2012-07-30

Subspace-based Semantic Concept Detection and Retrieval for Multimedia Information Systems

Chao Chen
University of Miami, andrew.matrix.chen@gmail.com

Recommended Citation

Chen, Chao, "Subspace-based Semantic Concept Detection and Retrieval for Multimedia Information Systems" (2012). *Open Access Dissertations*. 833.
https://scholarlyrepository.miami.edu/oa_dissertations/833

UNIVERSITY OF MIAMI

SUBSPACE-BASED SEMANTIC CONCEPT DETECTION AND RETRIEVAL FOR MULTIMEDIA INFORMATION SYSTEMS

By

Chao Chen

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

August 2012

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

SUBSPACE-BASED SEMANTIC CONCEPT DETECTION AND RETRIEVAL FOR MULTIMEDIA INFORMATION SYSTEMS

Chao Chen

Approved:

Mei-Ling Shyu, Ph.D.
Associate Professor of Electrical
and Computer Engineering

M. Brian Blake, Ph.D.
Dean of the Graduate School

Mohamed Abdel-Mottaleb, Ph.D.
Professor of Electrical and Computer
Engineering

Michael Wang, Ph.D.
Professor of Electrical and Computer
Engineering

Xiaodong Cai, Ph.D.
Associate Professor of Electrical
and Computer Engineering

Shu-Ching Chen, Ph.D.
Professor of School of Computing
and Information Sciences
Florida International University

CHEN, CHAO
(Ph.D., Electrical and Computer Engineering)
(August 2012)

Subspace-based Semantic Concept Detection and Retrieval for Multimedia Information Systems

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Mei-Ling Shyu
No. of pages in text. (170)

The prevalence of digital recording devices, the cheap cost of data storage as well as the convenience provided by the widely accessible Internet have created the demand to retrieve information according to users' requests from multimedia data sources. However, the multimedia information retrieval task has several challenges that need to be addressed, such as bridging the semantic gap, modeling from imbalanced data sets, and utilizing inter-concept relationships to enhance the retrieval performance of an individual concept.

To respond to the challenge of bridging the semantic gap, subspace modeling methods are proposed to address this issue as a classification task. The proposed subspace modeling methods construct a principal component (PC) subspace for each class, where the PCs are derived from the instances belonging to that class. The PCs are selected and ranked based on Fisher's criterion to reduce the searching effort and an iterative searching is utilized to determine the best PC set. Subspace modeling methods are proposed in this dissertation, including multi-class subspace modeling (MSM), binary-class subspace modeling (BSM), and subspace modeling on global and local structures (SMGL). Comparative experiments show that MSM, BSM, and SMGL can outperform some other well-known algorithms on a number of benchmark data sets.

To address the data imbalance challenge, two clustering-based subspace modeling methods called clustering-based subspace modeling (CLU-SUMO) and class selection and clustering-based subspace modeling (CSC-SUMO) are proposed. *K*-means clustering and/or semantic concept labels are used to partition the majority class (usually the negative class) into several groups, each of which is merged with the minority class (usually the positive class) to form a

much more balanced subset of the original data set. Then, the subspace model learned from the original data set is integrated with all the subspace models learned from the balanced subsets to form a classification framework. The experimental results on news and broadcast video data sets support the claim that the proposed CLU-SUMO and CSC-SUMO render better classification performance than some existing techniques that are commonly used to handle the data imbalance problem.

Finally, two ranking strategies that consider the inter-concept relationships are proposed to enhance the retrieval performance from the classification models of a target concept. The co-occurrence class between the target concept and the reference concept is generated and multiple corresponding analysis (MCA) is adopted to capture the correlation between the feature-value pairs (a partition of the attribute values) and one co-occurrence class *PP* (a class consisting of the instances containing both target concept and reference concept). Such correlation information is used to refine the ranking results from the classification models of the target concept to provide the final ranking scores. The effectiveness of all ranking strategies are attested by the experimental results on public news and broadcast video data sets as well as some image data sets, which demonstrates that the performance of the retrieval results is significantly improved after the proposed ranking strategies are applied.

Acknowledgments

First, I would like to express my appreciation to my advisor, Dr. Mei-Ling Shyu, for her excellent supervision and guidance on my research work. Her training style makes me improve myself and achieve great progress.

Also, my sincere gratitude is extended to Dr. Shu-Ching Chen from Florida International University. His determination, guidance and support direct me on my research work and the development of my personality.

In addition, I want to thank Dr. Mohamed Abdel-Mottaleb for his patience in answering so many questions from me in his pattern recognition classes. I enjoy the in-class interaction with you during my PhD study. Moreover, many thanks go to Dr. Michael Wang and Dr. Xiaodong Cai from the Department of Electrical and Computer Engineering at University of Miami for being my dissertation committee members.

I thank Dr. Walter G. Secada and Dr. Mary Avalos from the School of Education, University of Miami, and Dr. Claudia Schmid, Dr. Silvia Garzoli, Ms. Reyna Sabina from the Physical Oceanography Division, National Oceanic and Atmospheric Administration for their financial support.

Thanks to all my friends, my colleagues and particularly my TRECVID teammates in the Data Mining, Database, and Multimedia Research Group at University of Miami and the Distributed Multimedia Information Systems Laboratory at Florida International University whom I met during my study at the University of Miami.

Finally, I want to give my special gratitude to my parents and my girlfriend, Xiaochen Cai for their support and I would like to devote this dissertation to you.

CHAO CHEN

University of Miami

August 2012

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Multimedia Information Retrieval

Plain texts used to be a primary way to record and distribute information, but people prefer vivid images or videos and the accompanying audio, because they can provide a better experience for receiving and digesting new information. However, compared with text, multimedia data such as images, audio, and videos usually require much larger disk spaces for storage. In the past decades, these images, videos, and audio served mainly as the auxiliary information to the plain text to help understand the contents. Now, with the advance of Internet techniques, the extensive application of compression approaches, the cheap cost of data storage, and the prevailence of digital devices such as videocassette recorders and digital cameras, multimedia information is generated at an explosive speed and used in many applications. There is an urgent demand to retrieve the multimedia information from such a large multimedia data collection.

A common way to retrieve multimedia information is to search by the keywords. For example, the videos on Youtube are often accompanied with a brief description of the uploaded content or other text information given by the users. Each image on Flickr is always attached with a number of user-defined tags. These text data enable the retrieval of multimedia information via keywords, a very similar way to information retrieval from text documents. However, such a keyword-based information retrieval method may face the following difficulties.

- *No text information available*: Videos and images are often provided without any text data. Therefore, it requires huge human efforts to add related text information to the videos and images to make keyword-based retrieval applicable. Usually, such a labor-intensive effort is rather costly in terms of money and time.
- *Noisy and incomplete keywords*: Some text information provided by the users are irrelevant to the content of the images and videos. For example, an image that depicts a scene of autumn might have these irrelevant tags such as “beautiful”, “so cool”, and “photoshot?”. The irrelevant text information could compromise the effectiveness of the adopted keyword-based retrieval algorithm. In addition, it is often hard and even impossible to fully describe all the content within the videos and images. However, such an incomplete representation of the content could also compromise the retrieval effectiveness and lead to unsatisfactory retrieval results.

In response to the aforementioned difficulties of keyword-based information retrieval, content-based information retrieval methods were proposed [1, 2]. In content-based information retrieval, the contents of images and videos are described by low-level features, such as color, texture, shape, and other information directly extracted from the images and videos. However, these low-level feature-based approaches often fail to meet the users’ demands to retrieve high-level semantic information because of the so-called semantic gap issue [3] (to be discussed in Chapter 1.2). To narrow the gap between the high-level semantic content information and the low-level feature representation, semantic concept descriptors were proposed to serve as the intermediate features to assist video retrieval [4, 5]. Some semantic concept detectors have shown, by experiments, to provide a satisfactory detection accuracy to those concepts that are related to people (face, anchor), location (outdoor, vegetation), object (car/truck/bus), event (sports event), and others [6]. Hauptmann et al. [7] further reported that a few thousands of such semantic concepts could be enough to ensure accurate video retrieval. However, despite the early success of applying visual content descriptors to aid semantic information retrieval, there are still several challenges to be addressed.

- *How to bridge the semantic gap between semantic concepts and low-level features*?
- *How to handle the data imbalanced problem when building models to map low-level features to semantic concepts*?
- *How to achieve effective semantic concept detection and retrieval* ?

1.2 Semantic Gap

Generally speaking, the semantic gap is the disconnection between the digitalized data that can be recognized by the modern computer systems and the conceptual understanding in the minds of human beings. Smeulders et al. [3] described the semantic gap as “the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation”. For a given image containing a target concept (such as “tree”), human beings are able to interpret both the low-level information (such as color, shape, and texture) and the high-level target concept (such as “trees”) and other surrounding objects (such as “sky” and “house”). Computers can interpret low-level features in the binary presentation. However, it is hard for computers to recognize the target concept (tree) from binary digits, though it seems not to be a problem at all from humans’ perspectives. Such an inconsistency between the low-level descriptors (like color, shape, texture, and so on) and high-level semantic concepts (like “tree”, “sky”, “house”, and the like) leads to the well-known “semantic gap”.

One way to bridge the semantic gap between the low-level features and the semantic concepts is to make use of the techniques from the computer vision domain. Since humans usually describe the semantic contents via keywords, some research work proposed keyword-based approaches to learn the mapping between the objects in the images and a list of keywords. Duygulu et al. [8] proposed a machine translation model to translate the segmented image regions into a few keywords. A two-stage linking method was proposed by Kutics et al. [9] to first map the low-level features to the related keywords and then assign these keywords to the images in the second stage. Unlike region-based methods, Fan et al. [10] proposed an object-based approach that used salient objects for image content representation to achieve a mid-level understanding of the image content. They further proposed a hierarchical classification framework [11] in which the salient objects are used for image content representation and feature extraction. However, the region-based and object-based approaches rely on image segmentation techniques that are rather sensitive to illumination and sophisticated backgrounds. On the other hand, image-based approaches showed their merits: the features were extracted from the whole image and did not require any image segmentation. Recently, the bag-of-words representation of local visual feature descriptors (such as scale-invariant feature transform (SIFT) [12] features) has shown to render promising performance in image and video retrieval [13, 14].

There are also research works focusing on relevance feedback [15, 16, 17]. Users judge the results returned by the retrieval system according to a given user query. The relevant results judged by the users are used to refine the learning models. It may take a number of feedback rounds for the retrieval system to render satisfactory retrieval results. Relevant feedback is often integrated with the learning methods to provide an interactive way for the end users to further refine the retrieval performance of a content-based retrieval system. However, the weakness of relevant feedback is that it is sometimes inconvenient or even impossible to get the feedback from the users. In addition, the first-round retrieval is rather important for an incremental refinement of the retrieved results [18].

Data mining is another way to address the semantic gap issue. Data mining techniques include association rule mining, classification, clustering, and etc. [19]. Lin et al. [20] proposed an association rule mining method to generate and select association rules for semantic concept detection. He et al. [21] fused visual features and keywords for web image retrieval based on multi-model semantic association rules.

Clustering methods served as an unsupervised way to handle the semantic gap issue. Clustering-based methods such as the one used by Chen et al. [22] assumed that “images of the same semantics are similar in a way, images of different semantics are different in their own ways”. This assumption matches the idea of clustering that the intra-cluster dissimilarity should be small while the inter-cluster dissimilarity should be large. However, it is not uncommon to encounter such a situation in the real world that there is a large variance within the same semantics.

Apart from association rule mining and clustering, classification methods are used as mainstream approaches to tackling semantic gap. The classification methods have some relationships with clustering since those unsupervised classification methods actually perform the clustering job. However, one major difference between classification and clustering lies in the aspect that classification, especially supervised classification, requires user-provided labels to learn the models to bridge the semantic gap. Semi-supervised classification is close to unsupervised classification since both are based on the same prior assumption that data points near each other probably share the same class labels. If unlabeled data violate the assumption of semi-supervised learning, then the accuracy of the learning models will be degraded.

Recently, graph-based semi-supervised classification methods have gained much attention [23, 24]. In graphical models, images or video shots are treated as nodes and the similarity values (measured by different distance metrics) between these nodes are regarded as edges to construct a graph. Later, a graphical model is built and optimized on the graph by minimizing the regularized cost function [25]. One problem with the graphical model is measuring the distance between two nodes when constructing graphs. Therefore, a good distance metric plays a key role in the success of graph-based semi-supervised classification. Unfortunately, it is very difficult to justify in advance that a distance metric is appropriate for a semi-supervised learning task.

Supervised classification algorithms build the learning models from labeled training data. Although they require more effort to get the training labels, the classification accuracy is usually better than semi-supervised or unsupervised classification methods. A supervised classification algorithm called RankSVM [26] was used by [18] to bridge the semantic gap to facilitate medical image retrieval. Neural networks, Bayesian networks, and decision tree can also be found in [27, 28, 29] to participate in building semantic detection or retrieval models. In this dissertation, subspace modeling methods are proposed which also belong to supervised classification. The proposed subspace modeling methods can achieve dimension reduction and also provide competitive classification performance. The details related to subspace modeling will be elaborated in Chapter 3.2.

1.3 Data Imbalance

Data imbalance is commonly seen in multimedia information retrieval. For example, when building the models to bridge the semantic gap between low-level features and high-level semantics, the training instances pertaining to a target semantic concept (or called positive instances) are only a small proportion of the whole training set, while the training instances that do not contain the target semantic concept (or called negative instances) dominate the training set, forming an imbalanced data distribution. In this case, the target class (or called positive class, which is composed of the positive instances) is the minority class while the majority class is the non-target class (or called negative class). Please note that the data imbalance issue related to multiple classes is outside the scope of this dissertation. The imbalance ratio is measured by the size of majority class to that of the minority class and it is not uncommon to see such a ratio with 1000 : 1 or even 10000 : 1.

He et al. [30] categorizes the data imbalance problems into two cases: imbalance with rare instances and relative imbalance. The former refers to the case when the number of minority class instances is limited. Such a limited number of minority class instances makes model learning difficult because of the poor representation of the minority class. The other case, relative imbalance, is quite often seen in real-world applications. Usually, the number of the instances belonging to the minority class can reach certain levels, but such a number is still relatively small compared to the majority class. However, it is still possible to learn an accurate model for a minority class under the circumstance of relative imbalance.

Most of the popular classifiers such as support vector machines (SVM), decision tree, and neural networks are built on the assumption that the training data set is balanced. However, this assumption may not be true in real cases. Thus, the learning models suffer from a data imbalance problem and may not be able to produce accurate prediction results [31]. Without any strategy to handle the imbalance problem, these classification algorithms tend to predict all data instances as the members of the majority class since the learning is biased towards the majority class. For example, within a data set where 0.1% of the data instances are the positive instances and 99.9% are the negative instances, all instances are probably predicted to be negative since such misclassification of positive instances only produces a tiny prediction error (probably the minimum prediction error). Therefore, these important positive instances will never be correctly predicted in such a case. However, usually these positive instances are much more important than the negative ones.

There are a number of ways to handle the data imbalance issue. Broadly, they can be categorized into three types. The first type focuses on manipulating the data to achieve data balance. Resampling methods are of this type. Resampling can be further divided into oversampling and undersampling. Oversampling increases the size of the minority class by either replicating existing instances or synthetically generating new instances that belong to the minority class. Undersampling decreases the size of the majority class by discarding the instances belonging to the majority class. The second type views the trouble caused by the imbalanced data as a model quality problem, where the models learned from the imbalanced data are considered as poor and weak. Therefore, the second type aims to find a solution to "re-weight" and combine these weak models to achieve an improved model. The representative methods of the second type are various boosting and bagging methods. The last type relies on the cost function within the learning model. It is obvious that the cost to misclassify an instance of the minority class is higher than to misclassify an instance that belongs to the majority class. Therefore, some classification models take into consideration such inequivalent costs during the model training step, preventing the minority class from being dominated by the majority class. The representatives of the last type are cost-sensitive learning algorithms [32, 33].

To address the data imbalance problem when building subspace models, two clustering-based subspace modeling (CLUstering-based SUbspace MOdeling (CLU-SUMO) and Class Selection and Clustering-based SUbspace MOdeling (CSC-SUMO)) are proposed in this dissertation to build classification models on the imbalanced data set. The proposed methods generate a number of balanced data subsets to which the subspace modeling methods are applied. In this way, these subspace modeling methods built on the data subsets are used to refine the classification results given by the subspace models learned from the original imbalanced data set. The whole procedure will be presented in Chapter 3.3.

1.4 Semantic Concept Detection and Retrieval

A simple semantic concept detection and retrieval framework for multimedia data set may have three major components, namely a preprocessing component, a modeling component, and a postprocessing component. The functionality of the preprocessing component is to prepare the data for the modeling component. There are a few research directions involved in the preprocessing component. One research direction is to explore discriminant features that can be extracted from the content of images or videos. Evolved from the traditional features such as color histogram, shape, and texture to modern features like local binary patterns (LBP) [34], histograms of oriented gradients (HOG) [35], and scale-invariant feature transform (SIFT) [12], the discriminant ability of features keeps on increasing and pulls up the retrieval performance. However, it is still difficult to find a set of features that are always effective for all semantic concepts.

Most of the extracted features have different scales. To prevent the small-scale attributes from being dominated by the large-scale attributes, a common method is to apply data normalization. There are many normalization functions proposed to scale the data within a common range. For example, min-max normalization function can ensure the normalized data locate in the range of $[0,1]$. Equation (1.1) to Equation (1.3) show a list of normalization functions [36], where X_{min} is the minimum value of X, X_{max} is the maximum value of X, μ is the estimated mean of X, and σ is the estimated standard deviation of X.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}; \tag{1.1}$$

$$X' = \frac{X - \mu}{\sigma}; \tag{1.2}$$

$$X' = 0.5[tanh(\frac{0.01 \cdot (X - \mu)}{\sigma}) + 1]. \tag{1.3}$$

Attribute selection, sometimes called feature selection, is also widely adopted in the preprocessing component to find a set of relevant attributes with regard to a target concept. The direct benefit from the attribute selection method is that the learning model can be induced faster since the dimensionality of the attributes has been reduced. However, a trade-off must be made between the number of attributes retained and the performance of the learning model. Indeed, attribute selection could enhance the learning model by removing irrelevant or redundant attributes. However, the removal of too many attributes leads the learning models to be over-fitting to the training data and thus lacking generalization.

In addition to the aforementioned research directions, discretization also attracts many researchers. Some learning methods (such as those based on correspondence analysis) are designed to cope with nominal data. Therefore, sometimes it requires discretization algorithms to convert the data from a numeric representation to nominal values (such a conversion may facilitate the learning process [37]). Discretization could reduce the number of possible values of attributes and simplify the representation of the data. In general, discretization methods fall into two categories: one consists of unsupervised methods such as equal-width and equal-frequency discretization; the other is formed based on a group of supervised methods such as entropy-based information entropy maximization (IEM) and statistics-based chi-square.

The modeling component utilizes the classification algorithms to detect the target semantic concept in the data instances and render the corresponding ranking scores. The score value of each data instance reveals its relevance to the target concept. In the information retrieval area, there are plenty of learning algorithms to choose, including RankSVM [26], RankBoost [38], RankNet [39], and etc. However, most of them are pair-wised learning methods, which encounter serious challenges from data storage and model training time when dealing with a large-scale multimedia data set. Therefore, in semantic concept retrieval, the most prevailing modeling algorithms are based on support vector machines that are able to balance between model accuracy and training time. In addition, some papers also reported good results by employing logistic regression, maximal figure-of-merit (MFoM) [40], and other learning schemes. Furthermore, to overcome the incapability of learning ability from one kernel, multiple kernel learning (MKL) is proposed in [41]. However, the improvement of the performance by MKL is accompanied with an increase of learning time and computational cost. Therefore, it remains an explorative task to find a modeling method that is able to achieve a harmonic balance between modeling efficiency and modeling performance.

Although for some target concepts like "person" and "outdoor" the modeling component is sufficient to render satisfactory retrieved results, many other concepts, such as "dancing" and "adult", still call for improvements in the retrieved results [42]. Therefore, the third component (postprocessing) is often adopted to refine the retrieved results. A postprocessing step can provide a better ranking of the retrieval results based on the ranking scores from the modeling component by considering additional auxiliary information. For example, some semantic concepts may be difficult to retrieve, resulting in the modeling component rendering poor retrieval performance. However, by considering the inter-concept relationships, such as the ontology of semantic concepts, it is possible to significantly increase the retrieval performance for these concepts. The ontology describes the relationship among a groups of concepts and it is widely used to facilitate video and image annotation and retrieval [43, 44]. To remove the dependence on domain knowledge, automatic rule learning has been proposed in [45, 46, 47]. However, these methods are subjective to the quality of each detector and may hurt the annotation and retrieval performance after the inter-concept relationship is considered [46]. In this dissertation, MCA-based ranking strategies are proposed to use the inter-concept co-occurrence to refine the retreival results from the classification models. The proposed ranking strategies are explained in details in Chapter 3.4.

1.5 Contributions and Limitations

The contributions of the dissertation are listed as follows.

- *Subspace modeling methods to bridge the semantic gap are proposed.* In this dissertation, subspace modeling algorithms are proposed to address the semantic gap issue. The proposed methods are built on the principal component subspace in which the idea of anomaly detection is introduced to generate rules for classification purposes. Anomaly detection focuses on detecting the abnormal data patterns that deviate significantly from a normal pattern. Usually it is applied in the area of intrusion detection. To the best of my knowledge, few people have tried to use this idea to solve the multi-class classification problem and apply the idea to the field of semantic concept detection. Unlike some existing algorithms such as SVM that build a separation margin for different classes, the proposed subspace modeling methods try to capture the overall data distribution of the data instance to perform the classification job. Furthermore, the proposed methods are facilitated with dimension reduction as well as noisy removal functionalities. The PC selection method helps to reduce the dimension and build robust models, rendering competitive classification performance in terms of accuracy and F1 score. Finally, the subspace modeling methods are shown to be effectively deployed into the task of semantic concept detection to bridge the semantic gap.

- *Novel subspace modeling-based classification frameworks to handle the data imbalance problem are proposed.* Compared with the resampling methods, the proposed methods do not lose any data instances or produce extra minority instances. They integrate both the global subspace models built on the original data set with the local subspace models trained on the balanced subsets of the training data to handle the data imbalance problem. Therefore, it avoids the problems of resampling, such as losing critical instances belonging to the majority class and overfitting to the minority class. Compared with the boosting method, the proposed methods do not require a time-consuming "re-weighting" time. Therefore, they are more efficient than the boosting methods. In addition, compared with cost-sensitive classifiers, the proposed subspace modeling methods do not require defining any cost matrix which often requires substantial tuning efforts or domain knowledge. However, they can achieve equivalent effects by automatically assigning weights to the majority class within the subspace models.

- *New ranking strategies utilizing inter-concept relationships are proposed.* Two new ranking strategies are proposed to retrieve the desired target concept effectively by utilizing the inter-concept relationships. Compared with the ontology-based methods, they require less or no domain knowledge, and the generated rules are more specific than those from the WordNet-based methods. Furthermore, the proposed ranking strategies utilize the inter-concept relationships between semantic concepts on the attribute-level, while the other rule-based learning methods are built on a higher level (such as the model level) which may suffer from poor performance of the learning models.

