```
<product name="Regenerist">
    <brand name="Olaz"/>
    <category name="Personal care"/>
</product>
```

2. **Brand:**
   Brand commercials are one step above the regular product commercials. Rather than advertising a single product of a brand, this type of commercial advertises for the brand as a whole. Because a brand often only includes products belonging to a single category we have included the product category in the annotation of brand commercials whenever possible. The categories used for the brands are the same as the previous product categories.

```
<brand name="Postkrediet">
    <category name="Financial"/>
</brand>
```

3. **Store Chain:**
   This type of commercial is quite similar to the product commercial. In a way a store chain can be thought of as a brand. Unlike a brand, a store chain however is not directly associated with specific products. Commercials advertising store chains usually present the latest special offer of the store chain, which can be a product or a publicity campaign. Like with brands and products, store chains can be grouped in a number of categories. These categories however are not the same as those with brands and commercials.

```
<storechain name="Albert Heyn">
    <type name="Supermarket"/>
</storechain>
```

4. **Program Preview:**
   These commercials present a TV program that can be seen in the near future. They originate from the television channel rather than an external company. Still they can been seen as advertising something, namely the television channel itself. Because these type of commercials are quite common we cannot just ignore them. Even if we wanted to ignore them, we would still need to be able to identify them automatically in order to do so. The annotation for these commercials consists of a single attribute identifying the program being advertised.

```
<program_preview name="Dancing on Ice"/>
```

5. **Charity:**
   This type of commercial advertises for a charity fund or foundation. Again the annotation consists of a single attribute identifying the charity in question.

```
<charity name="Stichting Kinderstem"/>
```

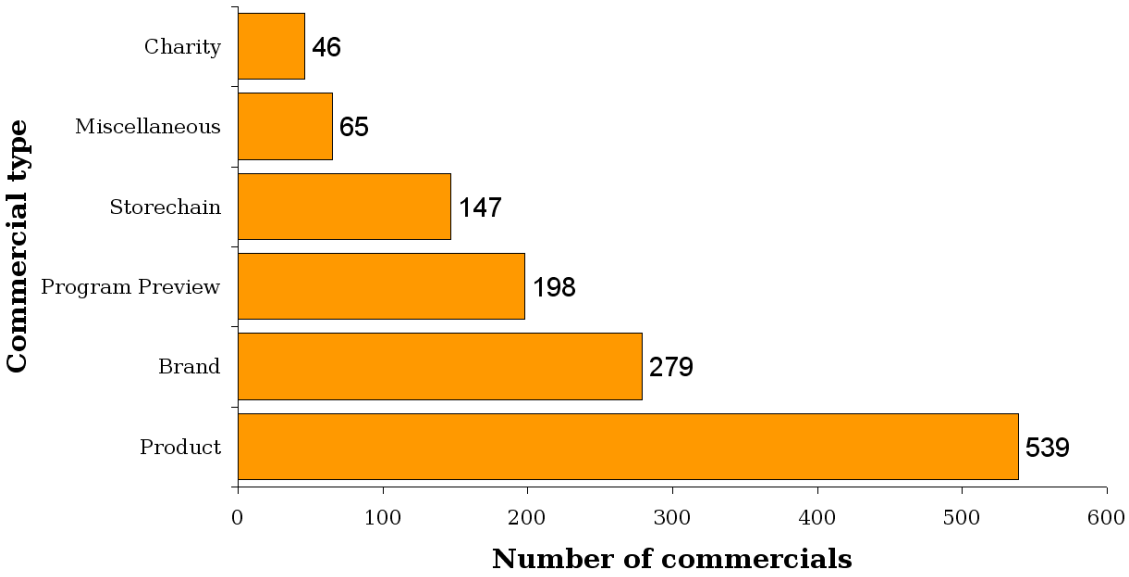
6. **Miscellaneous:**
   This category consists of those commercials that can not be classified into of the previous categories. A notable type of commercial that falls under this category are the non-commercial advertisements. These advertisements do not try to sell a certain product, brand etcetera but rather bring up certain social issues. They are fairly common in the Netherlands, but perhaps less so in other countries.

```
<miscellaneous name="Kijkwijzer"'/>
```

3.3 Collection statistics

As mentioned in the first section of this chapter the data collection consisted of a total of 1123 video fragments. Sixty-six of these fragments were deemed unusable during the process of annotation, leaving 1057 usable fragments. In these 1057 fragments a total of 1275 different commercials were recognized and annotated. The bar chart below shows us how these commercials were distributed across the commercial types defined in the previous section.

Figure 3.1: *Distribution of the television commercials across types.*



The final two bar charts show how the commercials of the product, brand and storechain were distributed across their respective categories. Like the commercial types the categories were defined ad-hoc. Another data collection would have resulted in different categories. Nevertheless the charts should give a general idea of the types of commercials encountered in the data collection.

Figure 3.2: *Distribution of the product and brand commercials across the various categories.*