The limitations of the proposed frameworks and methods are summarized below:

- Currently, the proposed subspace modeling methods are only shown to be effective for numeric data and they prefer those classes with Gaussian distribution. Therefore, it cannot be directly applied to nominal data set since the mean or standard deviation of a nominal attribute does not make sense.
- The time complexity for deriving principal components and eigenvalues from singular value decomposition is $O(N^2)$ in the training phase. Therefore, the training phase is suggested to be done off-line.
- It requires some empirical experience to select the parameter K in CLU-SUMO and CSC-SUMO. When deployed in a larger scale data set, the K-means clustering method requires an efficient implementation and a suitable K value.
- The selection of reference concepts in the first proposed ranking strategy requires domain knowledge. However, such domain knowledge is no longer required for the second ranking strategies. But the second strategies need to consider all possible combinations between the target concept and one or more reference concepts.

1.6 Organization

The remainder of this dissertation is organized as follows. Chapter 2 reviews the literature regarding the supervised classification methods, the approaches to handle data imbalance, and the approaches to enhance the retrieval results by inter-concept relationships. Chapter 3 introduces the design of the overall framework briefly and elaborates the details of the proposed methods to address the three challenges mentioned in this chapter. Chapter 4 shows the design and implementation of a prototype of Web-based semantic concept retrieval system, and Chapter 5 concludes the dissertation and proposes some directions for future work.

Chapter 2

Literature Review

In multimedia information retrieval, supervised classification algorithms serve to bridge the semantic gap between low-level features and high-level semantic concepts. A list of commonly used supervised classification approaches in multimedia research are reviewed in Chapter 2.1. The imbalanced data distribution within the multimedia data may negatively impact the performance of these supervised learning approaches. Thus, diverse approaches to handling data imbalance problem are illustrated and discussed in Chapter 2.2. Finally, Chapter 2.3 investigates the approaches that can further improve the performance of the retrieval results by considering the inter-concept relationships.

2.1 Supervised Classification Approaches in Multimedia Information Retrieval

Support Vector Machine (SVM) [48] is one of the most widely used classification algorithms in multimedia information retrieval, which generates hyper-planes that are capable of separating the data instances of different classes in a multidimensional space. However, the application of SVM on the original space is not satisfactory since there are too many non-separable cases. Fortunately, according to Mercer's theorem [49], even if the patterns are non-separable in the original space, through a nonlinear transformation, the patterns would be separable if the dimension after transformation is high enough.

Based on this theorem, many studies have focused on the study of the kernels of SVM [50, 51]. There are three standard kernels: linear, polynomial, and Gaussian. However, these kernels are quite generic and do not closely relate to the data. For this reason, the Fisher kernel [52] and Kullback-Leibler divergence-based kernels [53] are developed to improve SVM and are utilized in areas such as audio and image processing. For image and video retrieval, the chi-square kernel recently gained much attention with its high performance [54]. The chi-square kernel [55] was shown to perform better than the kernel of histogram intersection and normalized scalar products. SVM's advantages were summarized by [56] as 1) the utilization of the kernels, 2) the absence of local minima, 3) the sparseness of the solution, and 4) the capacity control obtained by optimizing the margin. Despite these advantages and successful deployment in many application areas, SVM has the following drawbacks: 1) the selection of SVM kernel parameters is not entirely solved; 2) high algorithm complexity and extensive memory requirement limit its usage in large scale data sets; and 3) it may have problems when processing the discretized data, as indicated by Suykens et al. [57].

Nearest neighbor (NN) method has been studied thoroughly by the researchers for decades. This is a lazy learning method that does not require building a model prior to classification. Moreover, nearest neighbor method has strong adaptability to different datasets. An extended form, K-nearest neighbor (KNN), is regarded as optimal in the sense of Bayes error rate, assuming the sampled size approximates infinity [58]. KNN used to perform inferiorly to SVM, but Vincent et al. [59] proposed a modified KNN called HKNN that claimed to perform as well as or even better than SVMs. Zhang et al. [60] proposed a hybrid framework which took advantage of the merits from both SVM and KNN. Their idea was to apply SVM locally to those closely related neighbors and the hybrid framework. It turned out to have a good tradeoff between time-complexity and classification performance. Since NN and KNN perform the classification task during the run-time, the classification process is time-consuming, especially when the training set is large. Besides, irrelevant or redundant attributes may compromise the performance of NN and KNN by biasing the similarity between instances.

Decision tree algorithms are also powerful learning methods. The decision tree algorithms build tree-like models to classify the data instances by their attributes. $C4.5$ [61] was the most well-known decision tree algorithm. It considered both the model training speed and error rate. The improved version, called $EC4.5$, came out soon after it prevailed. Decision tree algorithms often suffer from the overfitting problem. To avoid this problem, pruning strategies were usually deployed [62]. Decision tree algorithms were applied to a number of fields, such as semantic concept detection and video event classification. Chen et al. [63] combined decision tree with other multi-modal analysis to extract goal events. During the tree generation steps, the attributes were split by utilizing information gain. Snoek et al. [64] used $C4.5$ in their framework and compared $C4.5$ decision tree with maximum entropy as well as SVM for automatic classification of semantic events. Decision tree algorithms work well with highly relevant attributes, but may encounter problems when many irrelevant and interactive attributes are present.

Rule-based classifiers are applied to areas like video indexing [65], event detection [66], and etc. Zhou et al. [65] developed a rule-based classification system for basketball video indexing. The PRISM method was utilized by [66] to generate rules. One kind of the rule-based classification methods called associative classification became popular recently. It contains two stages in which the first stage fulfills rule generation and the second stage selects the most important rules. Associative classification was utilized to detect semantic concepts by [67] based on the correlation between feature-value pairs and concepts. Kobylinski et al. [68] used the occurrence count and spatial proximity when building a customized classifier to accurately classify the images. Repeated incremental pruning to produce error reduction (RIPPER) [69] is another rule-based learning algorithm, which is an optimized version of IREP (incremental reduced error pruning) [70, 71]. RIPPER constructs a rule set that covers all positive instances in order to fit the training data as close as possible. With regard to a large date set, while the error rate of RIPPER is the same as or slightly better than the $C4.5$ decision tree, RIPPER is more efficient than $C4.5$ when handling large and noisy data sets. However, like decision tree, many rule-based methods are sensitive to irrelevant attributes [69].

Neural networks have been popular in the machine learning society. They simulate a real biological neural network to build a mathematical model. There are many types of neural networks, such as feed-forward neural networks, radial basis function (RBF) networks, Kohonen self-organizing networks, recurrent networks, and others. Among them, feed-forward neural networks and RBF networks are commonly used. Multilayer perceptron (MP) is one kind of feed-forward neural network models, which consists of many learning techniques. Backward propagation (BP) [72] is a learning scheme that is most commonly used in MP. The error of the BP neural networks propagates from the output layer back to the previous layer and updates the weights of the hidden layer to minimize the output errors. It is shown by [73] that one hidden layer of perceptrons were adequate as universal approximations of any nonlinear functions. As for RBF networks, it is a linear combination of radial basis functions. Compared with MP, it can avoid suffering from the local minimization problem. It can also get the same approximation capability as MP. Therefore, a three-layer RBF network (only one hidden layer) is widely employed in practice. Neural networks have the advantages of detecting the implicit relationship between dependent or independent variables and offering accurate prediction performance. However, the black-box structure makes it hard to understand the classification rules. Furthermore, neural networks require long training time and heavy computation burdens when dealing with large-scale multimedia data sets. In addition, noisy training data and irrelevant attributes may also have a negative influence on the classification accuracy, resulting in problems such as overfitting.

To improve the classification accuracy, boosting methods such as AdaBoost are often used in conjunction with the classification algorithms. The classic AdaBoost was introduced by Freund and Schapire [74] to deal with binary classes. The boosting methods keep calling the base learning algorithms repeatedly in a number of rounds to find a suitable series of weights for the training set. Also, some extended AdaBoost methods, such as AdaBoost.M1, AdaBoost.OC [75], and AdaBoost.ECC [76], can be used to solve the multi-class problems. AdaBoost's major shortcoming lies in its long training time, as it utilizes an iterative method to decide the weight of each training instance and every iteration requires a process of classification. Thus, for learning algorithms such as neural networks, the AdaBoost methods will further increase the time for building the training model. Moreover, since AdaBoost is designed for improving the performance of weak learning methods, strong learning algorithms such as SVM benefit little from AdaBoost.

The aforementioned classification algorithms have a common problem of suffering from irrelevant attributes when building the models. To address this problem, dimension reduction techniques such as principal component analysis (PCA) are often applied, as a PC subspace is good at not only depicting the overall characteristics of data but also achieving the goal of dimensionality reduction. A common way to achieve dimension reduction in a PC subspace is to discard a proportion of minority PCs [77]. Vaswani et al. [78] proposed a principal component null space classification approach for image and video classification, which constructs a PC subspace on a reduced set of original PCs based on energy loss during the PCA transformation. One PC-based classifier, called collateral representative subspace projection modeling (CRSPM), was introduced by [79] to the multimedia applications. CRSPM consists of two modules - a classification module and an ambiguity solver module. The idea of CRSPM, which differentiates a normal class from the abnormal classes based on their chi-square distance, can be further extended to detect semantic concepts by regarding a concept class as a normal class and the other classes as abnormal ones.

One problem with these aforementioned PC-related subspace methods is that the advantages of the construction of appropriate PC subspaces are often underestimated. However, a good PC subspace could be vital to a satisfactory performance of subspace classification. In this dissertation, a novel subspace-based classification framework called multi-class subspace modeling (MSM) and its variation binary-class subspace modeling (BSM) are proposed. MSM and BSM determine their subspaces by choosing an optimized set of parameters to reach satisfactory classification accuracy. Fisher's criterion here helps to select the representative PCs on which a pair of training and testing principal component classifiers (PCCs) are built. In addition to utilizing the global dissimilarity as classification rules like MSM and BSM, a new framework called subspace modeling using global and local structure (SMGL) is proposed to integrate both the global dissimilarity and the local similarity to perform classification. The detailed descriptions of MSM, BSM, and SMGL can be found in Chapter 3.2.

2.2 Approaches to Handling Data Imbalance

There are a number of techniques to address the data imbalance issue. Data sampling is a common technique to learn from an imbalanced data set. The idea of sampling is to adjust the ratio between the positive instances and the negative instances that are used for training the classification models by reducing the number of negative (or majority) class instances and/or by increasing the number of positive (or minority) class instances. Therefore, data sampling can be further divided into oversampling and undersampling [80]. Oversampling aims to add more positive class instances to the original imbalanced data set so that the number of the positive class instances is comparable with the number of the negative class instances. New positive class instances can be generated either by simply replicating existing positive class instances (called random oversampling) or by synthetic sampling of the positive class instances (called synthetic minority oversampling technique (SMOTE)) [81]. The understanding of oversampling is quite straightforward: to balance the ratio between positive class and negative class without losing any information related to the instances of both the positive and the negative classes. However, random oversampling by replicating the positive class instances from the original data set could lead to an overfitting problem [82]. The other oversampling methods like SMOTE generate each synthetic positive class instance $X^{(new)}$ between two existing data instances, as shown in Equation (2.1).

$$X^{(new)} = (1-\varepsilon) \cdot X^{(i)} + \varepsilon \cdot \hat{X}^{(i)}, \tag{2.1}$$

where $X^{(i)}$ is an arbitrary positive class instance and $\hat{X}^{(i)}$ is randomly picked from the K-nearest neighbors of $X^{(i)}$. ε is a random variable between 0 and 1. Like random interpolation, it is reasonable to assume that there is an instance that lies between two existing instances if they are close to each other. However, over-generalization seems to be a big problem for SMOTE. Therefore, some adaptive synthetic sampling algorithms [83, 84] were proposed to consider the information about neighboring data instances, such as their class labels.

In contrast to oversampling, undersampling balances the ratio between the positive class and the negative class by removing the instances belonging to the negative (majority) class. The way that undersampling tries to balance the data set is quite simple and sometimes it is effective. Cieslak et al. [85] proposed a framework that finds an optimal sampling level for each meaningful region by identifying via a segmentation method based on the Hellinger distance. However, some important negative instances that represent the characteristics of the negative class could be discarded during the undersampling process and the training model could thus be compromised.

Data sampling methods directly manipulate data instances by either increasing or reducing the size of a specified class. Boosting methods handle the data imbalance issue in another way. The boosting methods acknowledge that the training models built from an imbalanced data set might be not good. However, an appropriate "re-weighting" of these weak training models can lead to a good classification result. Boosting methods combine weak learning models to reduce the negative influence caused by the data imbalance problem. Among them, AdaBoost [86] is a representative boosting algorithm. AdaBoost algorithm reweighs the training data instances and models iteratively during the training phase by minimizing the prediction errors produced by an ensemble of training models. In the classification phase, the class label of each test instance is determined by a voting of these weighted ensemble models. AdaBoost is proved to be effective in many real applications. However, the major drawback of AdaBoost as well as other boosting methods is that they usually require a time-consuming iterative process to find the optimal weights for the ensemble models.

There are also algorithms that integrate both boosting methods and sampling methods. For example, SMOTEBoost [87] was built by combining SMOTE and the AdaBoost.M2 algorithm. SMOTEBoost interactively uses SMOTE at each boosting step and the learning of the positive (minority) class is gradually strengthened and emphasized during iterative steps. Another example, the DataBoost-IM [88] method, aims to utilize a boosting procedure to ensure the predictive accuracy of the positive class and the negative class are both satisfactory. To prevent training models from overfitting, JOUS-Boost was proposed using the AdaBoost algorithm together with over/under-sampling and jittering of the training data [82].

Another category of approaches to address the data imbalance problem are cost-sensitive learning methods [33, 89, 90, 91]. In these methods, the costs associated with the misclassification of the positive class instances and the negative class instances are different. In the case where the positive class is dominated by the negative class, the misclassification of the positive instances should be given a larger cost than misclassifying the negative instances. Studies [32, 33] show that cost sensitive learning is able to render better performance than the sampling methods. Algorithms like cost-sensitive decision tree and cost-sensitive neural networks are well studied. However, cost-sensitive learning can also be integrated with the other classifiers. One of the problems of cost-sensitive learning is the configuration of the cost matrix. Although it is clear that misclassifying a minority class instance should be given a larger cost, a question arises when it comes to determining how much larger the cost value should be. Therefore, it is a challenging task to find a suitable cost matrix for cost-sensitive learning methods when they are used in an imbalanced data set.

In response to the challenges and difficulties of the aforementioned approaches, a clustering-based binary-class classification framework was proposed to address the data imbalance issue [92]. In this method, the original training data set is first divided into a positive class subset and a negative class subset, where the positive class subset is composed of all the training data instances containing the target concept and the negative class subset consists of the remaining data instances. In the original imbalanced data set, where the negative class dominates the positive class, the negative class subset is further divided into many negative groups by either clustering or holding out some other non-target concept classes within the original negative class subset so that the ratio of the data sizes between each negative group and the positive class subset is not large. Therefore, the data groups generated by combining each negative group and positive class subset do not suffer from the data imbalance issue. For each balanced group, subspace models are trained and optimized. Finally, the subspace models trained on all data groups are integrated with the subspace model built on the original imbalanced data set to form an integrated model. The integrated model is supposed to render better classification performance than the subspace model trained on the original data set alone. As implied by the proposed framework, local subspace models are built on balanced subsets of the original data so as to take advantages of the merit of under-sampling. Moreover, the utilization of a global subspace model ensures that no information will be lost. Within each subspace modeling, the weight parameter applied to the majority class prevents the majority class from dominating the minority class, which is equivalent to assigning a higher cost to misclassifying a minority class instance. Finally, an integration of the global subspace model and local subspace model by "re-weighing" their ranking scores avoids the time-consuming boosting step, though it may require domain knowledge to decide some of the parameters.

2.3 Performance Enhancement by Inter-concept Relationships

Wei et al. [43] proposed an ontology-enriched semantic space and an ontology-enriched orthogonal semantic space to facilitate the selection of concept detectors for video searching. These ontology-based methods usually build a knowledge base on WordNet, in which the nodes denote the semantic concepts and the relationships between concepts are represented as the edges. Wang et al. [93] summarized the limitations of WordNet as: 1) the knowledge between semantic concepts is too generic and lack of the flexibility to describe the actual relationships within a specific multimedia data collection; and 2)it is unable to handle lexicons out of WordNet.

Furthermore, other methods utilizing the inter-concept relationships to enhance performance of the semantic concept retrieval have also been proposed. Liu et al. [45] proposed an association and temporal rule mining method to infer the presence of the semantic concepts from inter-concept co-occurrence, reporting enhanced semantic concept detection accuracy. There are also graphical models [46, 47] built to capture the inter-concept relationships or utilize the consistency of semantic concepts to improve the annotation results. Qi et al. [94] utilized a correlative multi-label (CML) framework to model the correlations between concepts with strong interactivity. For some concepts, their work reported more than 10% improvement in terms of average precision (AP). Aytar et al. [95] presented a video retrieval framework using semantic word similarity and visual co-occurrence. The context between concepts was exploited by point-wise mutual information. The visual co-occurrence relations between concepts were also obtained. Evaluating the concepts from TRECVID 2006 and 2007 data sets, the semantic retrieval results performed 81% better than those of the text-based retrieval methods. Yan et al. [46] proposed a probabilistic graphic model to mine the relationship between video concepts. In their work, a set of concepts were grouped together to learn a multi-concept relation model via a probabilistic graphic model. Their paper reported that some concepts had benefited from the multi-concept relation model; while others could render worse performance than that of the baseline method. Their pioneer work actually implied that although such a multi-concept relation model showed encouraging results, there still existed "gold" undermining the relationship between concepts that required a deeper discovery. Jiang et al. [47] proposed an impressive approach called the domain adaptive semantic diffusion method (DASD) that utilized the consistency between semantic concepts to improve the annotation results. DASD treated the concepts as nodes and the concept affinities as the weights of the edges and thus built a semantic graph model to capture the relationships between concepts. Their experiments on TRECVID 2007 data sets [42] reported a 6.3% performance gain over the baseline method by using DASD. It revealed that such an inter-concept relationship is potentially significant for an effective concept retrieval framework.

The difference between the proposed framework and the previous work lies in the following aspects. First, some previous work regards correlation information as mutually useful. In other words, it considers concept "A" and concept "B" as both target and reference concepts to each other under the assumption that concept "A" and concept "B" would both benefit from their correlation. However, this may not be true in reality since the difficulty to retrieve concepts "A" and "B" may not be the same. For example, although the concepts "road" and "outdoor" have a strong correlation, "road" is much more difficult to retrieve than "outdoor" [4]. Therefore, "road" may benefit from such a correlation from "outdoor" because the correlation information from "outdoor" is quite reliable. Unfortunately, on the other side, "outdoor" may render worse performance if it utilizes the correlation information with "road". Therefore, in the proposed work, the correlation information is utilized uni-directionally. In addition, only those easy-to-retrieve concepts are regarded as the reference concepts and of which the relationship will be used to refine the ranking of the retrieved results of the target concepts. Second, the information of co-occurrence between concepts is viewed in a mutual manner in the previous work. That is, only when concept "A" and concept "B" both appear frequently does the relationship between "A" and "B" become valuable. However, this co-occurrence between concepts is viewed here in an individual manner. As long as there is a large chance (e.g., 90%) that "B" will occur when "A" appears, this co-occurrence relationship from “B” is valuable and should be taken into consideration, no matter how low the chance of “A” would appear when “B” occurs. It is worth mentioning that previous work may miss co-occurrence relationship that is not mutual, such as the co-occurrence relationship between “snow” and “outdoor”. Finally, previous work studied the correlation between concepts on the concept level, and the inter-concept relationships were derived from the class labels. However, the proposed frameworks further explore such inter-concept relationships on the attribute level (details to be explained in Chapter 3.4).

Chapter 3

The Proposed Semantic Concept Detection and Retrieval Framework

The overall design of the proposed semantic concept detection and retrieval framework is illustrated in Chapter 3.1. It aims to tackle the three challenges mentioned in Chapter 1 and responds to the challenges as follows. The semantic gap issue is regarded as a classification problem. In this chapter, a multi-class subspace modeling method [96] is proposed in Chapter 3.2.1 and its variation (a binary-class subspace modeling method) is proposed in Chapter 3.2.2 to address binary-class multimedia data sets. The subspace modeling method that integrates both global dissimilarity and local similarity is proposed in Chapter 3.2.3. Considering the data imbalance issue as well as the characteristics of the subspace modeling method, a clustering-based subspace modeling (CLU-SUMO) method is proposed in Chapter 3.3.1 which is able to build robust models from imbalanced data sets. Later, class selection and clustering-based subspace modeling (CSC-SUMO) method is proposed in Chapter 3.3.2 to integrate class selection with CLU-SUMO and reduce the cost of clustering as well as the time for training models. To address the last challenge, inter-concept relationships are taken into consideration to refine the semantic retrieval results from subspace models and semantic concept retrieval frameworks that utilize co-occurrence relationships between semantic concepts are proposed in Chapter 3.4.

3.1 Overview of Proposed Semantic Concept Detection and Retrieval Framework

The overview of the proposed semantic concept retrieval framework is shown in Figure 3.1. The framework consists of a preprocessing component and one semantic concept detection and retrieval component.

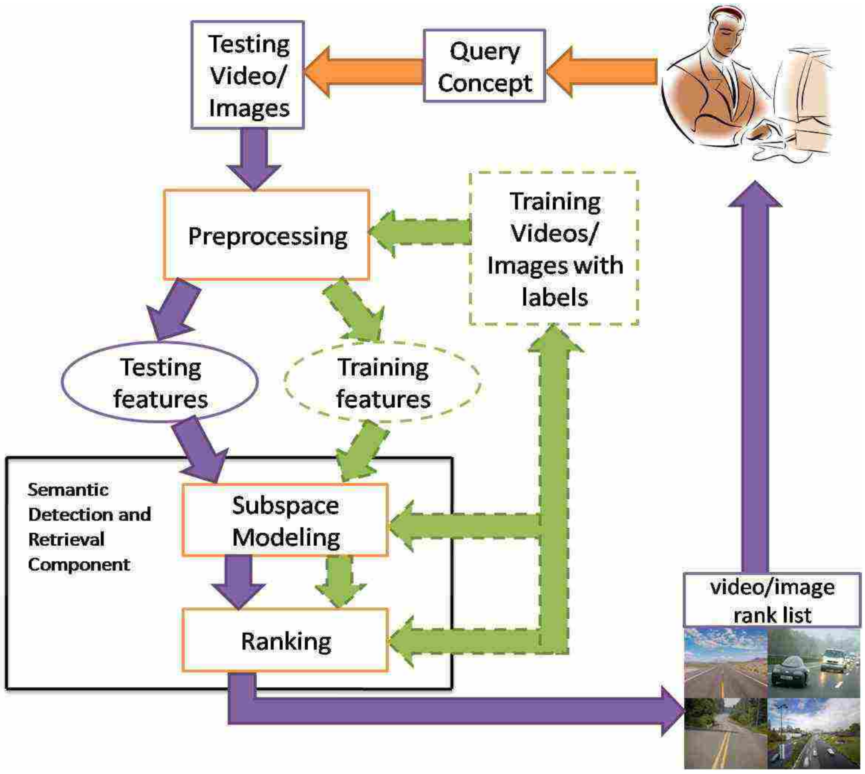


Figure 3.1: The overview of the proposed semantic concept detection and retrieval framework

The preprocessing component performs the role of extracting low-level features from the video contents. Shot boundary detection methods may be applied to segment each video into a sequence of shots, which are the basic units within a video. Two kinds of features could be extracted from videos: key frame-based features and shot-based features. Before extracting key frame-based features, key frame detection methods [97] can be employed to generate a collection of key frames to summarize the video contents. In the set of key frame-based features, the following features could be extracted: color dominant in RGB color space, color histogram in HSV space, color moment in YCbCr space, edge histogram, texture co-occurrence, texture wavelet, Tamura texture, Gabor texture, and local binary patterns. In addition, some mid-level features from face detection can also be captured, such as the number of faces. In the set of shot-based features, audio features and shot-based visual features are extracted. The extracted audio feature set is composed of volume-related, energy-related, and spectrum-flux-related features as well as the average zero crossing rates. As for shot-based visual features, the grass ratio is calculated as a mid-level feature, which is useful for detecting sports-related semantics such as soccer players and sports. Furthermore, a set of motion intensity estimation features are extracted, such as the center-to-corner pixel change ratio. Both of the categories of features target at representing/summarizing the videos.

Video representation is a difficult but helpful task in multimedia research. However, it is not the focus of this dissertation. The key frame-based features are commonly used to describe the video content and show that integrating both global and local key frame-based features is effective in the multimedia retrieval task.

However, key frame-based features cannot capture audio information and some shot-level information may be missing. Therefore, it needs shot-based features to serve as an auxiliary information source. Z-score normalization (shown in Equation (1.2)) is applied to the extracted key frame-based and/or shot-based features to prevent large attributes dominating the small attributes. Then, chi-square feature selection is adopted to select a list of relevant features for the query concepts. However, the other feature selection algorithms are also welcomed and could be easily employed in the framework to replace the chi-square feature selection method.

The input of the semantic concept detection is a binary-class data set with positive class instances containing the target concept and negative class instances without the target concept. The semantic detection and retrieval component makes use of MSM, BSM, or SMGL to build subspace models to detect the query (target) concept issued by a user. The MSM, BSM, or SMGL constructs a principal component subspace for the positive class and the negative class separately. Then, classification rules are generated based on the data distributions of positive instances and negative instances on their principal component subspaces. These classification rules are used later to predict whether or not a testing instance (representing a testing image or video shot) has the target concept. See Chapter 3.2. for the details about MSM, BSM, and SMGL.

For some concepts, the positive class and negative class have extremely imbalanced data distribution which compromises the subspace models. Clustering-based subspace modeling will be utilized to handle the data imbalance issue. One clustering-based subspace modeling method called CLU-SUMO addresses the data imbalance problem by clustering the negative class into several subsets, each of which merges with the positive class to form a much more balanced subset of the original data set. The idea of applying the clustering algorithm is to generate a number of groups where the data instances within the same group usually have a higher similarity, while the differences among the data instances between different groups should be larger. Since the clustering is quite time-consuming, when there are some peer semantic concepts existing with the target concept, another clustering-based subspace method called CSC-SUMO can be utilized to generate some subsets from original data set based on the class labels. In this way, the number of data instances participating in the clustering part is reduced and the efficiency of training clustering-based subspace modeling methods is improved. Details about CLU-SUMO and CSC-SUMO can be found in Chapter 3.3.

Subspace modeling does not consider inter-concept relationships. Therefore, the ranking subcomponent takes consideration of the inter-concept co-occurrence to rank the retrieved instances that are predicted to be relevant to the query concept. Within the ranking subcomponent, the co-occurrence classes are formed and used to refine the retrieval results from the classification models. The ranking strategies used in the ranking subcomponent are elaborated in Chapter 3.4.

The flowchart of the proposed framework is briefly described as follows. In training phase (shown by the dash arrow), training features are extracted from the training videos or images with labels. These training features are also normalized and appropriately selected so they are useful to detect the query (target) concept. Then, subspace modeling is used to build models to map the target concept to these training features. Furthermore, the inter-concept co-occurrence relationship with regard to each attribute of the training features is captured in the ranking step.

The testing phase (shown in the solid arrow) starts after the training phase is finished. The preprocessing component extracts and selects the same features (the name and the number of the features are the same) as those in the training phase. Also, the testing phase applies z-score normalization using the normalization parameters from the training data. In this way, testing features are generated and further input to the subspace models, where the semantic concept detection is performed. The subspace models provide preliminary ranking for the testing videos and images. Later, the inter-concept co-occurrence relationship based on each attribute renders the final ranking position of these testing videos or images in a list that will be returned to the end user who has issued the query concept.

3.2 Proposed Subspace Modeling Algorithms

Principal component subspace is able to depict the overall data structure from the perspective of variance. In addition, it is easy to achieve dimension reduction [98] by discarding trivial or minority PCs (those PCs whose corresponding eigenvalues are zero or very small). It is possible to use a particular PC subspace to characterize each class and generate a set of rules to achieve multi-class classification. Therefore, it promotes the idea to propose subspace modeling methods in which multi-class subspace modeling is capable of handling multi-class classification problem while binary-class subspace modeling is a special case of multi-class subspace modeling that is designed specifically to cope with binary-class data sets, such as the multimedia data sets in the representation of "one-against-all". In "one-against-all" representation, the instances containing the target concept are called positive instances and the rest of the instances are regarded as negative instances. This is a very commonly seen data preprocessing strategy when encountering a multimedia data set in which one instance may have multiple labels. Under such circumstances, binary-class subspace modeling can be applied to bridge the semantic gap and detect the semantic concepts within the data set.

3.2.1 Multi-class Subspace Modeling

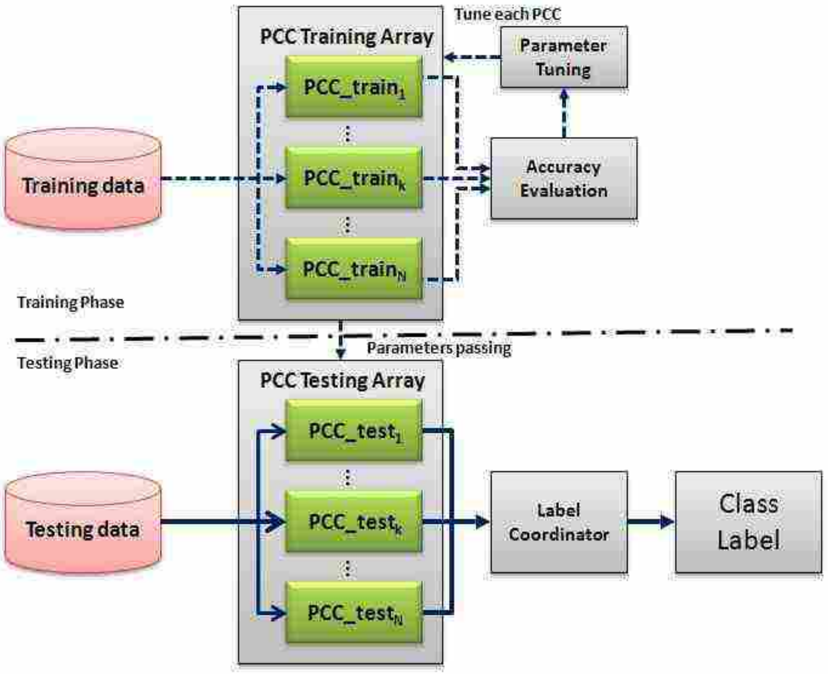


Figure 3.2: The proposed multi-class subspace modeling (MSM) framework

Figure 3.2 depicts the proposed multi-class subspace modeling (MSM) framework. It mainly consists of three parts, namely *principal component classifiers training array*, *principal component classifiers testing array*, and *label coordinator*. *Principal component classifiers training array* is an array of training principle component classifiers (PCC), each of which is called PCC_train; whereas *principal component classifiers testing array* includes the same number of testing PCCs, which are called PCC_test. Please note that each class i has the corresponding PCC_train_i and PCC_test_i pair that are used to recognize its data instances. Figure 3.3 and Figure 3.4 show the architectures of PCC_train_i and PCC_test_i, respectively.

As shown in Figure 3.2, MSM consists of training and testing phases. In the training phase, the training data go through the set of PCC_train_k, k=1 to N, where each PCC_train_k is in charge of recognizing the instances belonging to class k (as shown in the dashed lines). The training data are divided into normal and abnormal parts in each PCC_train_k; the data instances belonging to class k are considered as normal, while the data instances that do not belong to class k are said to be abnormal. The details are presented in the Chapter **Principal Component Classifiers**. Then, by evaluating and tuning the parameters involved in PCC_train_k, each PCC_train_k should be well-tuned for the training model of the class k. Here, an iterative process is adopted to tune the parameters since it is hard to find close-form solutions for these parameters.

During the testing phase, the parameters generated from PCC_train_k will be passed to the corresponding PCC_test_k for testing (e.g., the dissimilarity threshold for PCC_train_k is used in PCC_test_k). The testing data are input to all PCC_test_k, where k=1 to N, as indicated by the solid lines in Figure 3.2. Finally, it is not surprising to find that a testing data instance is not recognized by any PCC_test_k (called “unknown”), or is recognized by more than one PCC_test_k (called “ambiguous”) after collecting the resulting labels

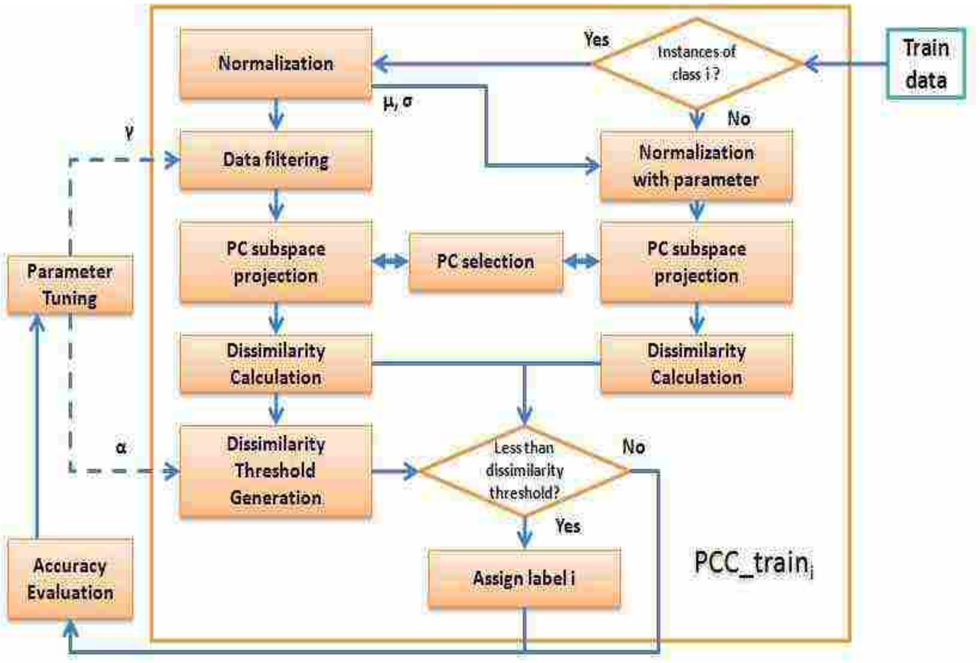


Figure 3.3: Training principal component classifier for class i (PCC_train_i)

from all PCC_test_k (k=1 to N). This problem is addressed by a module called *labor coordinator*, as shown in Figure 3.5, where all testing instances will be assigned one and only one label.

Principle Component Classifiers

There are two types of principle component classifiers, namely PCC_train and PCC_test, which are the basic units of the PCC training array and PCC testing array. In MSM, a pair of PCC_train_k and PCC_test_k is built for class k. A PCC captures the main characteristics of a class in the PC subspace and later utilizes certain rules to classify the data instances in this PC subspace. Unlike typical cases in which the PC subspace is commonly seen in unsupervised learning with the objective to achieve dimension reduction, the PC subspace utilized in MSM is designed for only one class for the purpose of outlining and summarizing the characteristics of that class. That is, one PC subspace is constructed for one class in MSM. The benefit is at least two-fold. On one hand, each

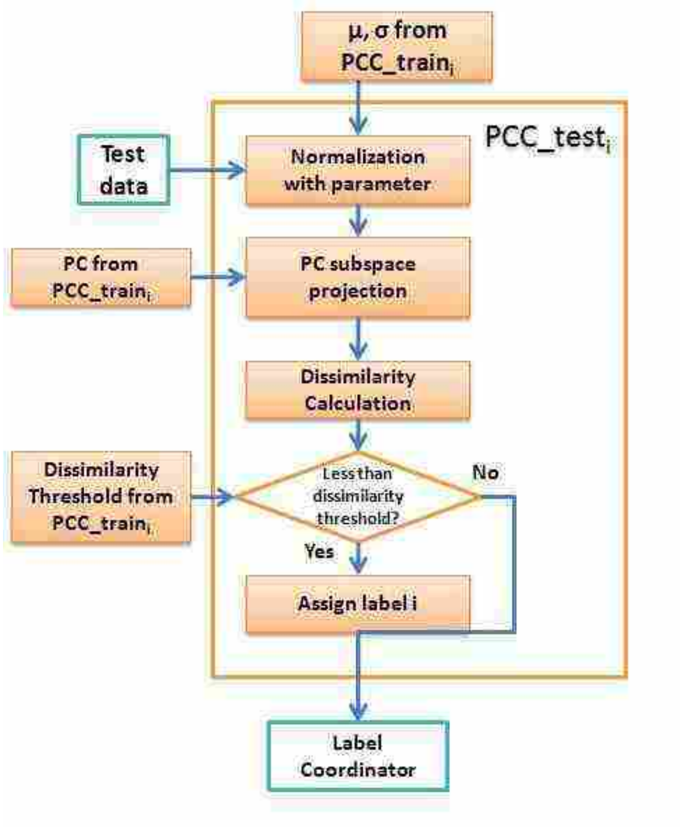

μ, σ from PCC_train$_i$
PCC_test$_i$
Test data
Normalization with parameter
PC from PCC_train$_i$
PC subspace projection
Dissimilarity Calculation
Dissimilarity Threshold from PCC_train$_i$
Less than dissimilarity threshold?
No
Yes
Assign label i
Label Coordinator

Figure 3.4: Testing principal component classifier for class *i* (*PCC_test$_i$*)

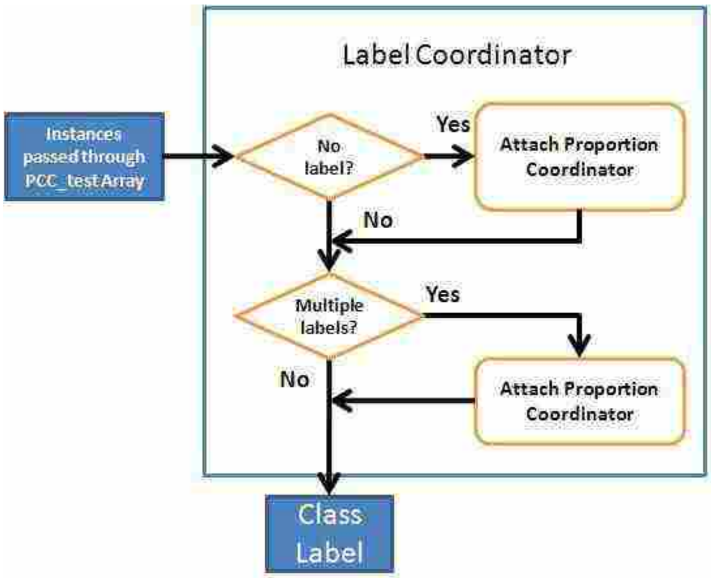

Label Coordinator
Instances passed through PCC_test Array
No label?
Yes
Attach Proportion Coordinator
No
Multiple labels?
Yes
No
Attach Proportion Coordinator
Class Label

Figure 3.5: Label coordinator

PC subspace captures the characteristics of one class more easily than those of multiple classes. On the other hand, it enables more flexibility and accuracy for building the classification model since different classes may be characterized by different PC subspaces.

Figure 3.3 shows the architecture of the training principle component classifier for class i (PCC_train_i). In PCC_train_i, the train data are first divided into two parts according to their labels. The data instances of class i (called normal data instances) go through normalization and data filtering processes. The data instances of the other classes (called abnormal data instances) are normalized using the parameters determined by the normalization process for the normal data instances. Both normal and abnormal data instances select the PCs to construct the PC subspace. A PC subspace projection step here will transfer the normal and abnormal data instances from the original space to the PC subspace, where dissimilarity to the center of class i will be calculated. The dissimilarity value of the normal data instances will later be used to generate the dissimilarity threshold for class i. Finally, all training data instances compare their dissimilarity values with the dissimilarity threshold. If the dissimilarity value of a data instance is less than the dissimilarity threshold, then it is assigned the label of class i. Then, an accuracy evaluation step is taken to carry out the iterations of parameter-tuning.

Figure 3.4 shows the architecture of testing principle component classifier for Class i (PCC_test_i). In fact, the steps are very similar to those of the abnormal data instances in PCC_train_i as shown in Figure 3.3, and so are the descriptions of those steps. As can be easily seen, when the dissimilarity value of a testing data instance is less than the dissimilarity threshold, label i will be assigned to it. No matter whether a label is assigned, the testing data instance goes to “label coordinator”.

The following two key issues arise when MSM is to be built: 1) How to construct a PC subspace where normal data instances can be separated from abnormal data instances as many as possible? 2) What criterion (or what rule) can be used to separate normal and abnormal data instances?

Principle Component Subspace Construction

Constructing a PC subspace is the first key issue to be addressed. In MSM, normalization, filtering, and PC selection play important roles in building this PC subspace. Normalization helps to prevent normal data instances from being dominated by a few large-scale features. Currently, there are a few types of normalization, as shown in Equation (3.1), Equation (3.2), and Equation (3.3). Equation (3.2) and Equation (3.3) ensure the data will be within a certain range after normalization. For example, after applying Equation (3.2), normalized data lie in the range of [-1,1]. Likewise, Equation (3.3) will ensure the normalized data lie at [0,1].

$$\mathbf{X} = \frac{\mathbf{T} - \mu}{\sigma}; \tag{3.1}$$

$$\mathbf{X_m} = \frac{\mathbf{T}}{\mathbf{max}|\mathbf{T}|}; \tag{3.2}$$

$$\mathbf{X_{mm}} = \frac{\mathbf{T} - \mathbf{min}(\mathbf{T})}{\mathbf{max}(\mathbf{T}) - \mathbf{min}(\mathbf{T})}, \tag{3.3}$$

where μ and σ stand for the mean and standard deviation of T, and $min(T)$ and $max(T)$ are the minimum and maximum value of T.

In MSM, Equation (3.1) is adopted to normalize the data since it has the effect of centralizing the normalized data. According to the central limit theorem, X approximates to the Gaussian distribution as long as the number of instances is large enough. Meanwhile, compared with the data after two other types of normalization, the centralized data here are easier for further analysis in a PC subspace. Data filtering may also contribute to construct a good PC subspace. The main concern of employing data filtering here lies in two aspects: 1) the real world data may not be “clean” and PC will be influenced significantly by those "noisy" data; and 2) part of the "clean" data may construct a better model than all of the "clean" data. For these concerns, a factor γ is then introduced in the data filtering step to determine the confidence interval of the retained "clean" data. This is done by first calculating each data instance's Mahalanobis distance $\mathbf{M}_i$, which is defined as follows.

$$\mathbf{M}_i = \sqrt{(\boldsymbol{X}_i - \bar{\boldsymbol{X}})\mathbf{S}^{-1}(\boldsymbol{X}_i - \bar{\boldsymbol{X}})'},$$

where

- $\{\boldsymbol{X}_i, i{=}1,2,\ldots,N\}$ is the instance after normalization;
- $\bar{\boldsymbol{X}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{X}_i$;
- $\mathbf{S} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{X}_i - \bar{\boldsymbol{X}})'(\boldsymbol{X}_i - \bar{\boldsymbol{X}})$.

Here, the Parzen window [99] is adopted to estimate the distribution of the Mahalanobis distance. The Parzen window is utilized extensively in diverse areas, such as classification [100], clustering [101], image segmentation [102], and etc. The Parzen window estimates the probability density function from the given samples by means of data interpolation. The Parzen window estimate is defined as follows:

$$\boldsymbol{P(M)} = \frac{1}{n}\sum_{i=1}^{N}\frac{1}{h_n^d}K\left(\frac{\boldsymbol{M,M}_i}{n}\right),$$

where K(•) is the window function or kernel in a d-dimensional space, $h_n > 0$ is the window width related to the kernel, and n is the number of samples.

Generally speaking, a large $\mathbf{M}_i$ indicates that data instance i deviates from the majority of the normal data instances. Therefore, applying a confidence interval γ on the distribution of Mahalanobis distance may help to retain better training data instances for building the model while wiping out some "noisy" or "trivial" data instances that may compromise the training model. Hence, according to the γ value, the threshold of the Mahalanobis distance $\mathbf{M}_{th}$ corresponding to the confidence interval of $(1-\gamma)\times 100\%$ will be computed. It is easy to see that γ should be between 0 and 1. The retained data instance k will satisfy the condition that $\mathbf{M}_k < \mathbf{M}_{th}$. It should be pointed out that the filtering step utilized here focuses on those normal data instances for each PCC. Some filtered normal data instances in one PCC_train will definitely show up in some other PCC_train, serving as abnormal data. Thus, the filtering step will not reduce the number of training data instances participating in training the model; instead all training data instances will be utilized in the PCC training array.

The aforementioned steps are used for the next important step: *PC selection*. The PCs selected are used for constructing a PC subspace where the original data will be projected for further analysis. A good PC subspace will help to separate normal data instance from abnormal ones. Therefore, the selection method focuses on the separation ability of each PC, where the Fisher criterion is utilized to select those “good” PCs.

Suppose that the original data **X** is projected to **Y** on the PC subspace using Equation (3.4), where each PC is derived from singular value decomposition (SVD). There is an important property [103] for **Y** that the variance of each column of **Y** is the same as the eigenvalue corresponding to that column, which means $var[Y_j] = \lambda_j \ , j = 1,2,\ldots$. Here, the PCs with zero eigenvalue are considered trivial ones that do not have any separation ability and thus they will simply be discarded. Therefore, $var[Y_j] > 0$ for all PCs. Due to the centralization effect of normalization, now each column of normal data instances holds the mean value of zero. To simplify the analysis further, each column of the projected data will be divided by the square root of λ_j so that the normal data instances will have a standard deviation value equal to **1**.

$$\mathtt{Y} = \boldsymbol{X} \cdot \mathbf{PC}, \tag{3.4}$$

where $\boldsymbol{X}$ is the training data after the normalization and data filtering steps. and $\mathbf{PC} = (PC_1, PC_2, \ldots)$ is a group of PCs.

The Fisher criterion defines the separation ability as the ratio of the variance between the classes to the variance within the classes, as shown in Equation (3.5).

$$\mathtt{S} = \frac{\sigma_b}{\sigma_w}, \tag{3.5}$$

where σ_b stands for the variance between the classes and σ_w stands for the variance within the class.