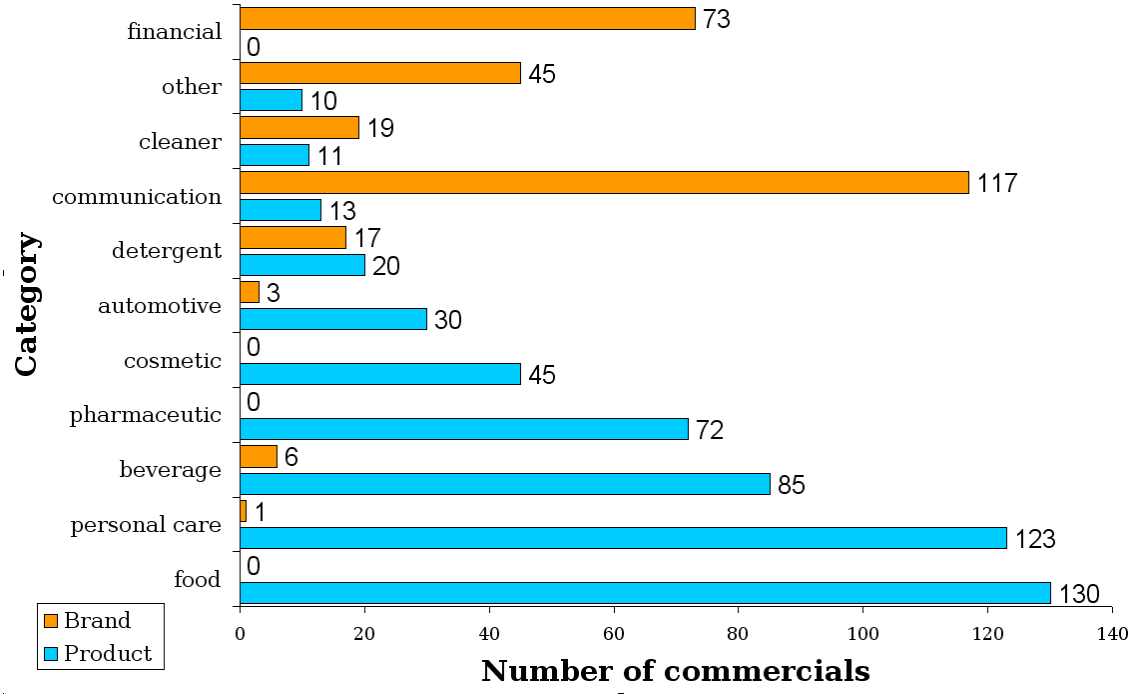
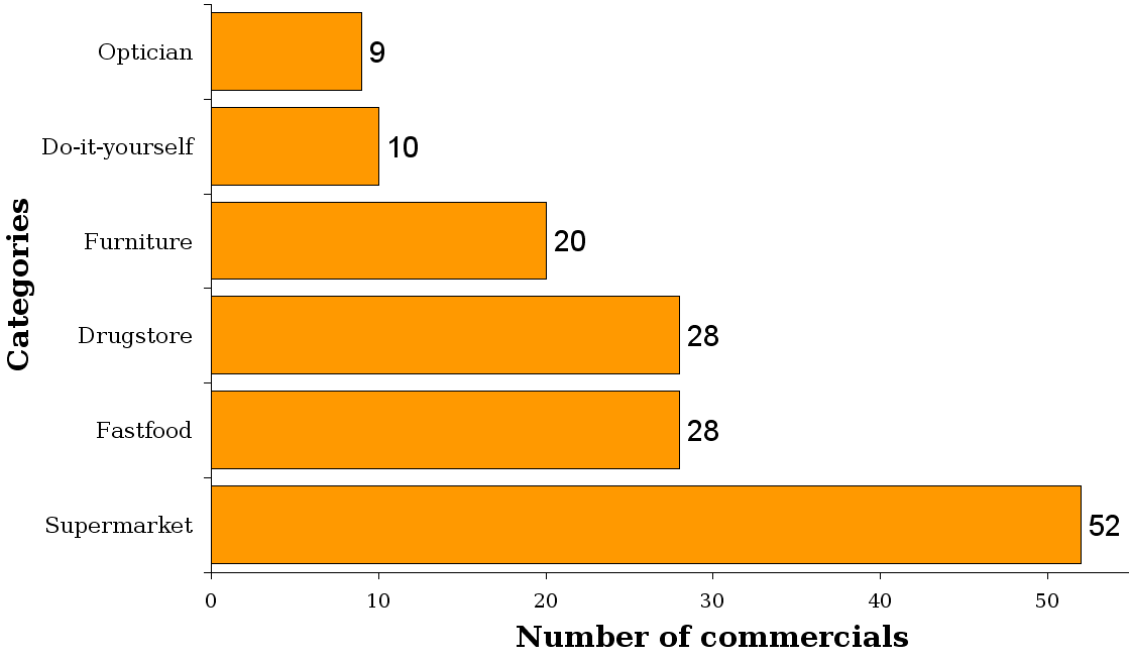


Figure 3.3: *Distribution of storechain commercials across the various storechain categories.*



Chapter 4

Experimental Results

In the previous chapters we have discussed the collection of television commercials and experimentation approach we used in the experiments. This chapter will discuss the results obtained using this collection and approach. Before we get to that however , we will first discuss the tools used to implement the classification approach and the evaluation measures used.

The results will be discussed in three parts. First we will discuss the process of tuning the vocabulary of visual words. The best performing vocabulary found during these experiments was selected for use in the rest of the experiments. After that the results of classifying the keyframes using several machine learning algorithms will be discussed. Finally we will present the results of what we are truly interested in, classifying the television commercials as a whole. To produce these classifications the classifications of the keyframes will be combined for each commercial.

4.1 Implementing the approach

To implement our classification approach we used various tools and implementations that were readily available. By using these pre-existing pieces of software we were able to implement the desired classification approach relatively quickly. In this section we will briefly describe the main tools used to implement the classification approach. It is not a complete specification of all the code used, but should provide enough information to reproduce the classification approach if desired.

- The avi video fragments were converted to mpeg using Mencoder, a part of the Mplayer software package. This package is freely distributed under the GNU general public license version 2. The package can be downloaded at the following site: `http://www.mplayerhq.hu`.
- Following that the video stream was extracted from the mpeg fragments, shot boundaries were detected in the video stream and keyframes were extracted. This was done using mpegtools, a set of C tools containing various programs to manipulate and analyze MPEG files. These tools were developed by the Multimedian project N5 group (`http://monetdb.cwi.nl/projects/trecvid/MN5/index.php/Main_Page`). The mpegtools is available as open source software and can be found at: `https://sourceforge.net/projects/spiegle/`.
- An implementation of the SIFT keypoint detector was used to extract the keypoints from the keyframes extracted in the previous step. This implementation was acquired from the website of D. Lowe, the researcher who first proposed the SIFT feature approach. The necessary binary can be found at: `http://www.cs.ubc.ca/~lowe/keypoints/`.

- A clustermodel was trained to serve as the visual vocabulary. This clustermodel was trained using an implementation of the K-means clustering algorithm. The implementation is a part of the WEKA machine learning toolkit which can be found at: urlhttp://www.cs.waikato.ac.nz/ml/weka/.
- A simple Java program was used to apply the trained clustermodel to each keypoint descriptor of a keyframe and build the keyframe representation needed for the machine learning algorithm. This Java program uses the WEKA Java api to cluster each keypoint descriptor in turn and construct a histogram of the clustered keypoint descriptors. A complete listing of the program can be found in appendix A.
- The WEKA machine learning toolkit was once again used to train the actual classifiers. All machine learning algorithms employed were readily available in the standard distribution of the WEKA toolkit. For the first two experimentation phases the WEKA experimenter was used to run the desired cross validation. In the last phase the predictions of the classifier were outputted to file in order to aggregate them later.
- To aggregate the keyframe classification results another simple Java program was used to process the file containing the keyframe predictions. This program is listed in appendix B.