So, it is easy to see that σ_w =1 for each PC. To measure the variance between classes, Hellinger distance [104] is introduced here to measure the distance between the normal and the abnormal instances. For the normal instances projected on PC_k, they satisfy normal distribution with a zero mean and one variance. For the abnormal data, it is assumed that their projection on each PC approximates Gaussian distribution for simplicity. This assumption usually holds as long as the number of abnormal instances is large enough. Thus, σ_b equals to the square of Hellinger distance, which is displayed in the form of measuring two normal distributions in Equation (3.6).

$$\mathtt{H(N,A)} = (1 - \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2+\sigma_2^2}} e^{-\frac{1}{4}\frac{(\mu_1-\mu_2)^2}{\sigma_1^2\sigma_2^2}})^{1/2}, \tag{3.6}$$

where

- μ_1 and σ_1 are the mean and standard deviation values of $\mathtt{N}$, which stands for the projected normal data on one PC.

- μ_2 and σ_2 are the mean and standard deviation values of $\mathtt{A}$, which stands for the projected abnormal data on the same PC.

Based on the Fisher criterion, the separation ability $\mathtt{S}_j$ of the j^{th} PC can be written as Equation (3.7).

$$\mathtt{S}_j = H_j^2, \tag{3.7}$$

where H_j stands for Hellinger distance calculated from the j^{th} PC.

According to Equation (3.7), all PCs can be ranked according to their separation ability. The larger S that a PC holds, the easier it is to separate normal data instances from abnormal instances on that PC. The value of $\mathtt{S}$ is located within 0 to 1, since $H \in [0,1]$. It is then possible to rank the PCs by their separation ability and to search the best PC combination in a linear way or just simply set a threshold for $\mathtt{S}$. In practice, a threshold PC_{th} is often required to remove those PCs that are of little help to separate normal data instances. Without this ranking to facilitate searching for the best combination of PCs, it may require an exhaustive search. Even a greedy forward or backward searching algorithm will take the time complexity of $O(N^2)$ to finish the search task, while the time complexity of the search in MSM is $O(N)$. Finally, the best combination of PCs will span the PC subspace for generating the classification rule.

Classification Rule Generation

As pointed out before, each column of normal data instances $\mathtt{Y}$ satisfies the following two conditions shown in Equation (3.8) and Equation (3.9).

$$mean(\mathtt{Y}_j) = 0\ , j = 1,2... \tag{3.8}$$

$$var(\mathtt{Y}_j) = \lambda_j\ , j = 1,2... \tag{3.9}$$

Because of these two useful conditions, a dissimilarity measure called *DSM* is proposed to calculate the dissimilarity of each instance towards a specified class. For data instance *i*, its *DSM* is defined as follows.

$$\texttt{DSM}_i = \sum_j \frac{Y_{ij}^2}{\lambda_j}, j = 1, 2... \tag{3.10}$$

where Y_{ij} is data instances i's projection on PC_j.

For normal data instances, their *DSM* dissimilarity values are small since Y_{ij} is distributed around 0. However, for the abnormal data instances, their *DSM* dissimilarity values may be very large as long as there is one PC_k that holds a large $\frac{Y_{ik}^2}{\lambda_k}$ value. Therefore, a classification rule related to this dissimilarity value is defined as follows.

CLASSIFICATION RULE

```
1   With regard to data instance $i$,
2   **if** $DSM_i \leq DSM_{thresh}$ **then**
3     assign the label of the current class to data instance $i$.
```

To get this DSM_{thresh}, a factor called α is utilized as the confidence level. Similar to the data filtering step, a Parzen window is applied to the DSM values of all normal data instances. DSM_{thresh} is the value corresponding to the confidence interval of $(1-\alpha) \times 100\%$ of the upper tail of *DSM* distribution. Here, the upper tail means the distribution tail that expands to positive infinity.

To optimize each PCC_train, two loops will be applied to search for the optimal parameters, one for γ and the other for α. Both γ and α values are within [0,1], and domain knowledge may further narrow this iteration range. The time complexity for such optimization is at most $O(N^3)$, if one loop is required for searching the PCs.

Label Coordinator

There are two possible issues after assigning labels to a data instance in the PCC testing array. This is when the label coordinator is applied to resolve such issues.

1. A testing data instance may not be recognized by any PCC, which is called "unknown";

2. A testing data instance may be recognized as normal by more than one PCC and is assigned multiple labels, which is called “ambiguous”.

Finding the criteria indicating how likely one data instance is towards different classes is not easy since different classes may have diverse PC subspaces and DSM_{thresh} values. For this purpose, the *Attach Proportion* measure in [105] is adopted. The *Attaching Proportion* $ATP_i^{(k)}$ of data instance i towards class label k is defined by Equation (3.11).

$$ATP_i^{(k)} = \frac{\mathtt{DSM}_i{}^{(k)}}{\mathrm{DSM}_{\mathrm{thresh}}{}^{(k)}}. \tag{3.11}$$

The architecture of the label coordinator is shown in Figure 3.5. When a testing data instance arrives at the label coordinator, the label coordinator will first check if it is an “unknown” data instance (i.e., no class label). Although none of the classes claims to recognize the “unknown” data instance as its member, each of them does judge the dissimilarity of the “unknown” data instance towards itself. This dissimilarity information can be taken into consideration to solve this “unknown” issue. The solution is to assign each “unknown” data instance with the label of the class with the lowest *Attaching Proportion* value, meaning that a smaller *Attaching Proportion* value implies a closer relationship between the instance and that class. On the other hand, a testing data instance may be “ambiguous” and thus the label coordinator is applied to determine only the one class label that has the lowest *Attaching Proportion* value to it. In case there is a tie in the smallest *Attaching Proportion* values, the testing data instance will be classified to the class possessing the lowest $\mathrm{DSM}_{\mathrm{thresh}}$ value. If there remains a tie in the $\mathrm{DSM}_{\mathrm{thresh}}$ values, then the class label of the data instance can be randomly selected. In such a manner, a testing instance is ensured to have exactly one class label.

Experiment Design

The data sets used for evaluating the performance are obtained from a few sources: UCI Machine Learning Repository [106], KDD CUP 1999 [107], the Local Area Network (LAN) Testbed [108], and TRECVID [42].

Table 3.1: Information about all data sets

date set ID	data set name	source	# of classes	# of attributes
Group 1	Iris	UCI	3	4
Group 2	Wine	UCI	3	13
Group 3	Statlog	UCI	4	18
Group 4	KDD	KDD	4	44
Group 5	SPECTF Heart	UCI	2	22
Group 6	Multiple Features	UCI	9	76
Group 7	Haberman's Survival	UCI	2	3
Group 8	Blood Transfusion	UCI	2	4
Group 9	Testbed	LAN	9	43
Group 10	Emergency vehicle	TRECVID	2	48
Group 11	Dog	TRECVID	2	48
Group 12	Mountain	TRECVID	2	48

Experiment Result

Table 3.2, Table 3.3, and Table 3.4 display the classification accuracy of MSM, C4.5 decision tree (C4.5), logistic regression (Logistic), support vector machines (SVM), AdaBoost-SVM, AdaBoost-C4.5, nearest neighbor (NN), K-nearest neighbor (KNN), Random Forest, decision table, one rule, naive Bayes, multilayer perceptron, repeated incremental pruning to produce error reduction (RIPPER), and a partial decision tree-based classification algorithm called PART for each group of data sets.

From Group 5 and Group 8, it can be clearly seen that MSM outperforms the other classification approaches even if the classes are imbalanced. For Group 5 and Group 8, their negative to positive ratios are 267:55 and 570:178, respectively. The superiority of MSM over the other classification methods is presented distinctly in Group 5. The

Table 3.2: Average multi-class supervised classification accuracy (with standard deviation) among MSM, SVM, Logistic, Naive Bayes, NN, KNN, AdaBoost-SVM, AdaBoost-C4.5, Decision Table, RIPPER, One Rule, PART, C4.5, Random Forest, and Multilayer Perceptron. Classification accuracy is shown below. (Groups 1-3)

Mean_Accuracy %	**Group 1**	**Group 2**	**Group 3**
MSM	97.32%(±0.56)	98.59%(±0.84)	82.73%(±0.68)
SVM	95.94%(±0.49)	97.14%(±0.77)	72.80%(±0.68)
Logistic	95.06%(±1.53)	96.63%(±1.09)	78.61%(±0.87)
Naive Bayes	95.07%(±1.29)	96.91%(±0.47)	44.92%(±1.20)
NN	95.34%(±0.84)	95.28%(±0.80)	68.54%(±0.91)
KNN	95.94%(±0.67)	96.01%(±0.62)	68.97%(±0.87)
AdaBoost-SVM	95.67%(±0.72)	96.80%(±1.39)	72.85%(±0.71)
AdaBoost-C4.5	93.86%(±1.50)	95.39%(±1.70)	74.79%(±1.21)
Decision Table	93.93%(±1.32)	86.94%(±3.86)	63.31%(±1.41)
RIPPER	92.59%(±1.62)	89.62%(±1.61)	69.79%(±1.69)
One Rule	94.11%(±0.98)	77.81%(±1.98)	51.54%(±0.91)
PART	93.93%(±1.51)	92.03%(±1.71)	71.14%(±0.78)
C4.5	94.00%(±1.45)	92.48%(±1.82)	70.98%(±0.80)
Random Forest	94.39%(±1.34)	95.95%(±1.17)	73.97%(±1.12)
Multilayer Perceptron	95.66%(±1.13)	96.79%(±0.61)	80.08%(±1.33)

Table 3.3: Average multi-class supervised classification accuracy (with standard deviation) among MSM, SVM, Logistic, Naive Bayes, NN, KNN, AdaBoost-SVM, AdaBoost-C4.5, Decision Table, RIPPER, One Rule, PART, C4.5, Random Forest, and Multilayer Perceptron. Classification accuracy is shown below. (Groups 4-6)

Mean_Accuracy %	**Group 4**	**Group 5**	**Group 6**
MSM	94.40%(±0.11)	84.04%(±0.32)	82.22%(±0.22)
SVM	94.39%(±0.14)	80.79%(±1.21)	81.86%(±0.42)
Logistic	93.31%(±0.24)	80.52%(±1.04)	78.42%(±0.38)
Naive Bayes	94.17%(±0.15)	76.48%(±1.23)	73.08%(±0.33)
NN	90.13%(±0.35)	76.85%(±1.40)	79.71%(±0.51)
KNN	94.00%(±0.17)	81.69%(±1.01)	81.82%(±0.29)
AdaBoost-SVM	94.03%(±0.29)	80.67%(±0.99)	81.49%(±0.44)
AdaBoost-C4.5	91.11%(±0.37)	80.64%(±1.54)	76.35%(±0.48)
Decision Table	94.36%(±0.30)	79.14%(±0.50)	52.51%(±2.31)
RIPPER	94.29%(±0.36)	80.19%(±2.34)	67.68%(±1.18)
One Rule	93.58%(±0.27)	79.40%(±0.00)	26.76%(±0.87)
PART	93.23%(±0.39)	79.89%(±1.31)	71.00%(±1.18)
C4.5	93.58%(±0.33)	78.65%(±1.61)	69.45%(±0.88)
Random Forest	91.13%(±0.26)	80.97%(±1.04)	76.82%(±0.37)
Multilayer Perceptron	93.76%(±0.31)	78.76%(±0.95)	80.82%(±0.47)

Table 3.4: Average multi-class supervised classification accuracy (with standard deviation) among MSM, SVM, Logistic, Naive Bayes, NN, KNN, AdaBoost-SVM, AdaBoost-C4.5, Decision Table, RIPPER, One Rule, PART, C4.5, Random Forest, and Multilayer Perceptron. Classification accuracy is shown below. (Groups 7-9)

Mean_Accuracy %	**Group 7**	**Group 8**	**Group 9**
MSM	75.13%(±0.76)	78.61%(±0.47)	98.77%(±0.10)
SVM	73.14%(±0.40)	76.16%(±0.09)	91.31%(±0.76)
Logistic	74.08%(±0.41)	77.29%(±0.40)	98.53%(±0.24)
Naive Bayes	74.67%(±0.58)	75.07%(±0.27)	98.58%(±0.09)
NN	66.99%(±2.17)	60.19%(±0.95)	98.72%(±0.17)
KNN	71.86%(±1.16)	76.58%(±0.86)	98.44%(±0.25)
AdaBoost-SVM	74.48%(±1.21)	76.79%(±0.57)	98.66%(±0.13)
AdaBoost-C4.5	72.16%(±1.30)	77.07%(±1.18)	98.42%(±0.20)
Decision Table	72.48%(±1.20)	75.53%(±0.97)	98.22%(±0.35)
RIPPER	71.80%(±1.03)	77.66%(±1.49)	97.42%(±0.52)
One Rule	71.93%(±1.87)	76.02%(±0.45)	56.51%(±1.05)
PART	71.93%(±1.22)	77.00%(±1.28)	98.64%(±0.17)
C4.5	71.80%(±0.95)	77.00%(±1.14)	98.62%(±0.17)
Random Forest	67.65%(±2.27)	73.09%(±0.90)	98.49%(±0.11)
Multilayer Perceptron	73.14%(±1.52)	77.61%(±0.75)	98.72%(±0.12)

Table 3.5: Average multi-class supervised classification accuracy (with standard deviation) among MSM, SVM, Logistic, Naive Bayes, NN, KNN, AdaBoost-SVM, AdaBoost-C4.5, Decision Table, RIPPER, One Rule, PART, C4.5, Random Forest, and Multilayer Perceptron. Classification accuracy is shown below. (Groups 10-12)

Mean_Accuracy %	**Group 10**	**Group 11**	**Group 12**
MSM	75.23%(±0.11)	77.02%(±0.15)	75.14%(±0.02)
SVM	75.00%(±0.00)	76.61%(±0.12)	75.00%(±0.00)
Logistic	72.98%(±0.49)	75.99%(±0.26)	74.31%(±0.26)
Naive Bayes	68.24%(±0.24)	68.25%(±0.44)	61.79%(±0.34)
NN	67.41%(±1.48)	70.84%(±0.56)	66.82%(±0.29)
KNN	73.64%(±0.87)	76.22%(±0.55)	73.21%(±0.29)
AdaBoost-SVM	71.05%(±1.21)	76.34%(±0.42)	75.00%(±0.01)
AdaBoost-C4.5	72.34%(±0.79)	73.27%(±0.67)	71.62%(±0.32)
Decision Table	74.50%(±0.32)	75.94%(±0.46)	74.67%(±0.16)
RIPPER	72.26%(±1.02)	75.28%(±0.56)	74.38%(±0.05)
One Rule	70.12%(±0.90)	72.31%(±0.75)	74.32%(±0.36)
PART	72.55%(±1.56)	75.14%(±0.89)	73.77%(±0.66)
C4.5	70.93%(±1.85)	72.89%(±0.75)	71.81%(±0.74)
Random Forest	74.70%(±0.21)	76.63%(±0.58)	74.93%(±0.10)
Multilayer Perceptron	67.70%(±0.59)	69.94%(±1.25)	66.52%(±1.27)

Table 3.6: Significance test for Group 1 to Group 12

Date set	**one-tail p-value**
Group 1	0.000
Group 2	0.001
Group 3	0.000
Group 4	0.406
Group 5	0.000
Group 6	0.019
Group 7	0.087
Group 8	0.000
Group 9	0.193
Group 10	0.004
Group 11	0.000
Group 12	0.019

accuracy of MSM is 84.04%, while the accuracy values of the compared methods are around 80%. For example, the accuracy values of SVM and Random Forest are almost 3% worse than that of MSM.

One thing worth mentioning is that the promising performance of MSM is not confined to only small data sets such as Group 1 and Group 3. Experimental results show that it also works well in medium-sized data sets such as Group 6, which is composed of 1800 data instances and 76 attributes.

The result of the significance test is shown in Table 3.6. As is shown in this table, MSM can outperform all the other classification algorithms significantly for six groups with p-value less than or equal to 0.1% based on a one-tail student's t-test.

3.2.2 Binary-class Subspace Modeling

In the semantic concept detection task, a common strategy is to build an individual model for each semantic concept. In this strategy, all training instances related to the target concept are regarded as positive class instances and the remaining instances are considered as negative class instances. In a multimedia data set, each instance is often attached with multiple labels. It is often difficult and requires complicated classification models to address such a multi-label problem. However, the "positive-negative" presentation of semantic concepts within a multimedia data set can simplify the multi-label problem into an easy-to-solve single-label classification problem since many classification algorithms [109, 110] assumed an instance only has one label when they were proposed.

The aforementioned MSM has a major limitation in that it requires a long training time since there are too many parameters requiring iterative steps to decide. Particularly, the iterative steps related to the selection of parameter γ in data filtering lead to a number of singular value decomposition operations which are very computationally expensive, especially for a large data set. The proposed binary-class subspace modeling (BSM) can be regarded as the special case of MSM that deals with a binary class data set. Compared with MSM, BSM has the following contributions:

- *The number of involved parameters is reduced.* Compared with MSM, the assignment of a final label to an instance (by label coordinator) can be simplified. For example, Equation (3.12) shows the MSM's rule of assigning a positive class label (P) instead of a negative class label (N) to an instance i.

$$\frac{\mathtt{DSM}_i{}^{(P)}}{\mathrm{DSM}_{\mathrm{thresh}}{}^{(P)}} < \frac{\mathtt{DSM}_i{}^{(N)}}{\mathrm{DSM}_{\mathrm{thresh}}{}^{(N)}}. \tag{3.12}$$

  This equation can be simplified as:

$$\mathtt{DSM}_i^{(N)} \cdot w - \mathtt{DSM}_i^{(P)} > 0 \tag{3.13}$$

  where $w = \frac{DSM_{thresh}{}^{(P)}}{DSM_{thresh}{}^{(N)}}$. In this way, only one variable w is required instead of $DSM_{thresh}{}^{(P)}$ and $DSM_{thresh}{}^{(N)}$.

- *PC selection is integrated with data filtering.* In MSM, the selection of data filtering parameter γ is accompanied with a number of singular value decomposition. In the proposed BSM framework, the singular value decomposition will only be applied twice, one for the positive class and the other for the negative class. The removal of part of the normal training instances in the PC selection step of BSM achieves the same effect as data filtering in MSM. Thus, such an integration reduces the computational cost and saves the time for training BSM models.

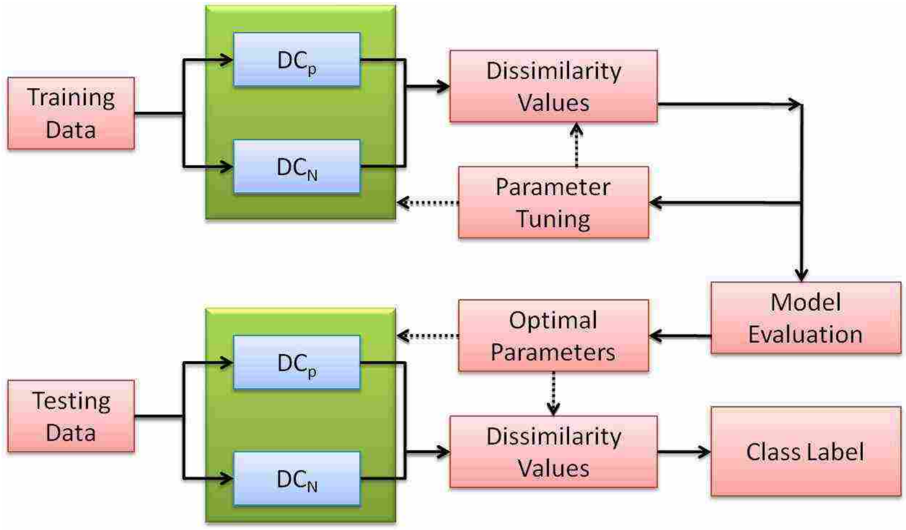


Figure 3.6: Binary-class subspace modeling

Fig. 3.6 presents the framework of the proposed BSM. There are two dissimilarity calculators: DC_P for training a positive one-class model of the target concept and DC_N for training a negative one-class model for non-target concepts. Each training data instance will get a pair of dissimilarity values, namely $DisTrain_P$ and $DisTrain_N$, from DC_P and DC_N, respectively. Later, a weighted parameter β is calculated and the label of each data instance is decided by comparing $DisTrain_P$ and $DisTrain_N^{'}$ (which is $DisTrain_N * \beta$). A model evaluation module then evaluates the performance of the algorithm and learns the optimal parameters automatically. Later, the optimal parameters are passed into two dissimilarity calculators (DC_P and DC_N) for testing data instances to calculate the pair of dissimilarity values ($DisTest_P$ and $DisTest_N$). The ranking scores and class labels for testing data instances are predicted by $DisTest_P$ and $DisTest_N^{'}$.

Dissimilarity Calculators

The dissimilarity calculators (DC_P and DC_N) consist of three steps: normalization, subspace projection, and dissimilarity calculation.

In the proposed approach, the z-score normalization process is applied separately to positive and negative data instances. Taking DC_P for example, the z-score normalization scales each feature of positive training data instances to be a zero mean and unit standard deviation. By means of normalization, (i) the characteristics of positive data instances can be prevented from being dominated by a few large-scale features, and (ii) the mean and standard deviation can roughly describe the statistical information about the positive data instances. Equation (3.14) shows the utilized z-score normalization, where $X^{'}$ stands for the data instances to be normalized, and $mean(X^{'})$ and $std(X^{'})$ are the mean and standard deviation of $X^{'}$.

$$X = \frac{X^{'} - mean(X^{'})}{std(X')}. \tag{3.14}$$

The next step is subspace projection. Its main goal is to de-correlate the features. The singular value decomposition (SVD) is adopted to de-correlate features of X in the original space. Through this decomposition, a list of eigenvalue-eigenvector pairs are derived, denoted as (λ_1, PC_1), (λ_2, PC_2), $\ldots$, (λ_l, PC_l). Each eigenvector here is also called a PC (principal component). In addition, an analysis on eigenvalues can help select representative eigenvectors to achieve dimension reduction, which simplifies further analyses. The process to search those representative PCs (R_PC_1, R_PC_2, $\ldots$,

R_PC_g) is important, since a suitable set of representative PCs will not only reduce the dimension of the data ($g < l$) but also remove the dimensions that are linearly dependent on the others (i.e., dimensions f whose eigenvalues $\lambda_f = 0$).

One of the contributions of BSM compared with MSM lies in its PC selection method. The process to search for those representative PCs consists of the following five steps:

(1) Instances $X = \{x_m\}$ are projected as $Y = \{y_{mr}\}$ onto the PC subspace as Equation (3.15), where the r-th column of Y, Y_r, satisfies the normal distribution where the variance equals λ_r.

$$y_{mr} = X_m \cdot PC_r = x_{m1}y_{1r} + x_{m2}y_{2r} + ... + x_{ml}y_{lr}, \tag{3.15}$$

where

- X_m is the m-th data instances of X;
- PC_r is the r-th PC corresponding to λ_r.

(2) A confidence interval ε is used to calculate the lower and upper bounds of Y_r corresponding to normal data instances to reduce the influence of noisy data.

(3) A variable called ω which denotes the percentage of abnormal data instances that lie outside the boundary calculated on a certain PC.

(4) Rank all PCs based on their corresponding ω values. The PC with the largest ω value ranks first, the second largest follows, and etc.

(5) Search the optimal combination of PCs using the first K ranked PCs. In this way, different PCs can be ranked and combined to reduce the time complexity of searching representative PCs (R_PC).

For further process, the normalized data instances X are projected on the subspace spanned by these representative PCs using SC_h=$X \cdot R\_PC_h$, representing the score of the h-th representative PC. Utilizing the important property, namely mean(SC_h)= 0 and var(SC_h)= λ_h, a dissimilarity distance measure can be defined to measure the closeness of each data instance to different training models.

Since each column is already uncorrelated as a result of subspace projection together with the aforementioned property of the projected data, a dissimilarity measure called *Dis* (shown in Equation (3.16)) can be obtained, where SC_h is calculated from the previous step and λ_h is the h-th eigenvalue from SVD ($\lambda_h > 0$). This measure is the square of the chi-square distance as shown and proved as follows.

$$Dis(SC) = \sum_h \frac{(\mathtt{SC}_h)^2}{\lambda_h}. \tag{3.16}$$

Proof 1 *Suppose there is a function* $ChiS(x) = Dis(x)^{\frac{1}{2}}$

- *Non-negative: it is obvious to see that* $ChiS(x) \geq 0$*;*
- *Symmetric: it is also easy to prove that* $ChiS(x-y) = ChiS(y-x)$*;*
- *Triangle inequality: for n-dimension vectors* $x = [x_1, x_2, ..., x_n]$ *and* $y = [y_1, y_2, ..., y_n]$, *suppose substituting* $(x_h/\lambda_h^{1/2})$ *with* $x_h^{(new)}$. *The proof of the triangle inequality of* $ChiS(x)$ *is the same as that of the triangle inequality of n-Dimensional Euclidean distance.*

Weight Parameter and Model Evaluation

Here is the example of DC_P. The dissimilarity value for positive data instances is small. However, since the negative data instances may be heterogeneous to the positive data instances in their data characteristics, they usually hold large dissimilarity values, indicating their poor fitness to the positive model. The dissimilarity value can therefore be used to distinguish the negative data instances from the positive instances. The same rules hold for positive data instances to the negative model.

Typically, there are too many misclassifications due to the data imbalance issue. By introducing a weight parameter, the algorithm can offer a fair treatment to the positive model, preventing it from being dominated by the negative model. The model evaluation part is quite significant in that it tunes β to achieve the optimal (or near-optimal) classification performance. The best β for the training model is searched via a small-step iteration by evaluating the $F1$-score of the training model.

From the empirical study, $init\_value$=-0.3, end_value=0.3, and the step size s=0.01. The process to select optimal β can be found as follows.

SELECTION OF OPTIMAL β

```
$F1best \leftarrow 0$;
$\beta_{OPT} \leftarrow init\_value$;
for $\beta \leftarrow init\_value$ to $end\_value$ with step $s$
    compute F1 using current $\beta$
    if F1 is higher than F1best then
        $F1best \leftarrow F1$;
        $\beta_{OPT} \leftarrow \beta$;
end
```

Classification Rules

After applying each testing data instance to two dissimilarity calculators (DC_P and DC_N) built by training data instances, the dissimilarity values $DisTest_P$ and $DisTest_N$ can be obtained. By comparing $DisTest_P$ and $DisTest_N^{'}$ (=$DisTest_N \times \beta_{OPT}$), it assigns the data instance with positive or negative label according to its dissimilarity towards positive and negative models. If $DisTest_P = DisTest_N^{'}$, then the classifier assigns the data instance to the positive class considering that the cost of misclassifying a positive data instance is higher than that of misclassifying a negative one. The classification rules are as follows.

- If $DisTest_P \leq DisTest_N^{'}$, assign positive label to the data instance;
- If $DisTest_P > DisTest_N^{'}$, assign negative label to the data instance.

The dissimilarity from BSM can also be used to generate ranking scores to measure the relevance of an instance to the target class (this will be used in later sections). With regard to a typical relevant positive data instance, it should be close to the positive model and far away from the negative model. In other words, $DisTest_N^{'}$ should be large and $DisTest_P$ should be small. Thus, the ranking score RS of a BSM is defined in Equation (3.17).

$$RS = DisTest_N^{'} - DisTest_P \tag{3.17}$$

A larger value of RS indicates a higher probability of the data instance belonging to the positive class. The proposed ranking strategy emphasizes the comparison of the dissimilarity that a data instance holds towards the positive model and negative model. The time complexity of the BSM is $O(N^2)$ at the training phase, which is mainly from the process of SVD, while such time complexity is $O(1)$ at the testing phase.

Experiments and Results

The high-level semantics to be extracted in the experiment are from TRECVID 2008 and 2009. “Bridge”, “emergency vehicle”, “kitchen”, “two people”, “driver”, “street”, “mountain” and “flower” are the concepts in TRECVID 2008 while “classroom”, “doorway”, “airplane-flying”, “bus”, “cityscape”, “demonstration protest”, “hand” and “singing” are from TRECVID 2009. The statistics information of these semantic concepts is shown in Table 3.7. There are various types of videos in TRECVID video collections, such as news magazines, science news, news reports, documentaries, educational pro-

Table 3.7: Information of high-level semantic concepts to be extracted, with name, number of positive instance in the testing set, and the ratio between positive and negative instances.

index	name	no. pos	pos/neg
1	bus	38	0.009
2	emergency vehicle	72	0.018
3	airplane flying	109	0.027
4	bridge	160	0.040
5	kitchen	169	0.042
6	demonstration protest	181	0.045
7	doorway	206	0.052
8	mountain	257	0.065
9	driver	263	0.066
10	flower	283	0.071
11	classroom	291	0.073
12	singing	292	0.074
13	cityscape	442	0.112
14	street	651	0.168
15	hand	928	0.242
16	two people	1534	0.420

gramming, and archival videos, which might be black-and-white or colored, and may or may not have sound. Therefore, experiments on these high-level concepts from TRECIVD videos could show the accuracy, robustness, and adaption of the proposed binary-class subspace modeling framework.

To show the efficiency and effectiveness of the proposed framework, it is compared with several other well-known classifiers, including K-nearest neighbor with K=3 (3NN), multilayer perceptron (MP), support vector machines (SVM), support vector machines with chi-square kernel (Chi), rule based JRip (JR), decision tree (C4.5), and AdaBoost with decision tree kernel (Ada). To get a standard implementation of each classification algorithm, the implementation in Waikato Environment for Knowledge Analysis (Weka) is adopted to perform the comparative experiment. The parameters of different classifiers is also tuned in the hope that the classifier is compared fairly with them. To evaluate the framework, the precision (pre), recall (rec), and F1-score (F1) performance metrics are adopted under the three-fold cross-validation approach.

The performance of BSM compared with seven other classifiers for all sixteen different concepts is shown from Table 3.8 and Table 3.9. From these tables, it can be seen BSM outperforms the other classifiers in both tables. The average performance over all concepts is shown in Table 3.10. It is obvious that BSM's F1-score value is on average at least 7% better than the other classifiers. Unlike Chi (SVM with Chi-square kernel), which has a high recall but an unacceptable low precision; and Ada (AdaBoost with Decision Trees kernel), which has a high precision but unacceptable low recall, BSM makes the best trade-off between precision and recall. Moreover, although the training steps require SVD decomposition, which might be expensive ($O(N^2)$), together with one PC selection step ($O(N)$) and an iteration to get optimal β value ($O(N)$), the total complexity of the training stage is $O(N^2)$ because of that SVD decomposition , PC selection and parameter optimization are not embedded into each other.

One highlight worth pointing out is that for those extremely imbalanced data, BSM can still work (though the performance is considered poor) while the other classifiers almost could not build good models. Take concepts "emergency vehicle" (in Table 3.8) and "bus" (Table 3.9) for example. As shown in Table 3.7, the number of positive instances for these two concepts are less than 100, but BSM reaches the highest F1-score value while the other classifiers even have 0 values for the F1-score.

3.2.3 Integration of Subspace Modeling on Global and Local Structures

Two subspace modeling methods (MSM and BSM) aiming to handle the semantic gap issue in multimedia information retrieval are proposed, which try to capture the overall structure of each class and construct subspaces where the classification rules are generated. The challenge in capturing the overall structure of each class is that the distribution of the data is rather complex and far from the Gaussian distribution in

Table 3.8: Comparative performance for eight concepts

Bridge	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.34	0.44	0.31	0.44	0.21	0.00	0.03	0.19
recall	0.10	0.10	0.10	0.03	0.08	0.00	0.44	0.23
F1	0.15	0.16	0.15	0.05	0.11	0.00	0.06	0.20
E. vehicle	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.17	0.11	0.00	0.00	0.02	0.00	0.02	0.06
recall	0.01	0.00	0.00	0.00	0.00	0.00	0.36	0.03
F1	0.03	0.01	0.00	0.00	0.01	0.00	0.03	0.04
Kitchen	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.32	0.37	0.16	0.12	0.13	0.00	0.04	0.15
recall	0.07	0.06	0.04	0.01	0.04	0.00	0.47	0.16
F1	0.11	0.10	0.06	0.02	0.06	0.00	0.07	0.15
Two people	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.42	0.48	0.49	0.52	0.43	0.00	0.29	0.38
recall	0.33	0.37	0.29	0.23	0.47	0.00	0.54	0.68
F1	0.37	0.42	0.36	0.31	0.43	0.00	0.38	0.48
Driver	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.27	0.37	0.36	0.11	0.21	0.00	0.06	0.19
recall	0.06	0.07	0.06	0.01	0.06	0.00	0.41	0.17
F1	0.10	0.12	0.10	0.01	0.09	0.00	0.10	0.17
Street	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.30	0.39	0.33	0.42	0.34	0.00	0.13	0.25
recall	0.15	0.20	0.15	0.06	0.19	0.00	0.42	0.48
F1	0.20	0.26	0.20	0.11	0.24	0.00	0.20	0.32
Mountain	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.16	0.24	0.18	0.19	0.14	0.00	0.06	0.12
recall	0.04	0.04	0.05	0.01	0.07	0.00	0.45	0.17
F1	0.06	0.06	0.08	0.01	0.09	0.00	0.11	0.14
Flower	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.21	0.14	0.13	0.00	0.17	0.00	0.06	0.13
recall	0.05	0.03	0.02	0.00	0.04	0.00	0.43	0.16
F1	0.08	0.04	0.04	0.00	0.07	0.00	0.11	0.15

Table 3.9: Comparative performance for another eight concepts

Classroom	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.18	0.31	0.19	0.00	0.19	0.00	0.06	0.16
recall	0.05	0.07	0.03	0.00	0.08	0.00	0.42	0.24
F1	0.08	0.11	0.05	0.00	0.11	0.00	0.11	0.18
Doorway	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.26	0.22	0.23	0.14	0.22	0.05	0.05	0.20
recall	0.04	0.05	0.05	0.01	0.11	0.41	0.41	0.22
F1	0.07	0.08	0.09	0.02	0.15	0.08	0.08	0.21
A. flying	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.24	0.53	0.42	0.44	0.24	0.00	0.03	0.16
recall	0.04	0.09	0.08	0.06	0.03	0.00	0.43	0.13
F1	0.07	0.15	0.13	0.11	0.06	0.00	0.05	0.14
Bus	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.22	0.13	0.00	0.00	0.14	0.00	0.01	0.54
recall	0.02	0.02	0.00	0.00	0.05	0.00	0.32	0.04
F1	0.03	0.03	0.00	0.00	0.08	0.00	0.01	0.08
Cityscape	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.25	0.28	0.23	0.83	0.21	0.00	0.04	0.18
recall	0.09	0.10	0.08	0.05	0.10	0.00	0.40	0.25
F1	0.13	0.15	0.12	0.09	0.13	0.00	0.07	0.21
D.Protest	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.32	0.43	0.23	0.83	0.21	0.00	0.04	0.22
recall	0.10	0.06	0.08	0.05	0.10	0.00	0.40	0.16
F1	0.15	0.11	0.12	0.09	0.13	0.00	0.07	0.18
Hand	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.30	0.37	0.36	0.42	0.35	0.00	0.19	0.30
recall	0.16	0.20	0.16	0.08	0.16	0.00	0.51	0.43
F1	0.21	0.26	0.22	0.14	0.21	0.00	0.28	0.34
Singing	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.34	0.47	0.33	0.45	0.30	0.00	0.06	0.20
recall	0.11	0.15	0.18	0.11	0.13	0.00	0.38	0.32
F1	0.16	0.22	0.23	0.18	0.17	0.00	0.11	0.23

Table 3.10: Average comparative performance for all concepts

AVERAGE	3NN	Ada	C4.5	JR	MP	SVM	Chi	BSM
precision	0.27	0.33	0.25	0.27	0.22	0.00	0.08	0.21
recall	0.08	0.09	0.08	0.04	0.10	0.02	0.40	0.25
F1	0.12	0.14	0.12	0.07	0.13	0.01	0.12	0.21

real cases. Therefore, the proposed MSM and BSM may encounter difficulties when facing non-Gaussian data distributions. On the other hand, the data instances with the same labels tend to have some common local structures. For example, the projection of the instances that have the same labels as the target class could fall in the discretized interval which has a strong correlation with the target class. Therefore, the study should explore and include such common local structures in the classification rules to improve the performance of semantic information retrieval. In this chapter, a new classification framework that integrates both the global structure and the local structure is proposed.

The Propose Framework

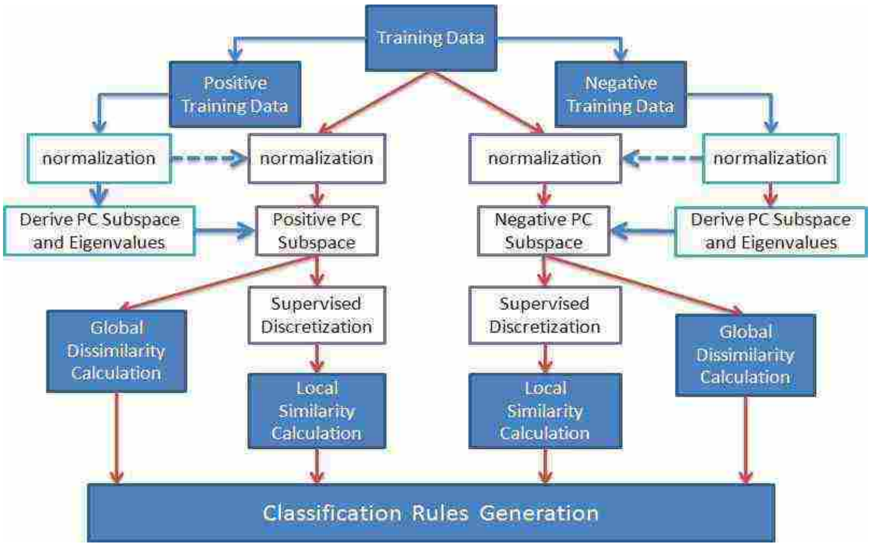

Training Data
Positive Training Data
Negative Training Data
normalization
normalization
normalization
normalization
Derive PC Subspace and Eigenvalues
Positive PC Subspace
Negative PC Subspace
Derive PC Subspace and Eigenvalues
Global Dissimilarity Calculation
Supervised Discretization
Supervised Discretization
Global Dissimilarity Calculation
Local Similarity Calculation
Local Similarity Calculation
Classification Rules Generation

Figure 3.7: Training phase

The training phase and testing phase of the proposed framework, subspace modeling on global and local structure (SMGL), are shown in Figure 3.7 and Figure 3.8, respectively. The framework is designed for a binary-class classification. Binary-class datasets are commonly seen in the semantic concept retrieval task, where one class represents a semantic concept (referred to as the target class or positive class) and the other class is formed by the instances which do not contain that semantic concept (referred to as the non-target class or negative class). As can be seen from the training phase, the training data are first divided into positive training data (labeled as positive class) and negative training data (labeled as negative class). Then, by following the basic idea of BSM, normalization and PC subspace derivation is included to calculate the global dissimilarity of the training data with regard to both the positive class and the negative class within the positive PC subspace and the negative PC subspace. To consider the local structure of the instances belonging to the positive (negative) class, the projected training data on the positive (negative) class are discretized into several intervals on all features, each of which forms the so-called feature-value pair.

Later, multiple correspondence analysis (MCA) measures the correlation between each feature-value pair and the two classes. Such a correlation is represented by the cosine value of the angle between the feature-value pair and the classes' projection on the first two major principal components within the MCA. The correlation of feature-value pairs to the positive (negative) class indicates the local similarity of an instance to the positive (negative) class. Then classification rules are generated from the global dissimilarity and the local similarity of training instances towards the positive and negative classes.

In the testing phase (see Figure 3.8), each testing instance goes through normalization and is projected on the positive and the negative PC subspace, respectively. Then, the global dissimilarity to the positive and the negative classes is calculated from the projected data on the positive and negative PC subspaces. By converting the projected

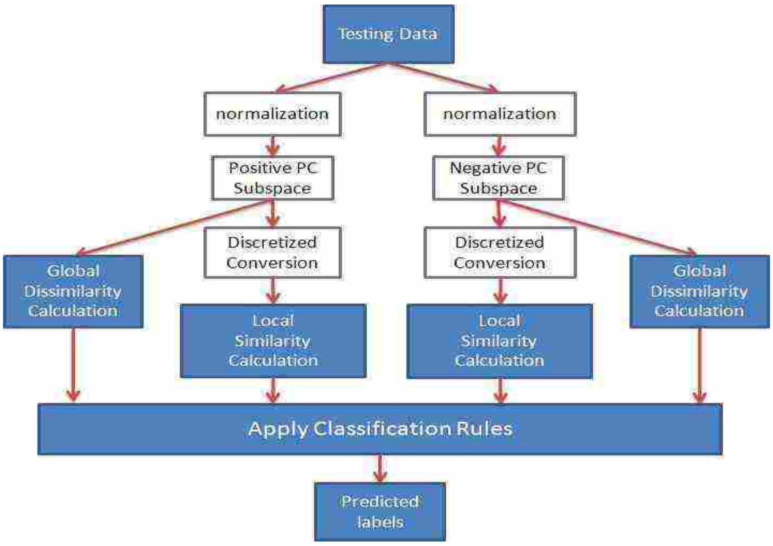

Testing Data
normalization
normalization
Positive PC Subspace
Negative PC Subspace
Global Dissimilarity Calculation
Discretized Conversion
Discretized Conversion
Global Dissimilarity Calculation
Local Similarity Calculation
Local Similarity Calculation
Apply Classification Rules
Predicted labels

Figure 3.8: Testing phase

data into nominal representation (feature-value pairs) and further into the correlations to the positive and the negative classes, the local dissimilarity is also derived. The classification rules are then applied on these global dissimilarity and local similarity values to predict the labels of the input instances.

Calculation of Global Dissimilarity

Let $X^{(p)}$=$\{X_1^{(p)}, X_2^{(p)}, \ldots, X_m^{(p)}\}$ be a set of positive training instances and $X^{(n)}$=$\{X_1^{(n)}, X_2^{(n)}, \ldots, X_t^{(n)}\}$ be a set of negative training instances. The whole training set X (X=$X^{(p)} \bigcup X^{(n)}$) is composed of the positive training set $X^{(p)}$ and the negative training set $X^{(n)}$, which contains m positive training instances and t negative training instances. Each instance in X is also represented by a feature vector. Assuming the dimensionality of the features is k, a positive instance $X_i^{(p)}$ is denoted as a vector $[X_{i1}^{(p)}, X_{i2}^{(p)}, \ldots, X_{ik}^{(p)}]$ and a negative instance $X_i^{(n)}$ is denoted as a vector $[X_{i1}^{(n)}, X_{i2}^{(n)}, \ldots, X_{ik}^{(n)}]$.

The objective of the normalization step is to prevent some features with large values from dominating those features with small values. In the proposed framework, Z-score normalization (shown in Equation (3.18) and (3.19)) is applied on $X^{(p)}$ and $X^{(n)}$.

$$Norm^{(p)} = \frac{X^{(p)} - \mu^{(p)}}{\sigma^{(p)}}. \tag{3.18}$$

$$Norm^{(n)} = \frac{X^{(n)} - \mu^{(n)}}{\sigma^{(n)}}. \tag{3.19}$$

$$Norm^{(p)}_{train} = \frac{X - \mu^{(p)}}{\sigma^{(p)}}. \tag{3.20}$$

$$Norm^{(n)}_{train} = \frac{X - \mu^{(n)}}{\sigma^{(n)}}. \tag{3.21}$$

where $\mu^{(p)}$ and $\sigma^{(p)}$ are the mean and standard deviation of $X^{(p)}$; $\mu^{(n)}$ and $\sigma^{(n)}$ are the mean and standard deviation of $X^{(n)}$.

For normalized positive instances and negative instances, a PC subspace is generated for each of them from the covariance matrix of the normalized instances. The eigenvalues and PC subspace are derived using singular value decomposition, as shown in Equation (3.22) and Equation (3.23) for the positive and negative instances.

$$Norm^{(p)'} * Norm^{(p)} = U^{(p)} * \Sigma^{(p)} * PC^{(p)'}. \tag{3.22}$$

$$Norm^{(n)'} * Norm^{(n)} = U^{(n)} * \Sigma^{(n)} * PC^{(n)'}. \tag{3.23}$$

For positive instances, let $\lambda^{(p)}$ ($\lambda^{(p)}$ = $[\lambda^{(p)}_1, \ldots, \lambda^{(p)}_{\rho^{(p)}}]$) be the sorted positive diagonal values of $\Sigma^{(p)}$ in a descending manner and the corresponding eigenvectors are denoted as $PPC^{(p)}$ ($PPC^{(p)}$=$[PC^{(p)}_1, \ldots, PC^{(p)}_{\rho^{(p)}}]$). Likewise, $\lambda^{(n)}$ ($\lambda^{(n)}$ = $[\lambda^{(n)}_1, \ldots, \lambda^{(n)}_{\rho^{(n)}}]$) and $PPC^{(n)}$ ($PPC^{(n)}$=$[PC^{(n)}_1, \ldots, PC^{(n)}_{\rho^{(n)}}]$) are defined for the negative instances. The training instances are then projected on the positive subspace spanned by $PPC^{(p)}$ and the negative subspace spanned by $PPC^{(n)}$ to get the projected training instances on positive and negative subspaces, as shown in Equation (3.24) and Equation (3.25).

$$\eta^{(p)} = Norm^{(p)}_{train} * PPC^{(p)}. \tag{3.24}$$

$$\eta^{(n)} = Norm^{(n)}_{train} * PPC^{(n)}. \tag{3.25}$$

where $Norm_{train}^{(p)}$ and $Norm_{train}^{(n)}$ is calculated by (3.20) and (3.21), respectively. The global dissimilarity with regard to the positive class and the negative class is derived as shown in Equation (3.26) and Equation (3.27).

$$\Omega_i^{(p)} = \sum_{j=1}^{\rho^{(p)}} \frac{\eta_{ij}^{(p)^2}}{\lambda_j^{(p)}}. \tag{3.26}$$

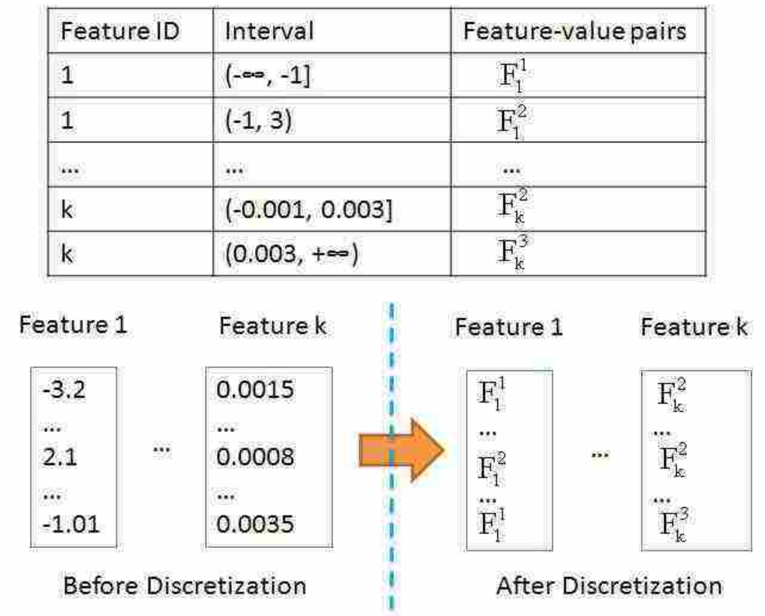
$$\Omega_i^{(n)} = \sum_{j=1}^{\rho^{(n)}} \frac{\eta_{ij}^{(n)^2}}{\lambda_j^{(n)}}. \tag{3.27}$$

$$\tag{3.28}$$

where $\eta_{ij}^{(p)}$ is the projection on the j-th PC of the i-th instances in the training set (X_i). For the positive instances, $\Omega_i^{(p)}$ is distributed close to 0.For the negative instances, $\Omega_i^{(n)}$ is distributed close to 0. Therefore, $\Omega_i^{(p)}$ and $\Omega_i^{(n)}$ shows the dissimilarity of X_i to the positive class and the negative class from the perspective of the characteristics of the positive class and negative class. The global dissimilarity is shown to be effective in classifying binary dataset in previous sections (MSM and BSM). This chapter will integrate local similarity with the global dissimilarity to form the classification rules.