4.2 Performance measures

Before discussing the results themselves we'll first introduce the performance measures used to evaluate the results. The most obvious performance measure we used is the proportion of instances that have been correctly classified compared to the total number of instances. This measure can also be called the accuracy. Using only this evaluation measure however would be a bit too succinct. In order to present the results a bit more extensively we also used the precision and recall measures. The precision indicates the number of instances correctly classified as a certain class in proportion to all instances classified as that class. The recall is the proportion of instances correctly classified as a certain class to all instances truly of that class. Note that both the precision and the recall are calculated for each class individually. To obtain a single value for both the precision and recall the per class values are averaged.

$$
\begin{aligned}
Accuracy &= \frac{\sum_{i=1}^{n} |classified(c_i) \cap class(c_i)|}{|instances|} \\
Precision &= \frac{1}{n} \sum_{i=1}^{n} \frac{|classified(c_i) \cap class(c_i)|}{|classified(c_i)|} \\
Recall &= \frac{1}{n} \sum_{i=1}^{n} \frac{|classified(c_i) \cap class(c_i)|}{|class(c_i)|}
\end{aligned}
$$

The set $C = \{c_1, c_2, ..., c_n\}$ denotes the class labels used. $classified(c)$ is the collection of instances that have been classified as class c, whereas $class(c)$ is the collection of instances truly of class c. Finally $instances$ is the entire set of instances.

4.3 Tuning the visual vocabulary

As stated in chapter 2 the representation of the keyframes is based on a visual vocabulary that has been constructed using an unsupervised learning algorithm. The number of words in the vocabulary will affect the performance of the classification, which means that there is an optimal vocabulary (size) to be found. In order to find a good visual vocabulary we have constructed a few different vocabularies of different sizes and trained and tested a classifier based on each

vocabulary. Based on the results we have chosen the best performing vocabulary for further use in our experiments. Note that the chosen vocabulary will be the best amongst the vocabularies evaluated, not necessarily the optimal vocabulary. To find the optimal vocabulary we would at least need to plot the performance of the classification over all possible amounts of words. This is obviously not practically feasible.

To construct the visual vocabulary, a K-means clustermodel was trained on the keypoints extracted from the keyframes. In principle the clustermodel should be trained on the entire available collection of keypoints. In total we extracted around ten thousand keyframes from the video fragments in our data collection. Each keyframe in turn has on average 1433 keypoints associated to it. That means we had somewhere in the neighborhood of 14 million keypoints at our disposal. Training a cluster model on all these keypoints unfortunately is undoable without the help of a supercomputer so we used a fraction of the keypoints to train the clustermodel. The training set for the K-means clustermodel was constructed by randomly choosing fifty keyframes. These fifty keyframes had 59.730 keypoints associated to them.

Another question was which sizes to choose for the different vocabularies. According to [LA07] the size of a vocabulary used in the bag-of-keypoints approach typically ranges from 200 to 4000 different words. In our test collection of television commercials a maximum of 226 different classes were distinguished depending on which level of annotation we wished to classify. Intuitively it seems logical that we want at least as many 'visual' words as we have classes. Because of this we have tried vocabulary sizes of 250, 500, 1000 and 2000 different words.

To acquire results which could be evaluated we used the four different vocabularies to classify the keyframes. That means we used the vocabularies to construct four different representations for each keyframe and subsequently trained a classifier on the different representation versions. In this case we used a J48 decision tree [Qui93] as the classifier. We performed a 10 fold cross validation with 10 iterations for each vocabulary size. That means a total of 100 results for each vocabulary size. The results of these experiments can be found in table 4.1.

Table 4.1: *Results of varying the vocabulary size.*

Vocabulary size	**250**	500		1000		2000	
Accuracy	**0.3698**	0.3546	•	0.3409	•	0.3191	•
Precision	**0.3457**	0.2461		0.2475		0.2305	
Recall	**0.3893**	0.3036		0.2693		0.2690	

◦, • statistically significant improvement or degradation compared to 250 words vocabulary based on paired T-test.

The table shows that the best results are achieved with a vocabulary of 250 visual words. The differences are not very big, but we can see a steady decrease in performance as the vocabulary grows in size. When comparing the vocabulary of 250 words against the larger vocabularies using a paired T-test based on the accuracy, the vocabulary of 250 words consistently performs significantly better than the others. Unfortunately the 250 word vocabulary does not perform significantly better based on the precision and recall, even though it does score best on both measures. Considering these results we continued the rest of the experiments using the 250 words vocabulary. Using the smallest vocabulary had the added side benefit of taking less training time.

Table 4.2: *Time taken to train the classifier in minutes*

Vocabulary size	250	500	1000	2000
J48 Decisiontree	53.46	166.12	302.28	680.36

4.4 Selecting a machine learning algorithm

Having found a good vocabulary for the keyframe representations we focused on the tuning of the next part of the classification, the machine learning algorithm used to classify the keyframes. In the chapter 2 we have discussed several candidate algorithms that can be used for the classification. We tried to assess which type of algorithm would be most effective for our task, but unfortunately there was no clear theoretical winner. So once again the only method of effectively comparing the different options was a practical one.

Once again a full coverage of all possible options is practically infeasible. Not only are there simply too many machine learning algorithms to cover, there are also simply no working implementations for all algorithms readily available. Instead we have selected an algorithm representative for each of the algorithm types we have mentioned in the experimental approach as well as a few extra. The algorithms used are as follows:

- **ZeroR**
  The ZeroR algorithm is an extremely simple classification algorithm. It determines the mode class (the class occurring most often in the training set) and classifies each instance as belonging to this class. Obviously the results of this classifier are nowhere near adequate, but we can use them to set a baseline for the other algorithms.
- **OneR**
  The OneR is but one level more complex than the ZeroR algorithm. The OneR algorithm effectively constructs a decision tree with just a single level. A single attribute is used to determine the classification for an instance. The result of this algorithm will be a bit better than the ZeroR algorithm, but it is still only useful to set a baseline for the other algorithms.
- **J48**
  J48 is an implementation of a decision tree learner also known as the C4.5 algorithm. The decision tree is built recursively by selecting the attribute that provides the greatest information gain when chosen to split the set of instances. After the entire tree is constructed useless branches of the tree are pruned to reduce the complexity. For more information on this algorithm we refer to [Qui93].
- **PART**
  The PART algorithm is a decision list based algorithm. The decision list is constructed by building a partial C4.5 tree in each iteration and creating a rule based on the chain leading to the best leaf in the tree. More information on this algorithm can be found in [FW98].
- **Naive Bayes**
  This is an implementation of a naive Bayesian classifier. Naive Bayesian classifiers make the simplifying assumption that each attribute of an instance is statistically independent of all other attributes. Though this a pretty large assumption to make, naive Bayesian classifiers have performed quite well in the past on various learning tasks. For more information on naive Bayes classifiers see [JL95].
- **SMO**
  SMO is short for Sequential Minimal Optimization, an algorithm that trains a support vector classifier. As mentioned before, a support vector classifier can only distinguish between two different classes. For our multi class problem this is solved using pairwise classification. Simply put a support vector machine is trained for each class label distinguishing between being of that class and not being of that class. More information on the sequential minimal optimization algorithm can be found in [Pla98].
- **Multilayer perceptron**
  The final algorithm we tried using was the multilayer perceptron. A complex neural network

algorithm available in the WEKA toolkit. The neural network is trained using the back propagation method. In this method the output of the network for a training instance is calculated. Following that the error of the neural network compared to the proper output is determined. This error is then recursively propagated back into the network. The weight of a neural node is adjusted to diminish the error, after which the error is further propagated to the nodes connected to that node according to the weight by which they are connected to that node.

Table 4.3: *Results of training different machine learning algorithms.*

Algorithm	**ZeroR**	OneR	J48	PART	Naive Bayes	SMO
Accuracy	**0.0251**	0.0706 ◦	0.3698 ◦	0.3569 ◦	0.2353 ◦	0.6040 ◦
Precision	**0.0000**	0.0000	0.3457 ◦	0.3269 ◦	0.4262 ◦	0.5098 ◦
Recall	**0.0000**	0.0000	0.3893 ◦	0.3467 ◦	0.0830 ◦	0.6471 ◦

◦, • statistically significant improvement or degradation compared to the ZeroR algorithm based on a paired T-test.

The results table shows how the different algorithms have performed when trained with the keyframe representation based on the 250 words vocabulary. As before we have calculated an accuracy, precision and recall score for each algorithm. Unfortunately the results of the multilayer perceptron are missing. It turns out that training such a classifier on our problem takes an inordinate amount of time. This is probably due to the complex nature of the neural network that has to be trained for this classifier. We did not have access to the raw computation power needed to train this classifier in a reasonable amount of time, so the results for this algorithm are not available. Note that the zero scores of the ZeroR and OneR algorithm on precision and recall do not mean that they are truly zero, but rather that they are smaller than 0.0001.

Against the baseline of the ZeroR algorithm all the other algorithms managed to perform significantly better, but this is not surprising nor informative. What we really want to know is how the more serious algorithms perform in relation to each other. Again we use a paired T-test to compare the algorithms to each other. The results of these T-tests are summarized table 4.4.

Table 4.4: *Comparison of the machine learning algorithms using paired T-tests.*

	ZeroR	OneR	J48	PART	Naive Bayes	SMO
ZeroR		1/0/0	1/1/1	1/1/1	1/1/1	1/1/1
OneR	0/0/0		1/1/1	1/1/1	1/1/1	1/1/1
J48	0/0/0	0/0/0		0/0/0	0/1/0	1/1/1
PART	0/0/0	0/0/0	1/1/1		0/1/0	1/1/1
Naive Bayes	0/0/0	0/0/0	1/0/1	1/1/1		1/1/1
SMO	0/0/0	0/0/0	0/0/0	0/0/0	0/0/0	

The cell in the first row second columns indicates that the OneR algorithm performs significantly better than the ZeroR algorithm based on the accuracy, but not based on either the precision or recall. A zero indicates that the algorithm in the corresponding columns did not significantly perform better than the algorithm in the corresponding row.

The summary above shows that the support vector algorithm SMO performs significantly better than all the other algorithms. Reviewing the percentages of correctly classified instances shows that both the ZeroR and OneR algorithm score very low as expected from such simple baseline algorithms. After that comes the Naive Bayes algorithm. Interestingly enough the Naive Bayes algorithm performs relatively well on precision, but falls short on the recall. It is not surprising to see the precision and recall have such an inversely proportional relationship as this is quite often the case. However all other algorithms come up with precision and recall scores more in the

same neighborhood. The two algorithms J48 and PART score relatively similar, though as said J48 scores statistically significantly better than PART. This is not really surprising as the PART algorithm builds partial J48 decision trees to construct its decision list.

4.5 Classifying the commercials

In the previous experiments we presented the results obtained directly from directly training on the keyframes and classifying them. However, our goal is to classify the television commercials, not just single keyframes. A single television commercial usually consists of several keyframes. Using the classifications for the keyframes we can determine a classification for the commercial as a whole. To accomplish this we have trained a classifier using the 250 word vocabulary and the best performing machine learning algorithm, SMO.

In order to train the classifier we needed to split up the test collection in a part for training and a part for testing. However we could not split the collection entirely at random. That is to say, rather than splitting the collection on a per keyframe basis as before we needed to split the collection on a per commercial basis. After all, if we want to determine the classification for the commercials we need to make sure all keyframes of a commercial are put into the same part of the collection. To accomplish this we selected half of the available fragments for training and used the other half of the fragments for testing. This made sure that a single commercial was not split up between the training and test collection (commercials that spanned several fragments were ignored during the experiments).

It should be noted that due to the fact that there are quite some fragments containing multiple commercials this split of the collection is not a perfect fifty-fifty percent split of the commercials, but it is reasonably close. In the table below we can see a specification of the split into a training and test set. Besides specifying the number of keyframes in each set according to their commercial type, the table also shows the amount of distinct class labels and number of commercials present in each set according to their commercial type. As we can see the split is not perfect, but the numbers between the training set and test set are quite close. It should be noted that neither the training nor the test set contain all 227 different classes contained in the entire collection. That means there are some classes present in the test set but not in the training set and vice versa.

Table 4.5: *Statistics of the training and test collection split.*

Commercial type	Training set			Test set		
	#key-frames	#commercials	#distinct classes	#key-frames	#commercials	#distinct classes
brand	870	144	34	925	135	38
storechain	273	85	14	240	62	13
preview	994	108	27	898	90	23
charity	219	25	15	220	21	13
product	1889	274	78	2057	265	78
miscellaneous	270	30	18	287	35	18
total	4515	666	186	4627	608	183

The tables below show us three sets of accuracy, precision and recall scores. Besides presenting the scores for the entire test set, the tables also show the scores when only the commercials with classes belonging to one of the main commercial types are taken into account. Note that these per category scores are still based on the normal classifications, not the classifications of the commercial types. The first set contains the results obtained by simply classifying the keyframes using the classifiers as we did before in our experiments. These results are quite a bit worse than the results the SMO classifier obtained in the previous phase of experimentation. This can be

explained by the fact that the classifiers in the previous phase were trained on 90 percent of the available keyframes, while the classifier in this case was trained on around 50 percent (4627 of the 9142) of the keyframes.