Calculation of Local Similarity

The local similarity is calculated based on the projected data within the positive PC subspace and the negative PC subspace. From the early description, $\eta^{(p)}$ and $\eta^{(n)}$ are the projection of the normalized training instances on positive PC subspace and negative PC subspace. To capture the local structure that may have correlation with the classes, a discretization method is necessary to partition each feature into several intervals representing the range of continuous values, which are also called **feature-value pairs**. There are several discretization methods to be chosen. Broadly speaking, these discretization methods fall into two categories: unsupervised discretization and supervised discretization. A simple unsupervised discretization method is “equal interval width" that divides the values of a feature into M equal-sized bins, where parameter M is defined by the users. The unsupervised methods ignore the class distribution on the continuous feature values and therefore the classification information will be lost by partitioning a continuous range into several intervals. Therefore, a supervised discretization method is adopted in the discretization step in the proposed framework. Among them, a minimum description length (MDL)-based method proposed in [111] is employed to discretize $\eta^{(p)}$ and $\eta^{(n)}$, where *minimum description length principal* is used to determine the stopping criteria for the recursive discretization steps. Figure 3.9 shows an example of discretization and conversion from numeric values to feature-value pairs. As can be seen from the figure, there is a mapping table recording the feature-value pairs and their corresponding intervals, which can be used to convert the numeric values into nominal values. Since a feature-value pair may have different correlations with the classes, which implies the local structure in terms of these feature-value pairs of the instances contains classification information (such as local similarity), it requires



Feature ID	Interval	Feature-value pairs
1	(-∞, -1]	F_1^1
1	(-1, 3)	F_1^2
...	...	...
k	(-0.001, 0.003]	F_k^2
k	(0.003, +∞)	F_k^3

Figure 3.9: An example of feature discretization

a correlation-based method to derive and measure such correlations. On the other hand, multiple correspondence analysis (MCA) is commonly applied to capture the correlation between multiple nominal variables. Therefore, the proposed framework utilizes MCA to measure the correlation between the feature-value pairs and the positive (or negative) class. Furthermore, such a correlation is defined as the local similarity towards the positive (or negative) class. The procedure to derive the correlation between the feature-value pairs and the classes is shown as follows.

DERIVING CORRELATION USING MCA

```
FOR each feature
    Construct an indicator matrix $Z$ with columns representing feature-value pairs
      and also class labels and rows representing instances.
    Generate probability matrix $P = B/N$ from Burt matrix $B = Z^T Z$, where $N$ is
      the grand total of Burt matrix $B$.
    Derive the residual matrix $S = D^{-1/2}(P - V^T V)D^{-1/2}$, where $V$ is the column
      total of $P$ and $D$ is a diagonal matrix whose diagonal elements equals to those
      in $V$.
    Apply singular value decomposition to S using $U \Sigma Q^T = SVD(S)$.
    Project $S$ on the subspace spanned by first two major principal components
      of $Q$ as $R$.
    Output the cosine value of the angle between each feature-vector pair and each
      class as the correlation value.
END
```

Figure 3.10 shows an example of the projected data (R) in a two-dimensional positive PC subspace (the first and second principal components are selected). The angle between the feature-vector pair F_1^1 (F_1^2) and the positive class is denoted as α (β). The correlation of F_1^1 (F_1^2) with the positive class is the cosine value of α (β). It is clear that the correlation between F_1^1 and the positive class is negative to the correlation between F_1^1 and the negative class. In the proposed framework, the correlation of each feature-value pair towards the positive class is concerned within the positive PC subspace while the correlation of each feature-value pair towards the negative class is concerned within the negative PC subspace.

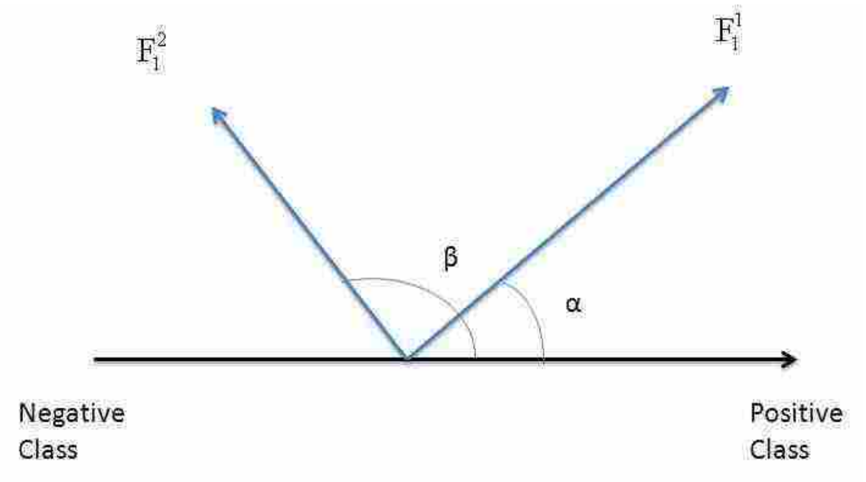
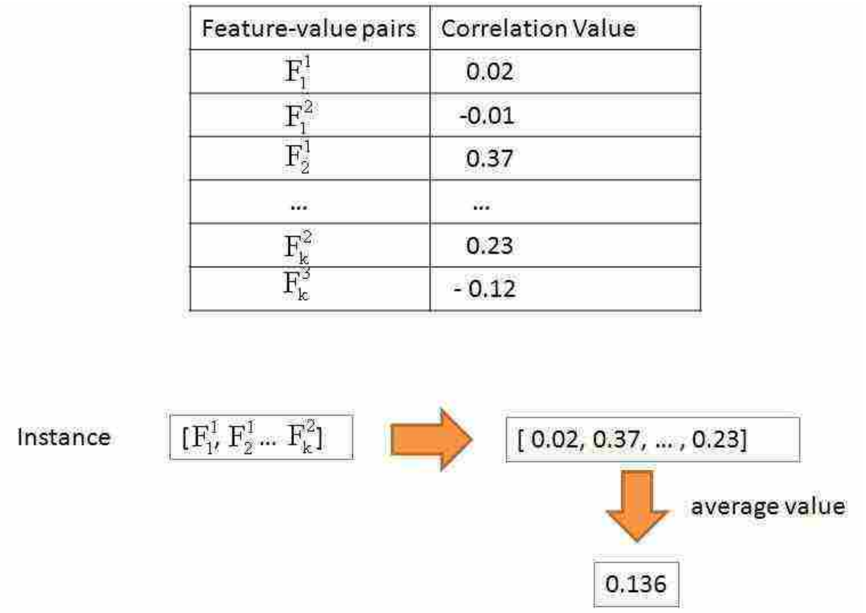


Figure 3.10: Correlation in terms of angles

Let W_i^j be the correlation of F_j^i with the positive class within positive subspace which is created by MCA. For each instance with a vector of feature-value pairs, the correlation value for a feature-value pair F_j^i can be looked up by searching the corresponding W_i^j. Then, the average of all the correlation values are defined as the **local similarity** in that an instance with a larger average correlation value indicating the higher probability that it belongs to a positive class. An example of calculating such local similarity is shown in Figure 3.11. The local similarity can be interpreted as a value to show the likelihood of an instance belonging to a positive class on average. For an instance i, the local similarity to the positive class in the positive PC subspace is defined as $M_i^{(p)}$, and the local similarity to the negative class in the negative PC subspace is defined as $-M_i^{(n)}$. Please note $M_i^{(n)}$ is the local similarity to the positive class in the negative PC subspace, which holds a value negative to the local similarity to the negative class, as implied in Figure 3.10.



Feature-value pairs	Correlation Value
F_1^1	0.02
F_1^2	-0.01
F_2^1	0.37
...	...
F_k^2	0.23
F_k^3	- 0.12

Figure 3.11: Calculate the local similarity of an instance

Generation of Classification Rules

The core of the proposed framework is to integrate the global dissimilarity and local dissimilarity of the instances into the classification rules. The classification methods proposed previously (such as MSM and BSM) only consider the global dissimilarity to generate the classification rules. In the proposed framework, the local dissimilarity is also taken into consideration. The proposed classification rules are shown as follows:

Classification Rules

```
if $\frac{M_i^{(p)}}{\Omega_i^{(p)}} > -\frac{M_i^{(n)}}{\Omega_i^{(n)}} + \beta$
    predict instance $i$'s class label as positive.
else
    predict instance $i$'s class label as negative.
end
```

The item $\frac{M_i^{(p)}}{\Omega_i^{(p)}}$ is the similarity of an instance i to the positive class from the perspective of global ($\frac{1}{\Omega_i^{(p)}}$) and local ($M_i^{(p)}$) similarity. The smaller the $\Omega_i^{(p)}$ is, the larger global similarity an instance i holds. Similarly, the item $-\frac{M_i^{(n)}}{\Omega_i^{(n)}}$ is the similarity of an instance i to the negative class by including both the global and local similarities to the negative class. The parameter β is a bias added to improve the classification performance, which can be determined by searching the β corresponding to the maximum classification performance (in terms of F1) of a positive class. Like MSN and BSM, SMGL holds a time complexity of $O(N^2)$ at the training phase and $O(1)$ at the testing phase.

Experiment

To validate the effectiveness of the proposed framework, experiments are conducted on several binary datasets from TRECVID2008 video collections. Forty-eight dimensional features are selected for different semantic concepts from a total number of 513 audio-visual features, including color histogram, edge histogram, wavelet texture and spectrum flux, and etc. Table 3.11 shows a list of binary datasets used in the experiments and their sources.

The proposed framework's performance (SMGL) is measured against several other well-known classification frameworks, including nearest neighbor (NN), K-nearest neigh-

Table 3.11: Binary datasets and their sources

concept name	source
two_people	TRECVID2008
driver	TRECVID2008
street	TRECVID2008
mountain	TRECVID2008
flower	TRECVID2008

Table 3.12: Performance of classification on concept "two_people"

Classifier	Precision	Recall	F1-score
SMGL	0.34	0.89	0.49
NN	0.40	0.42	0.41
3NN	0.42	0.34	0.38
MP	0.44	0.40	0.42
Ada	0.48	0.37	0.42
Chi	0.32	0.44	0.37
JRip	0.52	0.20	0.29
J48	0.49	0.31	0.37

Table 3.13: Performance of classification on concept "driver"

Classifier	Precision	Recall	F1-score
SMGL	0.12	0.27	0.17
NN	0.15	0.14	0.15
3NN	0.21	0.04	0.07
MP	0.17	0.07	0.10
Ada	0.32	0.07	0.12
Chi	0.06	0.43	0.10
JRip	0.00	0.00	0.00
J48	0.39	0.06	0.10

Table 3.14: Performance of classification on concept “street”

Classifier	Precision	Recall	F1-score
SMGL	0.22	0.56	0.31
NN	0.26	0.26	0.26
3NN	0.32	0.17	0.22
MP	0.37	0.14	0.20
Ada	0.41	0.21	0.28
Chi	0.14	0.40	0.20
JRip	0.42	0.06	0.11
J48	0.34	0.18	0.23

Table 3.15: Performance of classification on concept “mountain”

Classifier	Precision	Recall	F1-score
SMGL	0.09	0.44	0.15
NN	0.13	0.13	0.13
3NN	0.16	0.04	0.06
MP	0.16	0.08	0.10
Ada	0.26	0.05	0.09
Chi	0.06	0.42	0.11
JRip	0.10	0.01	0.01
J48	0.14	0.04	0.06

Table 3.16: Performance of classification on concept “flower”

Classifier	Precision	Recall	F1-score
SMGL	0.11	0.36	0.17
NN	0.17	0.16	0.16
3NN	0.24	0.06	0.10
MP	0.15	0.09	0.11
Ada	0.22	0.04	0.06
Chi	0.06	0.51	0.11
JRip	0.00	0.00	0.00
J48	0.17	0.01	0.02

Table 3.17: Average performance of classification on all concepts

Classifier	Precision	Recall	F1-score
SMGL	0.18	0.51	0.26
NN	0.22	0.22	0.22
3NN	0.27	0.13	0.17
MP	0.26	0.16	0.18
Ada	0.34	0.15	0.19
Chi	0.13	0.44	0.18
JRip	0.21	0.05	0.08
J48	0.30	0.12	0.16

bor when K=3 (3NN), multilayer perceptron (MP), support vector machines with chi-square kernels (Chi), rule based JRip (JRip), decision tree (J48), and AdaBoost with decision tree kernel (Ada), all of which are implemented by Weka [112] with parameter tuning, aiming to have a fair comparison. Three-fold cross-validation is conducted and the precision, recall, and F1-score (performance metrics) are adopted to evaluate the performance of all classifiers.

Table 3.12 to Table 3.17 display the experimental results of all classifiers. As can be seen from the results, the proposed framework is able to render better performance than the comparative approaches in terms of F1. For example, the proposed framework is at least 7% F1 better than the other classification frameworks for concept "two_people" and on average about 4% better than the other classification frameworks for all the concepts.

3.3 Subspace Modeling for Imbalanced Data

Like other popular supervised classification algorithms, the MSM and BSM methods are not specifically designed for imbalanced data. Therefore, both of them will suffer from a data imbalance problem. To balance the ratio between the majority class (usually negative class) and minority class (usually positive class), clustering-based subspace modeling methods are proposed to cluster the negative class (majority class) into several negative data groups. In this manner, the ratio between each negative data group and the positive class is more balanced than the original data set.

The advantages of such a clustering-based subspace modeling are as follows:

- The ratio between the majority class and minority class is decreased.
- No information is lost during the process of handling data imbalance.
- The training model of BSM can benefit from the clustering of the majority class.

The first two advantages are quite obvious. The third advantage can be seen when tackling an imbalanced data set in which the majority class holds a large intra-class variance and does not follow the normal data distribution. As implied by MSM and BSM, which assume each class as a group of data that follow approximately a normal distribution, such a characteristic of the majority class violates this assumption and thus may compromise the classification performance of subspace modeling. However, after clustering the majority class into several data groups, each data group has a smaller intra-class variance. In addition, the data distribution of each group is closer to the Gaussian distribution when its variance becomes smaller and smaller.

3.3.1 Clustering-based Subspace Modeling

Figure 3.12 shows the overall framework of CLUstering-based SUbspace MOdeling (CLU-SUMO) [92]. In this framework, the negative training data set TrN is clustered into K groups, namely $TrN^{(1)}, \ldots, TrN^{(K)}$. Each of the K groups is combined with TrP to from Group 1, ..., Group K, as shown in Figure 3.12. Each Group K is modeled by a BSM. In the classification phase, the ranking scores of a testing data instance from BSMs of all groups and the original data set are then integrated using different weights. The label of that testing data instance is then predicted by checking if the integrated

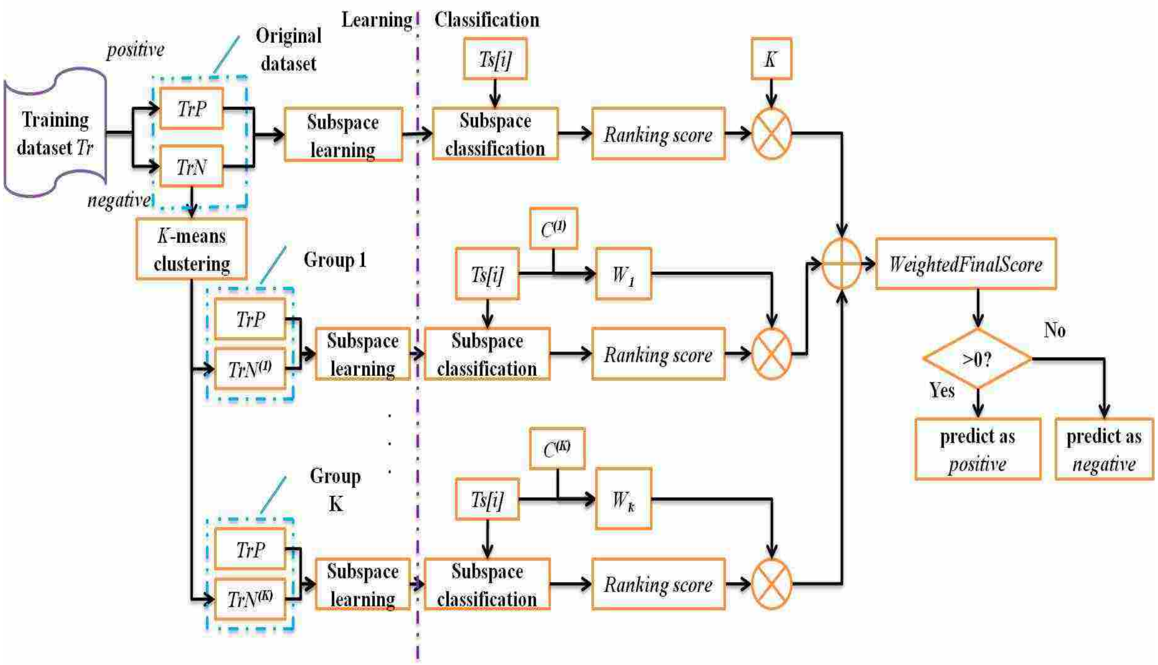


Figure 3.12: The clustering-based subspace modeling

score is greater than zero or not. The learning and classification of CLU-SUMO is shown in CODE 3.1.

CODE 3.1: CLU-SUMO: LEARNING & CLASSIFICATION

```
**Learning Step**:
Divide training data set $Tr$ into positive set $TrP$ and negative set $TrN$.
Apply $K$-means method to cluster $TrN$ into $K$ clusters $TrN^{(1)}, \ldots, TrN^{(K)}$.
Derive $C^{(1)}, \ldots, C^{(K)}$, which are the centroids of $TrN^{(1)}, \ldots, TrN^{(K)}$.
Build Group $j$ by combining $TrN^{(j)}$ with $TrP$, $j = 1, \ldots, K$.
Apply subspace learning on the original data set as well as on Group 1 to Group $K$.
**Classification Step**:
For a testing data instance $Ts[i]$, apply subspace classification using the
    parameters from subspace learning models and get the ranking scores[a] from all
    binary-class subspace models.
Derive the weight for Group $j$ using $W_j = exp(-1 * ||Ts[i] - C^{(j)}||)$, $j = 1, \ldots, K$.
Calculate $WeightedFinalScore$ by combining the weighted ranking scores
    (ranking score multiplied by a weight) from all subspace models.
Predict $Ts[i]$ as positive, if $WeightedFinalScore > 0$.
Predict $Ts[i]$ as negative, otherwise.
```

[a] the ranking score can refer to BSM.

Experiment Setup

The data sets used in the experiments are from the MediaMill Challenge Problem [4], which uses eighty-five hours of news and broadcast video data. There are five experiments in the challenge problem and the training and testing data sets in Experiment 1 are used in the experiments. The training data set consists of $30,993$ data instances and 120 attributes, while the testing data set has $12,914$ data instances. The positive and negative ratios for the concepts used in the experiment are shown in Table 3.18.

Five concepts are selected with positive to negative ratio between 0.043 to 0.074. Therefore, these data sets are very imbalanced and thus suitable to prove the effectiveness of the proposed framework.

Table 3.18: The positive and negative training instance ratios for concepts

ID	Concept	Positives (P)	Negatives (N)	P-to-N ratio
16	building	2126	28867	0.074
19	car	1509	29484	0.051
20	meeting	1405	29588	0.048
21	female	1359	29634	0.046
22	military	1283	29710	0.043

.

Table 3.19: Performance of classification on concept “building”

Classifier	Precision	Recall	F1
CLU-SUMO	0.28	0.56	0.37
BSM	0.33	0.34	0.33
SVM	0.45	0.19	0.27
NB	0.20	0.54	0.30
NN	0.29	0.19	0.23
KNN (K=3)	0.24	0.38	0.29
Ada	0.33	0.17	0.22
C4.5	0.29	0.28	0.28
MP	0.37	0.31	0.34

Table 3.20: Performance of classification on concept “car”

Classifier	Precision	Recall	F1
CLU-SUMO	0.25	0.32	0.28
BSM	0.43	0.19	0.26
SVM	0.34	0.24	0.28
NB	0.08	0.56	0.14
NN	0.28	0.25	0.27
KNN (K=3)	0.18	0.36	0.24
Ada	0.41	0.18	0.25
DC4.5	0.23	0.24	0.24
MP	0.28	0.20	0.23

In the experiments, all classifiers take the same training and testing data sets and the performance from all classifiers is evaluated in terms of F1-score, which is the harmonic mean of precision and recall.

For BSM and CLU-SUMO, the *init_value*, *end_value*, and step s of β for the framework are selected as -3, 3, and 0.02, respectively, in the learning phase of subspace modeling. To balance the positive and negative classes in the generated data groups, K is chosen to be 5 in the experiments so that the positive to negative ratio is, on average, a little more than 1/5. With regard to the classification algorithms used for performance comparison, a list of popular approaches such as support vector machines (SVM), naive Bayes (NB), nearest neighbor (NN), K-Nearest Neighbor (KNN), AdaBoost with C4.5 algorithm (Ada), C4.5 algorithm (C4.5), and multilayer perceptron (MP) available in Weka [112] are used. These classifiers produce the probability that a testing data instance belongs to the positive class, which is defined as the probability of positiveness (PoP) in this dissertation. Correspondingly, probability of negativeness (PoN) is defined to describe the probability that a data instance belongs to the negative class. The classification rules based on the PoP are shown as follows:

$$\begin{cases} \text{IF PoP} \geq \text{PoN}, \text{THEN assign positive label} \\ \text{IF PoP} < \text{PoN}, \text{THEN assign negative label} \end{cases}$$

or in an equivalent form:

$$\begin{cases} \text{IF PoP} \geq 0.5, \text{THEN assign positive label} \\ \text{IF PoP} < 0.5, \text{THEN assign negative label} \end{cases}$$

In response to the data imbalance issue, an adaptive threshold τ is used instead of 0.5 to achieve an equivalent effect as the “reweighting” method. Therefore, the classification rule is modified as follows.

$$\begin{cases} \text{IF PoP} \geq \tau, \text{THEN assign positive label} \\ \text{IF PoP} < \tau, \text{THEN assign negative label} \end{cases}$$

Table 3.21: Performance of classification on concept “meeting”

Classifier	Precision	Recall	F1
CLU-SUMO	0.28	0.29	0.29
BSM	0.39	0.19	0.25
SVM	0.34	0.25	0.28
NB	0.07	0.84	0.12
NN	0.16	0.16	0.16
KNN (K=3)	0.13	0.33	0.19
Ada	0.25	0.17	0.20
C4.5	0.22	0.16	0.19
MP	0.22	0.35	0.27

Table 3.22: Performance of classification on concept “female”

Classifier	Precision	Recall	F1
CLU-SUMO	0.15	0.22	0.18
BSM	0.17	0.15	0.16
SVM	0.18	0.11	0.14
NB	0.03	0.68	0.06
NN	0.08	0.15	0.10
KNN (K=3)	0.06	0.32	0.11
Ada	0.18	0.08	0.11
C4.5	0.08	0.14	0.11
MP	0.11	0.21	0.14

τ is searched from 0.1 to 1 with a small step size 0.02 for all the aforementioned comparative algorithms to get their best F1-scores on the testing data. In this way, it is believed to be more reasonable and fair to compare the proposed framework with these comparative methods using an adaptive threshold.

Experimental Results and Analyses

The experimental results are shown from Table 3.19 to Table 3.23. The results reveal that the proposed framework CLU-SUMO is better than or as good as all comparative approaches with regard to all concepts used in the experiments. Table 3.24 shows that on average CLU-SUMO is at least 3% better than the other comparative methods. Because of the low F1-scores in the experiments, the 3% improvement is quite valuable. Another

Table 3.23: Performance of classification on concept “military”

Classifier	Precision	Recall	F1
CLU-SUMO	0.26	0.35	0.30
BSM	0.29	0.20	0.24
SVM	0.35	0.17	0.23
NB	0.11	0.70	0.20
NN	0.21	0.16	0.18
KNN (K=3)	0.17	0.30	0.22
Ada	0.28	0.08	0.13
C4.5	0.18	0.25	0.21
MP	0.28	0.26	0.27

contribution of CLU-SUMO is shown in Tables 3.19, 3.21, and 3.23. If the subspace model is trained on the original data set alone, the performance in terms of F1-score may be inferior to multilayer perceptron for some data sets. However, if clustering-based subspace modeling is applied, the performance is the best among all the compared classification algorithms. This improvement is from the weighted voting of these $K+1$ subspace models. K subspace models are created on K new data groups, which are more balanced than the original data set. These learning models may capture better positive class patterns than the model trained by only the original data set. However, this statement is true only if K is appropriately selected. If K is too small, then the improvement is not obvious. On the other hand, if K is too large, then within some new data groups, the minority class could now be the negative class and the subspace models may be overfitting to the positive class.

3.3.2 Integration of Class Selection and Clustering for Binary-class Subspace Modeling

In Chapter 3.3.1, a clustering-based subspace modeling method called CLU-SUMO was proposed. CLU-SUMO utilizes K-means clustering to build K negative data groups from the original negative training subset. Each negative data group is combined with the original positive training subset to generate K training data groups. Subspace mod-

Table 3.24: Average F1 on all 5 concepts

Classifier	mean F1
CLU-SUMO	0.28
BSM	0.25
SVM	0.24
NB	0.16
NN	0.19
KNN (K=3)	0.21
Ada	0.18
C4.5	0.20
MP	0.25

eling method is used to build models on each training data group as well as the original imbalanced data set to predict the ranking score (soft label) for each testing data instance. Next, a combination of these ranking scores is compared with a decision threshold (the threshold is 0 in CLU-SUMO) to predict the final label of the testing data instance. The CLU-SUMO framework has improved the classification performance with the help of clustering the negative data instances.

Here, the CLU-SUMO classification framework is further enhanced by integrating semantics information and clustering in the construction of a set of balanced data groups to address the data imbalance issue for multimedia data. In CSC-SUMO, the following enhancements are achieved.

- Speed up the model training procedure. In the proposed framework, the clustering step is applied after some non-target concept classes are held out as negative data groups. The idea behind such a hold-out strategy is that those data instances of the non-target class usually share some common data characteristics and semantics. Since the purpose of applying a clustering method is to find data groups whose data instances share similar data characteristics, from the view of semantics, it is reasonable to regard each non-target concept class as one negative data group

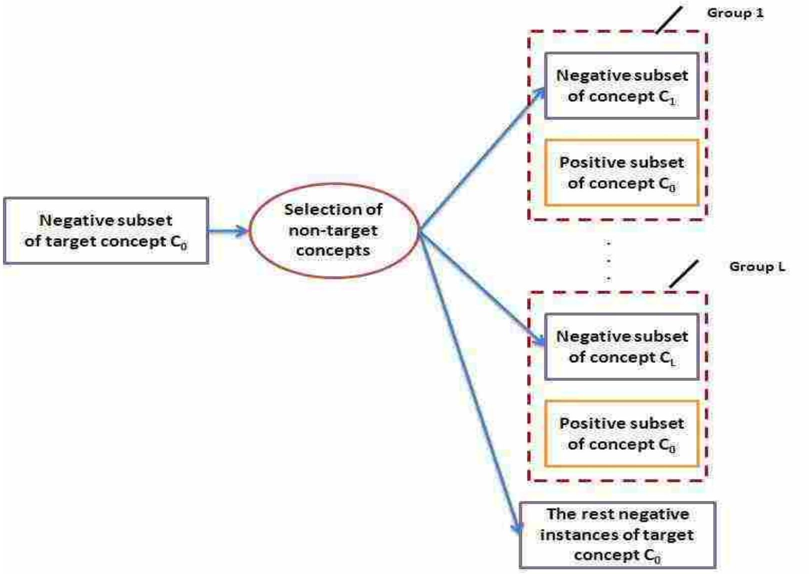

Negative subset of target concept C_0
Selection of non-target concepts
Group 1
Negative subset of concept C_1
Positive subset of concept C_0
Group L
Negative subset of concept C_L
Positive subset of concept C_0
The rest negative instances of target concept C_0

Figure 3.13: Generation of balanced training subsets by class selection

though the intra-group similarity cannot be guaranteed to be as small as the one generated by a clustering method.

- Some of the generated data groups hold semantic meanings. Each non-target class corresponds to a particular concept. Therefore, the generated rules that rely on these concepts can help to interpret their meanings. Furthermore, the semantic relationship between concepts can potentially be utilized to help improve the detection results of the target concept.

The proposed CSC-SUMO classification framework consists of three procedures: the generation of balanced training subsets by class selection (as shown in Figure 3.13), the generation of balanced training subsets by clustering (as shown in Figure 3.14), and integrated subspace modeling and classification (as shown in Figure 3.15).

During the first procedure, a number of non-target concepts are selected based on the following pre-defined criteria which are chosen via domain knowledge and empirical studies.

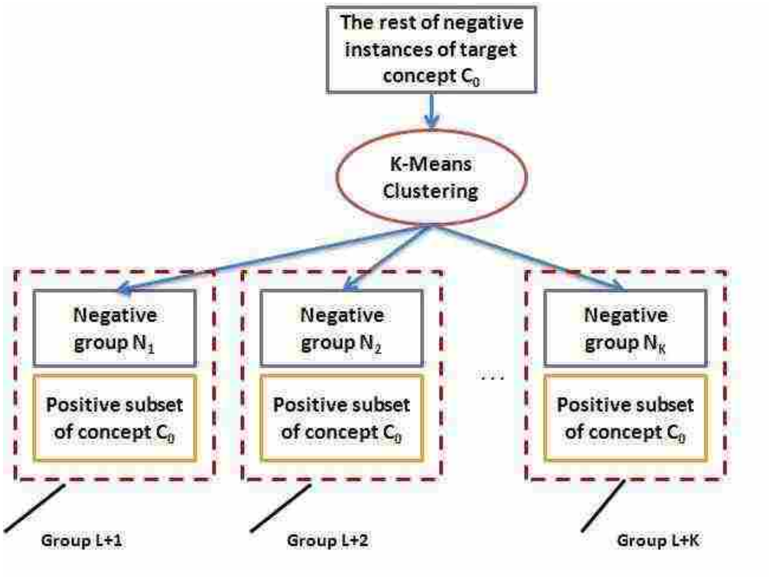

The rest of negative instances of target concept C_0
K-Means Clustering
Negative group N_1
Positive subset of concept C_0
Negative group N_2
Positive subset of concept C_0
...
Negative group N_K
Positive subset of concept C_0
Group L+1
Group L+2
Group L+K

Figure 3.14: Generation of balanced training subsets by clustering

- The ratio of the selected non-target concept class to the target concept class should fall within the interval of [0.5, 2].

- The overlapping of non-target concept class and target concept class should be below 1%.

- The overlapping between the selected non-target concept classes must below 50%.

The first criterion ensures that each group in Figure 1 is balanced. The second criterion requires the non-target concept class to overlap with the target concept as little as possible, considering that too much overlap could make it hard to learn separation rules from the generated balanced groups. The third criterion aims to reduce the number of groups generated by the first procedure. If the overlap between two non-target concept classes is large, then it is not necessary to generate a data group for each of them since one non-target concept class is already enough to describe the majority of the data instances belonging to the other concept class. Since the selected non-target

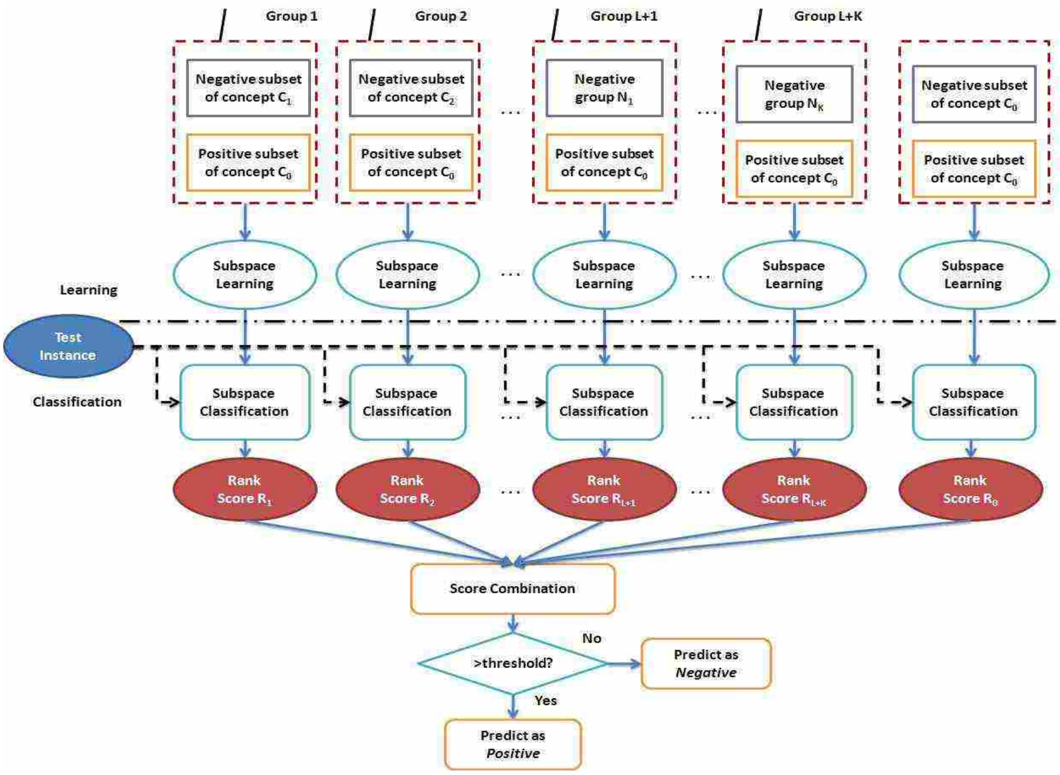


Figure 3.15: Integrated subspace modeling and classification

concept classes may not cover the whole negative subset of the target concept, the size of the remaining negative data instances should be small, which will be the input to the second procedure to cluster them into several data groups.

The advantages of utilization class label information lie in two areas: 1) the efficiency of the clustering-based binary classification framework is enhanced; 2) it facilitates to interpret the semantic meanings within the generated rules.

In the second procedure, the K-means clustering method is used to cluster the remaining negative data instances after the first procedure to form more data groups. This procedure is the same as the one proposed in CLU-SUMO. Until procedure 2, each negative data instance is assigned to one or more data groups since the data groups generated from the first procedure may have some overlapping negative data instances (i.e., those data instances belonging to two or more selected non-target concept classes).

All the balanced data groups and the original imbalanced training data set are trained and optimized by the subspace modeling method, as shown in Figure 3.15. The learning and classification (with the ranking scores) of the subspace modeling is briefly introduced in CODE 3.2 and CODE 3.3. Please note that the subspace modeling used here is a simplified version of BSM in order to speed up the training process. For example, to reduce the iterative loops, the weight parameter β is not used in the learning phase of BSM. The definitions of the functions used in CODE 3.2 and CODE 3.3 are shown as follows.

Definition 1 (function Z) *For an $m \times n$ matrix $A = a(i,j)$ and a $\rho \times n$ matrix B,*

$$Z(B,A) = \begin{bmatrix} (B(1,:) - \mu(A))/s(A) \\ . \\ . \\ . \\ (B(\rho,:) - \mu(A))/s(A) \end{bmatrix}$$

where $\mu(A)$=$[\mu_1(A),\ldots,\mu_n(A)]$ is calculated by Equation (3.29) and $s(A)$=$[s_1(A),\ldots,s_n(A)]$ is calculated by Equation (3.30). It can be observed that $Z(A,A)$ is the z-score normalization of A.

$$\mu_j(A) = \frac{1}{m}\sum_{i=1}^{m} a(i,j), j = 1,2,\ldots,n \qquad (3.29)$$

$$s_j(A) = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(a(i,j) - \mu_j(A))^2}, j = 1,2,\ldots,n \qquad (3.30)$$

CODE 3.2: SUBSPACE MODELING: TRAINING PHASE

```
Input:
    (1) A set of training data instances $Tr$
    (2) Training labels
Output: $pl^{(opt)}$, $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$,$s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$,
    $PC(TrP)$, $PC(TrN)$
________________________________
Divide training data set $Tr$ into positive class $TrP$ and negative class $TrN$
    according to the training labels.
Apply normalization function Z and SVD to positive and negative classes and
    derive the projected data $PCP(Tr, PC(TrP))$ and $PCP(Tr, PC(TrN))$.
Iteratively search $pl$ to optimize the F1 Score of the learning model.
Output $pl^{(opt)}$ corresponding to the best F1 score,$\mu(TrP)$, $\mu(TrN)$, $s(TrP)$
    and $s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$, $PC(TrP)$ and $PC(TrN)$.
```

Definition 2 (SVD) *The standard SVD (singular value decomposition) is shown in Equation (3.31).*

$$A = U\Sigma V^T. \tag{3.31}$$

The SVD function of an $m \times n$ matrix A produces $\lambda(A)$ and $PC(A)$, where $\lambda(A)$ is the positive diagonal elements of $\Sigma^T\Sigma$ sorted in a descending manner. In other words, $\lambda(A)$={$\lambda_1(A)$,...,$\lambda_\theta(A)|\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_\theta(A) > 0\}$. $PC(A)$ is the eigenvectors from V that correspond to the sorted $\lambda(A)$.

Definition 3 (function PCP) *Suppose there are an $m \times n$ matrix $B = \{b(i, j)\}$ and eigenvectors $PC(A)$=$\{PC_1(A), \ldots, PC_\theta(A)\}$, where $PC_i(A)$ is an $n \times 1$ vector, i=1,.... The principal component projection (PCP) of B on PC(A) is defined as follows.*

$$PCP(B,A) = \{B * PC_1(A), \ldots, B * PC_\theta(A)\}. \tag{3.32}$$

Equation (3.32) shows that $PCP(B,A)$ is an $m \times \theta$ matrix. If $PCP_{(x,y)}(B,A)$ is used to denote the element of $PCP(B,A)$ at the x-th row and y-th column, then $PCP_{(x,y)}(B,\ A)$ is the projection of the x-th row vector of B on y-th eigenvector of $PC(A)$.

Definition 4 *Based on Definitions 2 and 3, it can further defines the score function* $Score(B,A,pl)$=$[Score_1(B,A,pl),\ \ldots,\ Score_x(B,A,pl),\ \ldots,\ Score_m(B,A,pl)]^T$*, where* $Score_x(B,A,pl)$ *is defined in Equation (3.33).*

$$Score_x(B,A,pl) = \sum_{y=1}^{pl} \frac{PCP_{(x,y)}(B,A) \times PCP_{(x,y)}(B,A)}{\lambda_y(A)}, \tag{3.33}$$

where pl can be any integer between 1 *and* θ*.*

CODE 3.3: SUBSPACE MODELING: CLASSIFICATION PHASE

```
Input:
(1) Testing data instance $Ts[i]$, $i$=1 to $\omega$ (the total number of testing
data instances)
(2) Output from the learning phase: $pl^{(opt)}$, $\mu(TrP)$, $\mu(TrN)$, $s(TrP)$,
$s(TrN)$, $\lambda(TrP)$, $\lambda(TrN)$, $PC(TrP)$, $PC(TrN)$
Output: ranking score of $Ts[i]$

Derive the projected testing data instance by applying $Z$ function and subspace
    projection to calculate the $Score(Ts[i],\ TrP,\ pl^{(opt)})$ and $Score(Ts[i],\ TrN,\ pl^{(opt)})$.
Let $S$=$Score(Ts[i],\ TrN,\ pl^{(opt)})$+$Score(Ts[i],\ TrP,\ pl^{(opt)})$
Output ($Score(Ts[i],\ TrN,\ pl^{(opt)})$-$Score(Ts[i],\ TrP,\ pl^{(opt)})$)/$S$ as the ranking
    score of $Ts[i]$.
```

For a testing data instance *Ts[i]*, the generated ranking scores from the subspaces are combined by a score combination module to produce a final ranking score. The final ranking score $R_{final}[i]$ of *Ts[i]* is calculated by Equation (3.34).

$$R_{final}[i] = (L+K) \cdot R_0 + \sum_{j=1}^{L+K} e^{(-(1+||Ts[i]-C_j||))} \cdot R_j, \tag{3.34}$$

Table 3.25: The positive and negative training instance ratios for concepts

concept	Positives (P)	Negatives (N)	P-to-N ratio
car	1509	29484	0.051
military	1283	29710	0.043
vegetation	1198	19795	0.040
sports	1166	29827	0.039
graphics	897	30096	0.030
people_marching	597	30396	0.020
soccer	517	30476	0.017
screen	475	30518	0.016

where C_j is the centroid of the j-th negative data group generated either from the first or the second procedure, and $||\cdot||$ stands for the norm operation. If $R_{final}[i]$ is larger than a threshold value, then *Ts[i]* is predicted as positive. Otherwise, *Ts[i]* is predicted as negative. In the experiment, this threshold value is set to 0.

Experiment Setup

The data sets used for the experiment are from news and broadcast videos [4]. The training data set and testing data set are divided in advance by the provider. The training data set is composed of a total of $30,993$ data instances with 120 attributes, while there are $12,914$ data instances with the same number of attributes. A number of concepts corresponding to an imbalanced binary-class data set are selected. The information related to these concepts is shown in Table 3.25. The positive-to-negative (P-to-N) ratio of these concepts varies between 0.016 to 0.051. Therefore, such imbalanced data sets are suitable to evaluate the effectiveness of the proposed framework.

In the experiment, all classifiers take the same training and testing data sets and the performance from all classifiers is evaluated in terms of F1 score. For CSC-SUMO, *K* is carefully chosen so that the ratio of the positive data instances to the negative data instances is on average 1:2, balancing the positive and negative classes in the generated data groups.

Table 3.26: Performance of classification on concept “car”

Classifier	Precision	Recall	F1
CSC-SUMO	20.28%	45.56%	28.07%
AdaBoost-C4.5	48.20%	12.00%	19.20%
CostDTree	18.30%	21.70%	19.80%
ResampleLG	28.39%	26.76%	27.55%

Table 3.27: Performance of classification on concept “military”

Classifier	Precision	Recall	F1
CSC-SUMO	23.98%	42.24%	30.59%
AdaBoost-C4.5	32.60%	5.30%	9.10%
CostDTree	17.90%	17.10%	17.50%
ResampleLG	16.32%	68.59%	26.37%

With regard to the classification algorithms used for performance comparison, a list of popular approaches such as AdaBoost with C4.5 algorithm (AdaBoost-C4.5), cost-sensitive decision tree (CostDTree) and classic re-sampling method are used, which are available in Weka [112]. The cost matrix CM used by CostDTree is shown below.

$$CM = \begin{bmatrix} 0 & 1 \\ \omega & 0 \end{bmatrix}$$

where ω is set to the negative-to-positive ratio for each target concept. For a re-sampling method, a logistic regression model is trained on the re-sampled training data set and later is used for predicting the class labels of testing data set. This method is denoted as re-sampling with logistic regression model (ResampleLG). The re-sampling percentage is tuned according to different data sets.

Experimental Results

The experimental results on the selected 8 concepts are shown from Table 3.26 to Table 3.33. The average F1 scores for all classifiers including CSC-SUMO are shown in Table 3.34. The results show that the proposed CSC-SUMO framework outperforms all the comparative approaches in terms of F1 measure (about 10% to 16% improvement).