Looking at the different categories we see that the charity commercials score lowest amongst the six commercial types. Considering that the group of charity commercials is the smallest of them all this is not so surprising. There simply is not as much training data available for this type of commercial. On the other hand the commercial type with the best scores is the store chain type, even though this is not the largest type group.

Table 4.6: *Classification results on the individual keyframes.*

	Accuracy	Precision	Recall
brand	0.3730	0.2918	0.2710
storechain	0.7292	0.6733	0.6337
preview	0.5646	0.2994	0.2850
charity	0.3455	0.1681	0.2139
product	0.4920	0.3878	0.3706
misc	0.5784	0.3159	0.2529
total	0.4930	0.3490	0.3305

The next set shows the scores obtained by aggregating the classification results of the keyframes to classifications for the commercials. In this case each keyframe classification was given equal weight. Basically this means that the final classification of the commercial is the majority vote amongst its keyframes. On the whole these scores are higher than the scores of the keyframe classification.

When looking at the scores of the different commercial types we see that the amount of improvement is not the seem for all types. The results of the brand and product commercials improve by quite a bit. The results for charity and miscellaneous commercials however do not seem to improve to a noticeable degree at all. It could be that the amount of improvement is related to the size of the of the groups in this case, but there is no way to tell for sure.

Table 4.7: *Results of classifying the commercials using majority voting.*

	Accuracy	Precision	Recall
brand	0.6296	0.4512	0.4358
storechain	0.8710	0.7124	0.6973
preview	0.6778	0.3498	0.3716
charity	0.3810	0.1553	0.2632
product	0.7208	0.5115	0.5288
misc	0.5714	0.3257	0.3472
total	0.6891	0.4398	0.4573

In the final set the keyframe classifications were weighted by the probability assigned to the predicted class label. The idea behind this is that is the classifier has assigned the class label to the keyframe with greater certainty it should be a better indicator of the class label of the entire commercial. Again we can see a clear improvement to all three evaluation measures when the keyframe classifications are aggregated. On the other hand there does not seem to be a marked difference between the two methods of aggregating the keyframes.

Table 4.8: *Results of classifying the commercials using weighted voting.*

	Accuracy	Precision	Recall
brand	0.6370	0.4792	0.4334
storechain	0.8710	0.7231	0.6900
preview	0.6667	0.4057	0.3818
charity	0.4286	0.1886	0.3158
product	0.7094	0.5032	0.5390
misc	0.6000	0.3815	0.3889
total	0.6875	0.4595	0.4708

Chapter 5

Conclusions

In the introduction we have explicitly posed two different questions regarding the classification of television commercials. Obviously in the conclusion of this report we want to see these two questions answered. To recapitulate, the two questions were:

1. Which techniques need to be used in order to classify television commercials?
2. How well can a television commercial classifier perform using available techniques?

We will summarize the answers we have found to these two questions in the next two sections. After that we will conclude the report by going over the possible directions of future research in order to improve the classification of television commercials and video fragments in general.

5.1 Method of the classification

In chapter 2 we have extensively discussed the different techniques needed to go from a raw video fragment to a classification of said fragment. It turns out this is not a simple matter of feeding the video fragment to a machine learning algorithm. The amount of data present in a video fragment is simply too vast to make such a naive approach possible. Using the available literature we have determined an approach in which the raw video data is reduced to a more manageable level in a number of consecutive steps. Only at the very end of this pipeline of techniques are the machine learning algorithms employed to train an actual classifier. In short the approach works as follows:

1. Each video fragment is divided into a number of shots using a histogram based shot boundary detection algorithm.
2. For each shot a keyframe is extracted by simply selecting the middle frame of the shot.
3. Low level SIFT descriptors are extracted from each keyframe.
4. The SIFT descriptors are clustered into a visual vocabulary using the K-means algorithm.
5. A histogram representation is constructed for each keyframe containing the descriptors.
6. A classifier is trained on the keyframe representation and used to classify the keyframes.
7. The keyframe classifications are aggregated over the commercials to classify them.

While we are quite confident we have employed a good approach we can not make the claim we have used the best possible approach. The classification is comprised of too many different steps, for all of which there are multiple options. For most steps there is no option known to be the best possible. The only way to optimize the classification approach is a long process of tweaking and testing the various available options. In the first parts of our experiments we have tried to optimize the classification approach by determining a good size for the vocabulary of visual 'words' and comparing a number of different machine learning algorithms. However these optimization efforts are incomplete as exhaustively testing the full range of possibilities is practically infeasible.

5.1.1 Optimizing the visual vocabulary

The results of optimizing the visual 'words' vocabulary size have shown that the smallest tested vocabulary of just 250 words performs best. Logically speaking that means the optimal vocabulary size lies somewhere in between 1 and 250 words, but there is no way to be sure which size it is without testing all possibilities in between 1 and 250. [WCM05] suggests that a small vocabulary can be constructed from a large initial vocabulary without a significant loss in performance. The constructed small vocabulary in this paper contains only a few hundred 'words'.

The question is why a larger vocabulary is not necessarily better than a small one. The answer can be found in [WCM05]. If two different visual 'words' do not help discriminate between classes there is no point to distinguishing between them. For example if word A and B are always an indication of the same class X, there is no reason to distinguish between them for our classification. Thus we can get the same performance with less 'words'. However that does not explain why in our experiments the smaller vocabulary was not just as good as the larger vocabularies, but actually outperformed them. A possible explanation for this is that when using more 'visual' words the differences in representations between images of the same class will be larger. One image of a class could have a histogram of 'visual' words far different from another image of the same class. As a result the machine learning algorithm will have a harder time learning a general representation for all the instances of a class. The end result of this is a loss in performance when using larger vocabularies.

Another question that we unfortunately cannot answer concerns the possible effect of choosing the training set for the construction of the visual vocabulary. As mentioned in chapter 4 we only used a fraction of the total number of available keypoints to train the clustermodel for the visual vocabulary. If we had chosen a different fraction of the keypoints we could have very well had different results. Unfortunately we do not have the necessary data to retrace from which keyframes the keypoints were taken to train the clustermodel. Because of this there is no way to tell how the chosen training set has affected the results.

5.1.2 Selecting a good machine learning algorithm

In our comparison of machine learning algorithms the support vector classifier clearly outperformed all the other algorithms we tried. This is not surprising when looking at the ZeroR and OneR algorithms which are obviously overly simplistic baseline algorithms. However the support vector classifier also performs much better than the decision tree and Bayesian network classifiers. This result supports the fact that in many image annotation papers support vector machines are used as the machine learning algorithm of choice [PD07] [DWF+04] [PDCB06]. Unfortunately there is no clear theoretical explanation as to why the support vector machine solution performs so much better than the other algorithms. As stated in the experimental approach chapter a support vector machine finds the optimal separating hyperplane in the feature space in order to make the classification between two classes. Apparently the commercials can relatively effectively be separated using such a linear hyperplane for the various classes. We could validate this notion by

analyzing the data set ourselves regarding the linear separability of the commercials from a certain class versus all the other commercials. This would take quite some time though. Furthermore we could analyze a dataset beforehand regarding the linear separability of the classes to assess the appropriateness of a support vector machine, but this would kind of defeat the point of using a machine learning algorithm to learn the classification.

5.2 Performance of the classification

In the last phase of our experimentation we managed to produce a classifier that was able to classify 68 percent of all commercials correctly after training on almost half of the commercials. The accuracy on a per keyframe basis in this experiment was only around 49 percent. This is a far better result than should be expected by randomly choosing one of the 227 different classes, but it is by no means a perfect score. In a practical system we would need the classification accuracy to be close to 100 percent before we would call it dependable.

Making a comparison between our experiments and the results presented in literature articles is not exactly easily done. There are almost no articles on image and or video classification that use exactly the same data collection, training/test set split and evaluation measures. Because of this, making a direct comparison of the results is practically impossible. Nevertheless we would still like to make a general assessment of our classifier against other image classifiers presented in literature. [DWF+04] [XZ06] [WCM05] all present results for tasks quite similar to our task of classifying keyframes.

- The approach taken in [DWF+04] is the most like our own approach. It uses a visual vocabulary as well as a support vector machine as its classifier. The classification accuracy during a 10-fold cross validation in this paper is quite a bit higher than ours (85% vs 60%). However their test collection contains a mere 7 different classes.
- As said before the approach presented in [WCM05] tries to optimize the efficiency of the visual vocabulary by reducing a large initial vocabulary to a small vocabulary similar in size to our own 250 word visual vocabulary. For the rest their approach is quite similar to our own. The results presented in this paper are based on a 50 percent training/test set split. Once again their accuracy exceeds our own accuracy achieved when the collection was split in a 50/50 fashion. However their collection also contains just a few classes (9) and furthermore the objects in their test collection are segmented from the background which reduces the noise of the input data.
- Like the approach in the previous paper, the approach taken in [XZ06] optimizes an initial visual vocabulary to achieve better results. Again the performance measures presented in this paper are noticeably better than our own. Both the precision and recall obtained during a 5-fold cross validation exceed our own precision and recall scores. On the other hand their test collection once again contains a lot less classes than our test collection, only 25 classes are distinguished.

At first sight looking at the comparisons our classifier does not perform too good. Fortunately there are a few arguments to be made in our defense. First of all there is the fact that our test collection with 227 different classes is quite a bit more complex than the test collections used in the other papers. Furthermore we can argue that the objective of the other papers is to classify objects actually present in the images. In our case on the other hand we wish to classify the images to the subject of the corresponding commercial. Such a classification is arguably on a higher level of abstraction.

Another thing that should be mentioned are duplicates in the data collection we used. In chapter 3 we mentioned that the data collection consisted of commercials recorder in a five day period. Seeing as commercials are often repeated it should not be surprising that there are quite some duplicates in our collection. Unfortunately we can not give a precise number for the amount of duplicates. In total there were 1275 commercials in our data collection and 227 different class labels. So there are roughly 5 commercials of the same class on average. Most of these can probably be considered near duplicates of each other. With so many duplicates our research can be compared to the work of [KSH04]. However, the goal of that article is to simply detect the near-duplicates. We on the other hand also strived to classify the near-duplicates with the right labels.

All in all we consider our classifier to be reasonably good. Which of course does not mean there is no more room for improvement.

5.3 Further research

It should come as no surprise that there are plenty of directions for further research on the classification of television commercials and videos in general. So rather than exhaustively mentioning every possible avenue of further research we will identify a number of more interesting directions based on the findings of our own research.

5.3.1 Lack of comparisons

While there is plenty of literature available on image and video classification there is a distinct lack of a clear standard to be extracted from all this literature. The literature gives no clear idea of which techniques to use when we want to analyze images or videos. Furthermore there is no real standard for the evaluation of image and video analysis systems. Most articles use their own data collections, which makes making comparisons hard if not downright impossible.

5.3.2 Optimizing the visual vocabulary

During our experiments we built several different visual vocabularies using the K-means clustering algorithm. However this is not exactly the smartest way of constructing a visual vocabulary. We have already mentioned a paper [WCM05] that suggests a more intelligent way of constructing a visual vocabulary. It starts by building an initial large size vocabulary. After this follows an iterative process of merging clusters that are not discriminative amongst each other. The resulting vocabulary is far smaller than the initial vocabulary yet performs almost just as well. [XZ06] also suggest a method of optimizing the visual vocabulary. In this case the optimized vocabulary is reported to actually outperform the non optimized vocabulary. Even more suggestions for building a better visual vocabulary can be found in [PDCB06] [FSMST05] [HC05]. Employing one of these methods for building a better visual vocabulary could very well lead to better performance and efficiency.

5.3.3 Analyzing additional data streams

During the start of this project we decided not to look at the audio stream of the commercials in order to classify them. The reason for this was the practical concern that we simply did not have the necessary time to sufficiently handle this extra data stream. For future research however it would be quite interesting to see the added value of analyzing the audio stream as well as the

video stream. Besides using a combination of the video and audio stream for the classification it would also be informative to see a comparison of using either the video and audio stream exclusively. Another option of adding another type of data to the classification input is to employ optical character recognition techniques. A lot of commercials contain some form of text. If this text could be accurately extracted from the video stream it might be very beneficial for the classification process. The optical character recognition technique would have to be robust to a wide array of different font styles though.

5.3.4 Using object classification prior to commercial classification

In the previous section we mentioned that we directly used the keyframe representations to classify the commercials. SIFT features however have been developed initially to recognize objects. Taking this into account we might consider first recognizing objects in the keyframes and subsequently classifying the objects to a label for the commercials. To take full advantage of this additional step of recognizing objects we would need an additional data collection containing clear pictures of a large amount of different objects. Besides this practical concern there is the question if this extra step would have sufficient merits. Adding another step to an already complex process of classification would be wasteful unless the added step significantly improves the results. There is probably no telling whether this would be the case without actually trying it out.

Bibliography

[BR96] J.S. Boreczky and L.A. Rowe. Comparison of video shot boundary detection techniques. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 170–179, 1996.

[DLW05] R. Datta, J. Li, and J.Z. Wang. Content-based image retrieval - approaches and trends of the new age. In *Proceedings of the International Workshop on Multimedia Information Retrieval*, pages 253–262, Singapore, November 2005.

[DWF+04] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, May 2004.

[FML04] S.L. Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1002–1009, July 2004.

[FSMST05] J. Farquhar, S. Szedmak, H. Meng, and J. Shawe-Taylor. Improving bag-of-keypoints image categorisation. technical report, University of Southampton, 2005.

[FW98] E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. In J. Shavlik, editor, *Fifteenth International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

[GD05] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the IEEE International Conference on Computer Vision*, volume II, pages 1458–1465, 2005.

[HC05] W.H. Hsu and S.-F. Chaug. Visual cue cluster construction via information bottleneck principle and kernel density estimation. In *Proceedings of the international Conference on Image and Video Retrieval*, 2005.

[JL95] G.H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–45, San Mateo, 1995. Morgan Kaufmann.

[KC01] I. Koprinska and S. Carrato. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16(5):477–500, 2001.

[KSH04] Ke.Y, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, 2004.

[LA07] N. Lazic and P. Aarabi. Importance of feature locations in bag-of-words image classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages I641–I644, 2007.

[LFM04] V. Lavrenko, S.L. Feng, and R. Manmatha. Statistical models for automatic video annotation and retrieval. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 17–21, Montreal QC, Canada, May 2004.

[Low04] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LSP05] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[LW06] J. Li and J.Z. Wang. Real-time computerized annotation of pictures. In *Proceedings of the ACM Multimedia Conference*, pages 911–920, Santa Barbara, California, October 2006.

[LZLM07] Y. Liu, D. Zhang, G. Lu, and W. . Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[MM04] D. Metzler and R. Manmatha. An inference network approach to image retrieval. In *Proceedings of the International Conference on Image and Video Retrieval*, pages 42–50, Dublin City, July 2004.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[NLM99] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[PD07] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[PDCB06] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. In *Lecture Notes in Computer Science*, volume 3954 LNCS, pages 464–475, 2006.

[Pla98] J. Platt. Machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[PMDK06] G.Th. Papadopoulos, V. Mezaris, S. Dasiopoulou, and I. Kompatsiaris. Semantic image analysis using a learning approach and spatial context. *Semantic Multimedia*, 4306:199–211, 2006.

[Qui93] J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[RHC99] Y. Rui, T.S. Huang, and S-F. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[RTG00] Y. Rubner, C. Tomasi, and L. J. Guibas. Earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[SWS+00] A. Smeulders, M. Worring, S. Santini, A. Gupta, , and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transcripts on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[VS06] J. VanDeWeijer and C. Schmid. Coloring local feature extraction. In *Lecture Notes in Computer Science*, volume 3952 LNCS, pages 334–348, 2006.

[WCM05] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of the IEEE International Conference on Computer Vision*, volume II, pages 1800–1807, 2005.

[XZ06] F. Xu and Y.-. Zhang. Feature selection for image categorization. In *Lecture Notes in Computer Science*, volume 3852 LNCS, pages 653–662, 2006.

[ZI06] Q. Zhang and E. Izquierdo. A bayesian network approach to multi-feature based image retrieval. *Semantic Multimedia*, 4306:138–147, 2006.

[ZMLS07] J. Zhang, M. Marsza?ek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007.

[ZT01] L. Zhang and T. Tretter. An overview of video abstraction techniques. technical report HPL-2001-191, HP Laboratory, 2001.

Appendix A

KeypointClusterer.java

```
1
2  import java.io.*;
3  import weka.core.*;
4  import java.util.*;
5  import java.io.BufferedWriter;
6  import java.io.FileWriter;
7  import weka.clusterers.*;
8  import weka.core.converters.*;
9
10 /**
11  * Clusters all the keypoints in the given directory and builds histograms representing the
12  * keyframes from the clustered keypoints. The histograms are saved to an arff for each
13  * video fragment.
14  * @author Tom Nijmeijer
15  *
16  */
17 public class KeypointClusterer {
18
19   public static void main(String[] args) {
20     //Check if the proper number of arguments has been specified
21     if(args.length != 3) {
22       System.out.println("Usage: KeypointClusterer <directory> <cluster model>
23 ␣␣␣␣␣␣␣<destination file>");
24       System.exit(1);
25     }
26
27     //Check if the first argument specifies a valid directory
28     File rootdir = new File(args[0]);
29     if(!rootdir.isDirectory()) {
30       System.out.println("Please specify an existing directory to cluster.");
31       System.exit(1);
32     }
33
34     try {
35       //Create the clusterer to be used from the saved model
36       ObjectInputStream ois = new ObjectInputStream(new FileInputStream(args[1]));
37       Clusterer clusterer = (Clusterer) ois.readObject();
38       ois.close();
39
40       //Create an empty instances object with the proper
```

```
41        //headers based on the loaded clustermodel
42        int numClusters = clusterer.numberOfClusters();
43        FastVector atts = new FastVector();
44        atts.addElement(new Attribute("framenumber"));
45        for(int i = 0; i < numClusters; i++) {
46          atts.addElement(new Attribute("cluster"+(i+1)));
47        }
48        Instances header = new Instances("Keyframe",atts,0);
49
50        //Create an array containing all subdirectories of the directory specified
51        File[] subdirs = rootdir.listFiles(new FileFilter() {
52          public boolean accept(File file) {
53            return file.isDirectory();
54          }
55        });
56
57        //For each subdirectory do:
58        for(int i = 0; i < subdirs.length; i++) {
59          File subdir = subdirs[i];
60          System.out.print("Processing subdirectory: "+subdir.getName()+"....");
61
62          //Create an array containing all arff files in this subdirectory
63          //Each arff file contains the keypoint descriptors of a keyframe
64          File[] arffs = subdir.listFiles(new FileFilter() {
65            public boolean accept(File file) {
66              return file.getName().endsWith(".arff");
67            }
68          });
69
70          //Create an instances object in which to put the keyframe representations
71          //of the current video fragment being processed denoted by the current
72          //subdirectory
73          Instances result = new Instances(header, arffs.length);
74
75          //For each arff file do:
76          for(int j = 0; j < arffs.length; j++) {
77            //Load the arff file as an instances object
78            File arff = arffs[j];
79            FileReader reader = new FileReader(arff);
80            Instances instances = new Instances(reader);
81
82            //Construct a new instance in which to tally the clustered keypoints
83            Instance newInstance = new Instance(1, new double[numClusters+1]);
84
85            //Extract the frame number from the filename
86            String framenum = arff.getName();
87            framenum = framenum.replaceAll("\\.key\\.arff", "");
88            framenum = framenum.replaceAll("frame_(0*)", "");
89
90            try {
91              int frameno = Integer.parseInt(framenum);
92              newInstance.setValue(0,frameno);
93            } catch(NumberFormatException e) {
94              System.out.println("Badly formatted filename, skipping...");
95            }
96
97            //Cluster each instance and add 1 to the proper attribute in the result instance
98            for(int k = 0; k < instances.numInstances(); k++) {
```

```
99              try {
100               int cluster = clusterer.clusterInstance(instances.instance(k));
101               newInstance.setValue(cluster+1, newInstance.value(cluster+1)+1);
102             } catch(Exception e) {
103               System.out.println("Failed to cluster a keypoint, skipping...");
104             }
105           }
106
107           //Add the keyframe's histogram to the instances object of the video fragment
108           result.add(newInstance);
109         }
110
111         //Save the arff-file of the current video fragment
112         String destFile = subdir.getPath()+"/"+args[2];
113         BufferedWriter writer = new BufferedWriter(new FileWriter(destFile));
114         writer.write(result.toString());
115         writer.flush();
116         writer.close();
117
118         System.out.println("Done!");
119       }
120     } catch(IOException e) {
121       System.out.println("Failed to create a necessary reader or writer.");
122       System.exit(1);
123     } catch(ClassNotFoundException e) {
124       System.out.println("Failed to load the specified clustermodel.");
125       System.exit(1);
126     } catch(Exception e) {
127       System.out.println("Failed to read the number of clusters in the loaded clustermodel.");
128       System.exit(1);
129     }
130
131     System.out.println("All Done.... "+new Date().toString());
132   }
133 }
```

Appendix B

KeyframeAggregator.java

```
1
2  import java.io.*;
3  import java.util.regex.*;
4  import java.util.*;
5
6  /**
7   * Aggregates the predictions of the keyframes to predictions for the commercials
8   * @author Tom Nijmeijer
9   *
10  */
11 public class KeyframeAggregator {
12   private static String currentclass;
13   private static HashMap counts;
14
15   /**
16    * Writes the prediction for the current commercial to the standard output.
17    */
18   public static void writeCommercialPrediction() {
19     Set keys = counts.keySet();
20     Iterator iter = keys.iterator();
21     String majorityvote = "";
22     double majoritycount = 0;
23
24     while(iter.hasNext()) {
25       String current = (String)iter.next();
26       Double currentcount = (Double)counts.get(current);
27       if(currentcount.doubleValue() > majoritycount) {
28         majoritycount = currentcount.doubleValue();
29         majorityvote  = current;
30       }
31     }
32
33     System.out.println(majorityvote+"#"+currentclass);
34   }
35
36   public static void main(String[] args) {
37     //Check if the minimum number of arguments has been specified
38     if(args.length < 2) {
39       System.out.println("Usage: KeyframeAggregator <file> <useprobabilities>");
40       System.exit(1);
```

```
41      }
42
43      //Check if the first argument specifies a valid file
44      File infile = new File(args[0]);
45      if(!infile.exists()) {
46        System.out.println("Please specify an existing file");
47        System.exit(1);
48      }
49
50      //Cast the second argument to a boolean
51      boolean useprobs = Boolean.parseBoolean(args[1]);
52
53      //Open a writer to which the cleaned up keyframe predictions will be written
54      //if this has been specified using a third argument
55      BufferedWriter output = null;
56      if(args.length > 2) {
57        try {
58          output = new BufferedWriter(new FileWriter(args[2]));
59        } catch (Exception e) {
60          System.out.println("Failed to open the output file.");
61          System.exit(1);
62        }
63      }
64
65      //Create the reader needed to read the predictions
66      BufferedReader input = null;
67      try {
68        input = new BufferedReader(new FileReader(infile));
69      } catch (Exception e) {
70        System.out.println("Failed to open the input file.");
71        System.exit(1);
72      }
73
74      currentclass = "";
75      counts = new HashMap();
76
77      String line;
78      Pattern regex = Pattern.compile("^(\\d+)\\s(.+)\\s(0\\.[0-9]+)\\s(.+)\\s$");
79      try {
80        //Read each line and extract the variables from it using a regular expression
81        while((line = input.readLine()) != null) {
82          Matcher match = regex.matcher(line);
83          match.find();
84
85          int instancenum = Integer.parseInt(match.group(1));
86          String prediction  = match.group(2);
87          double probability = Double.parseDouble(match.group(3));
88          String trueclass   = match.group(4);
89          trueclass = trueclass.replaceAll("\'","");
90
91          //When a new true classification is encountered the current commercial
92          //has ended. Write the prediction for the current commercial and start
93          //the aggregation for the new commercial.
94          if(!trueclass.equals(currentclass)) {
95
96            if(!currentclass.equals("")) {
97              writeCommercialPrediction();
98            }
```

```
99
100             //Start a new hashmap if we are starting a new commercial
101             counts = new HashMap();
102             currentclass = trueclass;
103           }
104
105           //If the current prediction already exists in the hashmap counting
106           //the votes increase the vote for the prediction accordingly
107           if(counts.containsKey(prediction)) {
108             Double oldcount = (Double)counts.get(prediction);
109             Double newcount;
110             if(useprobs)
111               newcount = new Double(oldcount.doubleValue()+probability);
112             else
113               newcount = new Double(oldcount.doubleValue()+1);
114             counts.put(prediction, newcount);
115           }
116           //Otherwise add a new entry for the new prediction and instantiate
117           //it with the proper vote weight
118           else {
119             if(useprobs)
120               counts.put(prediction, new Double(probability));
121             else
122               counts.put(prediction, new Double(1));
123           }
124
125           //Write the stripped down keyframe prediction to an output file
126           //if it has been specified in the arguments
127           if(output != null) {
128             output.write(prediction+"#"+trueclass+"\n");
129           }
130         }
131
132         //Write the last commercial prediction to the standard output
133         if(!currentclass.equals("")) {
134           writeCommercialPrediction();
135         }
136
137       } catch(IOException e) {
138         e.printStackTrace();
139       } catch(NumberFormatException e) {
140         e.printStackTrace();
141       }
142
143       if(output != null) {
144         try {
145           output.flush();
146           output.close();
147         } catch(Exception e) {}
148       }
149     }
150 }
```