Table 3.28: Performance of classification on concept “vegetation”

Classifier	Precision	Recall	F1
CSC-SUMO	10.50%	67.11%	18.16%
AdaBoost-C4.5	39.10%	7.20%	12.10%
CostDTree	14.00%	16.40%	15.10%
ResampleLG	37.08%	11.02%	16.99%

Table 3.29: Performance of classification on concept “sports”

Classifier	Precision	Recall	F1
CSC-SUMO	24.62%	33.23%	28.28%
AdaBoost-C4.5	58.50%	11.30%	18.90%
CostDTree	11.50%	20.80%	14.80%
ResampleLG	18.50%	31.45%	23.30%

Table 3.30: Performance of classification on concept “graphics”

Classifier	Precision	Recall	F1
CSC-SUMO	45.69%	60.13%	51.92%
AdaBoost-C4.5	75.60%	28.30%	41.20%
CostDTree	34.00%	37.40%	35.60%
ResampleLG	35.35%	50.78%	41.86%

Table 3.31: Performance of classification on concept “people_marching”

Classifier	Precision	Recall	F1
CSC-SUMO	30.79%	24.95%	27.57%
AdaBoost-C4.5	36.70%	3.40%	6.20%
CostDTree	18.20%	18.90%	18.50%
ResampleLG	9.89%	75.23%	17.48%

Table 3.32: Performance of classification on concept “soccer”

Classifier	Precision	Recall	F1
CSC-SUMO	71.88%	60.53%	65.71%
AdaBoost-C4.5	75.00%	55.30%	63.60%
CostDTree	9.10%	65.80%	15.90%
ResampleLG	12.80%	42.11%	19.63%

Table 3.33: Performance of classification on concept “screen”

Classifier	Precision	Recall	F1
CSC-SUMO	64.15%	13.88%	22.82%
AdaBoost-C4.5	82.40%	5.70%	10.70%
CostDTree	6.00%	11.40%	7.90%
ResampleLG	9.62%	23.67%	13.68%

Table 3.34: Average F1 on all concepts

Classifier	mean F1
CSC-SUMO	34.14%
AdaBoost-C4.5	22.63%
CostDTree	18.14%
ResampleLG	23.34%

The performance of the ResampleLG method seems to be unstable. For example, for some concepts like “car”, the F1 value of ResampleLG is slightly worse (about 0.5%) than CSC-SUMO. However, for the concept “soccer”, the F1 value of ResampleLG is much smaller than CSC-SUMO. This is due to the fact that resampling methods often require an appropriate selection of sampling percentage, which has a large impact on the prediction quality of the classifiers. However, it is often hard to determine such an appropriate sampling percentage.

With regard to CostDTree, there is an inherent problem related to the configuration of the cost matrix. Similar to the re-sampling method, it is also difficult to build a cost matrix that can always render satisfactory classification results. The AdaBoost-C4.5 method is able to provide better results than CostDTree. However, it requires a time-consuming model training process to achieve better results. In a situation where the training time is a major concern, AdaBoost-C4.5 may have its limitations.

As shown previously, the weighted voting of all subspace models can improve the classification results. This is mainly because these K subspace models were built on balanced data sets, and each balanced learning model could learn the patterns belonging to the positive class accompanied by a proportion of data instances of the original negative subset. The selection of K definitely will have an influence on the final classification results. On one hand, a small K value helps little to improve the classification results in terms of F1 measure. On the other hand, a large K value may increase the number of involved training models and could cause an overfitting problem, since too few negative data instances are trained in each subspace model. In the training phase, the CSC-SUMO and CLU-SUMO both have a time complexity of $O(N^2)$ because of performing SVD. However, such a time complexity can be reduced to $O(1)$ at the testing phase.

3.4 Semantic Concept Retrieval using Inter-concept Relationships

The aforementioned classifiers mainly focus on individual model for a semantic concept. However, the semantic concepts in the real world are often inter-correlated with each other. Therefore, to improve further the performance of semantic concept retrieval based on the individual model, here several ranking strategies that utilize the inter-concept co-occurrence to retrieve a target semantic concept are proposed.

3.4.1 Semantic Concept Retrieval using Inter-concept Co-occurrence

There is a list of peer work [47, 46] mentioned in Chapter 2.3. The difference between the proposed framework and the peer work lies in the following aspects. First, previous work regards correlation information as mutually useful. In other words, it considers concept “A” and concept “B” as both target and reference concepts to each other under the assumption that concept “A” and concept “B” would both benefit from their correlation. However, this may not be true in reality since the difficulty to retrieve concepts “A” and “B” is not consistent. For example, although concepts “road” and “outdoor” have strong correlation, “road” is much more difficult to retrieve than “outdoor” as can be seen from [4]. Therefore, “road” may benefit from such a correlation from “outdoor” because the correlation information from “outdoor” is quite reliable. Unfortunately on the other side, “outdoor” may render worse performance if it utilizes the correlation information with “road”. Therefore, in the proposed work, the correlation information is utilized uni-directional. Only those easy-to-retrieve concepts are regarded as the reference concepts and of which the relationship will be used to refine the ranking of the retrieved results of the target concepts. Second, the information of co-occurrence between concepts is viewed in a mutual manner in previous work. That is, only when concept “A” and concept “B” both appear frequently, the relationship between “A” and “B” becomes valuable. However, this co-occurrence between concepts is viewed in an individual manner. As long as there is a large chance (e.g., 90%) that “B” will occur when “A” appears, this co-occurrence relationship from “B” is valuable and should be taken into consideration, no matter how low the chance of “A” would appear when “B” occurs. It is worth mentioning that previous work may miss a co-occurrence relationship that is not mutual, such as the co-occurrence relationship between “snow” and “outdoor”. Finally, previous work studied the correlation between concepts on the concept level. The inter-concept relationship is derived from the class labels. However, the proposed framework further explores such an inter-concept relationship in the attribute level (details to be explained in Chapter 3.4.1).

Definitions

Before elaborating the framework, several basic concepts and terms to be used in the framework are first introduced.

Definition 5 (Target Concept) *A **target concept** refers to a concept whose retrieval performance is concerned by the current task.*

Definition 6 (Reference Concept) *A **reference concept** refers to a concept whose occurrence is accompanied with the target concept with a high chance. A concept is considered as a reference concept when (i) the concept has a high percentage of chance to occur if the target concept appears, and (ii) the concept is easy to retrieve.*

Definition 7 (Feature-value Pair) *Suppose that one discretization method is applied to an attribute A_i and creates a few partitions A_i^1, A_i^2, ..., A_i^P, where i is an identifier of the attribute and P is the number of partitions created for attribute A_i. Each partition is called a **feature-value pair**. Particularly, A_i^j stands for the j-th partition of attribute A_i.*

Definition 8 (DIAG Function) *Let $A = \{a_1, a_2, ..., a_N\}^T$ be an N × 1 vector. A DIAG function is defined in Equation (3.35) which transforms A into an N × N matrix D.*

$$D = DIAG(A), \tag{3.35}$$

where each element d_{ij} in D satisfies

$$d_{ij} = \begin{cases} a_i & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

Deriving attribute-based co-occurrence relationships

There are a number of ways to capture the inter-concept relationship. In this dissertation, the focus is put on the correlations between the attributes and the co-occurrence between the target concept and the reference concept. It will be more accurate to utilize the inter-concept relationship on the attribute or sub-attribute level than on the concept labels. Compared with those methods that capture the correlation between concepts based on the class labels [94, 47], the proposed correlation in this dissertation is more closely related with the observation details of the data instances.

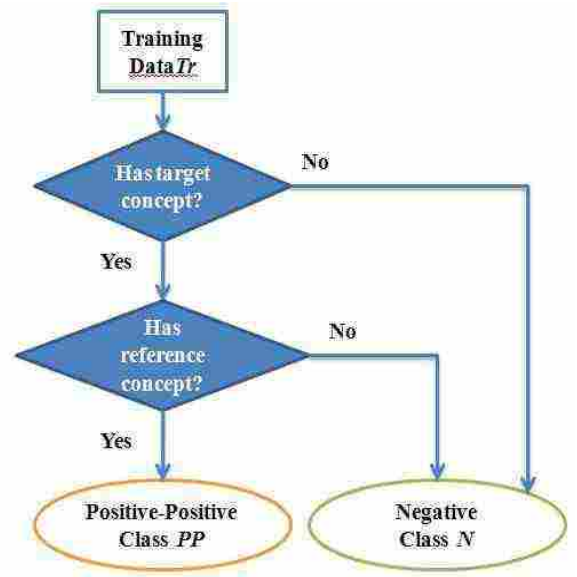


Figure 3.16: Construction of co-occurrence classes

To capture the aforementioned correlation, a co-occurrence class is built from the training set as shown in Figure 3.16. The training data instances which belong to both the target concept and the reference concept are in the positive-positive class (*PP*). The rest of the training data instances are in the negative class (*N*). Since the attributes used in semantic concept retrieval are mostly numeric, but some techniques that describe the relationship between two attributes, like correspondence analysis, only take nominal data as the input, a discretization method is applied before deriving the correlation and co-occurrence relationships. In this dissertation, the minimum description length (MDL)-based supervised discretization method is employed to discretize the training and testing data into several feature-value pairs corresponding to the co-occurrence classes. After discretization, multiple correspondence analysis (MCA), which is able to capture the correlation between more than two variables [113, 114], is employed to build a correlation table for each feature-value pair and its impact weight.

The construction of the correlation table using MCA is discussed as follows. Suppose there is a discretized input matrix $\Pi \in R^{\eta \times \eta}$ as shown in Table 3.35. Each row

Table 3.35: A discretized input matrix

Attribute 1	...	Attribute T	PP	N
A_1^1	...	A_T^1	1	0
A_1^2	...	A_T^1	0	1
...	...	...	...	...
A_1^1	...	A_T^2	0	0

Table 3.36: Indicator matrix of the input matrix

A_1^1	A_1^2	...	A_T^1	A_T^2	...	PP	N
1	0	...	1	0	...	1	0
0	1	...	1	0	...	0	1
...	...	...	...	...	...	...	...
1	0	...	0	1	...	0	0

stands for a data instance. A_1^1 and A_T^1 are the feature-value pairs. For concept classes such as PP and N, 1 means a data instance belongs to that class, while 0 means it does not. For example, in Table 3.35, the first data instance has 1 on PP, meaning it contains both the target concept and the reference concept.

The input matrix is accompanied with an indicator matrix and a Burt matrix. The indicator matrix is an equivalent form to represent the discretized input matrix, where each element is either 1 or 0. For example, the indicator matrix I of Table 3.35 is shown in Table 3.36. The Burt matrix B is generated using Equation (3.36). Let g be the grand total of B as shown in Equation (3.37). The probability matrix Γ is defined as $\Gamma = B/g$. It is easy to observe that Γ is as symmetric as B and the element of Γ is between 0 and 1. CODE 3.4 shows the procedure to derive the impact weight of each feature-value pair towards class PP from the probability matrix Γ.

CODE 3.4: DERIVING IMPACT WEIGHTS FROM Γ

```
Input:
    a probability matrix $\Gamma \in R^{\eta \times \eta}$
Output:
    impact weights $W(A_i^j, PP)$
----------------------------------------
Calculate $V \in R^{\eta \times 1}$, which are the column totals of $\Gamma$.
Derive diagonal matrix $D$ by applying Equation (3.35)
    (see Definition 8) to $V$.
Generate a centralized matrix Z using Equation (3.38).
Apply eigendecomposition (see Equation (3.39)) on Z to
    derive its eigenvectors $Q$=$\{q_1, q_2,...,q_\eta\}$ in a descending
    order of the corresponding eigenvalues.
Project Z as (X,Y) on the subspace spanned by $q_1$
    and $q_2$, as shown in Equation (3.40).
Derive impact weights $W(A_i^j, PP)$ for each feature-value
    pair $A_i^j$ on class $PP$ using Equation (3.41).
Output impact weights $W(A_i^j, PP)$ for $A_i^j$ on $PP$.
----------------------------------------
```

$$B = I^T I \tag{3.36}$$

$$g = \sum_{i=1}^{K} \sum_{j=1}^{K} b_{ij} \tag{3.37}$$

$$Z = D^{-\frac{1}{2}} (\Gamma - VV^T)(D^T)^{-\frac{1}{2}} \tag{3.38}$$

$$Z = Q\Lambda Q^{-1} \tag{3.39}$$

$$(X,Y) = Z * (q_1, q_2) \tag{3.40}$$

$$W(A_i^j, PP) = \frac{(X_{PP}, Y_{PP})^T \cdot (X_{A_i^j}, Y_{A_i^j})}{|(X_{PP}, Y_{PP})| \cdot |(X_{A_i^j}, Y_{A_i^j})|}. \tag{3.41}$$

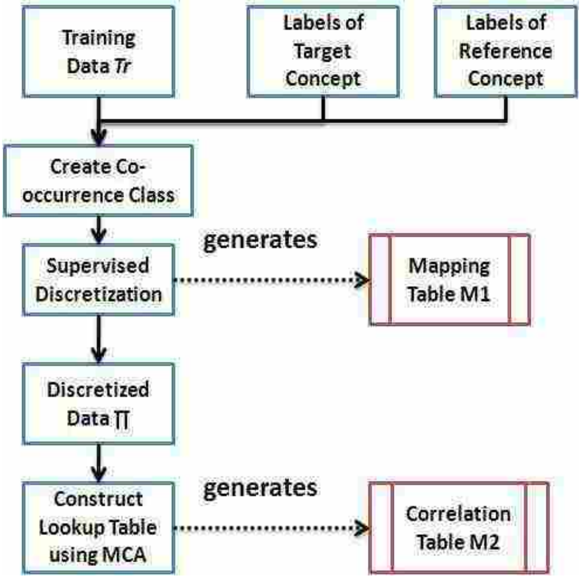


Figure 3.17: Detailed procedure of learning phase

The proposed ranking framework

CODE 3.5: LEARNING PHASE

```
Input:
    Training data instance $Tr$, training labels of the
    target concept and reference concept.
Output:
    Mapping table M1 and correlation table M2
Create co-occurrence class following the procedure of
    Figure 3.16.
Apply supervised discretization method to discretize
    the numeric training data. Create a mapping table M1
    to store the generated feature-value pairs and their
    corresponding ranges of partitions.
Apply MCA on the discretized data $\prod$ to generate
    a correlation table M2 between feature-value pairs and
    impact weights by following the procedure of CODE 3.4.
Output tables M1 and M2
```

The proposed ranking framework consists of two phases: a learning phase and a ranking phase. It is assumed that the target concept is known and the reference concept is appropriately selected based on the defined criteria or by empirical study. In the learning step, the correlation between feature-value pairs and *PP* is learned and a relationship is established via constructing a correlation table for each feature-value pair and class *PP* from the training data instances (Tr). The detailed procedure of the learn-

Table 3.37: An example of mapping table M1

Feature-value pair	Partition range
A_1^1	(-∞, -0.1]
A_1^2	(-0.1, 2.2]
...	...
A_i^j	(-5.1, -3]
...	...

Table 3.38: A correlation table M2

Feature-value pair	impact weight
A_1^1	$W(A_1^1, PP)$
A_1^2	$W(A_1^2, PP)$
...	...
A_i^j	$W(A_i^j, PP)$
...	...

ing phase is shown in Figure 3.17 and CODE 3.5. The generated mapping table M1 (Table 3.37) records the feature-value pair with their ranges of values for an attribute. The generated correlation table M2 (Table 3.38) is made of feature-value pairs and their corresponding impact weights to class PP.

The ranking phase utilizes these two tables generated from the learning phase to rank the ranking scores by considering the correlation between the target concept and reference concept. Each testing data instance (Ts) is described by a vector of attributes. Therefore, after looking up mapping table M1 and correlation table M2, a vector of feature-value pairs and impact weights are generated. The generated impact weights are then summed together to form the ranking weight for each data instance. The details of ranking phase can be found in Figure 3.18 and CODE 3.6.

CODE 3.6: Ranking phase

```
**Input**:
    Testing data instance $Ts$, mapping table M1 and
    correlation table M2 from the learning phase.
**Output**:
    Final ranking score ReS
─────────────────────────────
Get $RS(Ts, \varphi_t)$, the ranking score of $Ts$ from ranking
    model $\varphi_t$ for the target concept.
Look up mapping table M1 and find the feature-value
    pair vector.
Look up correlation table M2 and find the
    corresponding impact weight vector.
Calculate the summation total $W$ of the impact weight
    vector and use $W$ as the ranking weight.
Calculate ReS, the final ranking score of $Ts$ with regards to
    the target concept by using ReS=$RS(Ts, \varphi_t) \cdot (1+W)$.
Output ReS
─────────────────────────────
```

Experiment Setup

To show the effectiveness of the framework, experiments are conducted on the MediaMill Challenge Data Set. The MediaMill Challenge Data Set contains 101 semantic concepts and five different experiments. In the experiment, the features in the first experiment are used. There are $30,993$ data instances in the training data set and $12,914$ data instances in the testing data set. For different semantic concepts, the testing data set provides the ranking scores and labels of all testing data instances for evaluation.

The experiment is carried out by first building a co-occurrence probability matrix CP from the training data set, in which each element CP_{ij} is given by Equation (3.42).

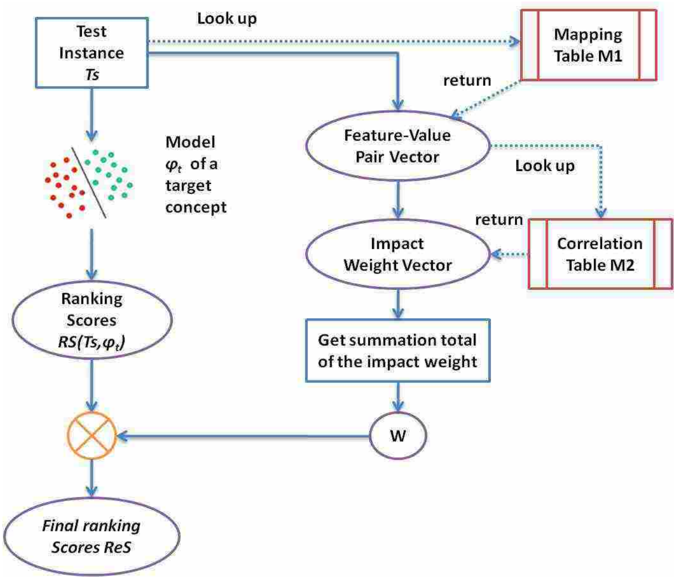

Test Instance *Ts*
Look up
Mapping Table M1
return
Feature-Value Pair Vector
Model φ_t of a target concept
Look up
return
Impact Weight Vector
Correlation Table M2
Ranking Scores *RS(Ts,φ_t)*
Get summation total of the impact weight
W
Final ranking Scores ReS

Figure 3.18: Detailed procedure of ranking phase

$$CP_{ij} = \frac{\#\text{ of instances belonging to both concepts } i \text{ and } j}{\#\text{ of instances belonging to concept } i} \tag{3.42}$$

For a target concept i, a reference concept k is selected if

- CP_{ik}=max$\{CP_{ij}\}$;
- $CP_{ik} > 0.9$; and
- $k \neq i$.

Table 3.39 shows the target concept, reference concept and the co-occurrence probability (CP) with regard to the target concepts that are used in the experiment. It is worth noting that the selection of the reference concepts does not require domain knowledge. The selection is rather objective and depends only on the co-occurrence probability between the concepts. The performance is evaluated using average precision (AP), which is commonly adopted to evaluate the effectiveness of the retrieval. The AP is defined in Equation (3.43).

Table 3.39: Concepts and their co-occurrence probability

Target Concept	Reference Concept	CP
entertainment	people	0.93
urban	outdoor	1
sky	outdoor	1
road	outdoor	1
map	graphics	1
snow	outdoor	1

$$AP = \frac{1}{||I_+||} \sum_{\omega=1}^{\widehat{N}} r_\omega \cdot \frac{1}{\omega} \sum_{o=1}^{\omega} r_o \tag{3.43}$$

- $||I_+||$ is the number of relevant images with respect to the query concept;
- $\widehat{N}$ is the number of retrieved images;
- r_o is defined in Equation (3.44).

$$r_o = \begin{cases} 1, & \text{if the image o is relevant} \\ 0, & \text{otherwise} \end{cases} \tag{3.44}$$

The AP before and after applying the ranking framework is compared in different scales to show the gain of the performance.

Results and Analyses

The AP of the baseline and the final ranking results by applying the proposed ranking framework are demonstrated in Figure 3.19 to Figure 3.24. The AP are evaluated at the first 10, 20, 30, 40, 50, 60, 80, 100, 150, 200, and all data instances. From these figures, the improvement in AP made by the proposed ranking framework is clear, especially for the first 10, 20 and 30 retrieved results. In Figure 3.19, the AP of the baseline is almost 0; while it can reach above 60% after ranking. The AP on the first 200 is still almost 20% better after the proposed ranking framework is applied. In Figure 3.22, the target concept “road” provides an AP of 60% for the first 10 retrieved results. It is even more exciting to see that the AP on the first 10 is improved to be perfect after weighted ranking using the correlation between “road” and “outdoor”. The same result is achieved on the target concept “map”. After ranking using its correlation with “graphics”, the AP of the first 10 retrieved data instances reaches 100%. Even for easy-to-retrieve concepts like “sky”, the correlation with the reference concepts “outdoor” can further be used to improve their retrieval performance.

In addition to the aforementioned promising results, it can also be observed that the AP after the weighted ranking tends to get close to the baseline with an increased number of retrieved data instances. There are several aspects to interpret the phenomena. First, the number of misclassified data instances increases when more data instances are retrieved. The increase of misclassification will definitely compromise the retrieval performance in terms of AP. Second, for some positive data instances with lower ranking scores that are very easy to misclassify, the correlation in terms of the impact weights does not help boost their rankings. This is reasonable since the attribute values of these positive data instances may be either on the margin or deeply in the distribution area of negative data instances.

In sum, a semantic concept retrieval framework is proposed with a new ranking algorithm that considers not only the high co-occurrence relation between concepts (i.e., both concepts appear frequently), but also the low/none co-occurrence relation between concepts (i.e., the appearance of one concept indicates a very small or zero chance of the appearance of the other concept). This study is motivated by the ideas that (1) the correlation among the concepts is useful and can be used to enhance the retrieval results; and (2) the proposed ranking algorithm considers the inter-concept relationships in the attribute level.

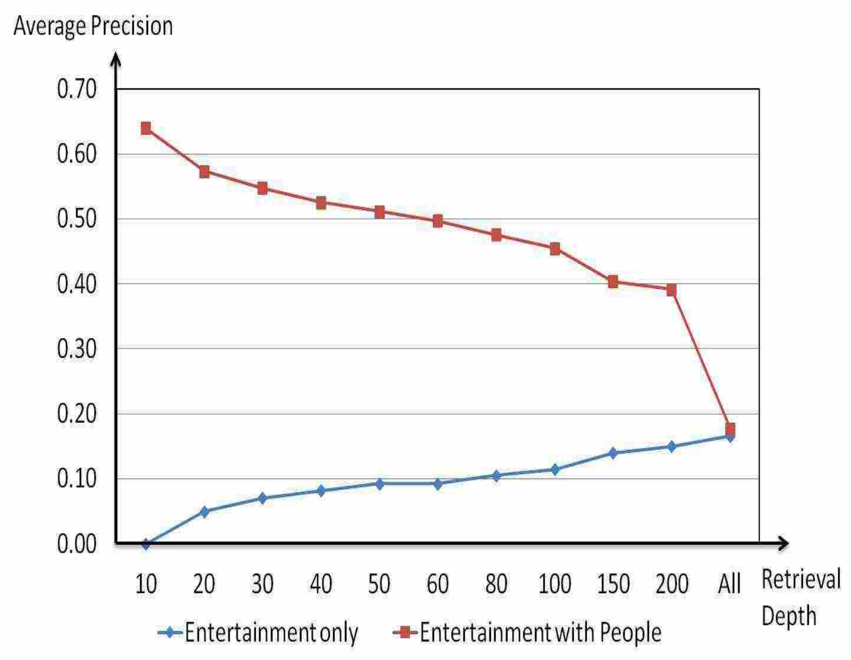


Figure 3.19: Average precision of concept “entertainment” with or without reference concept “people”

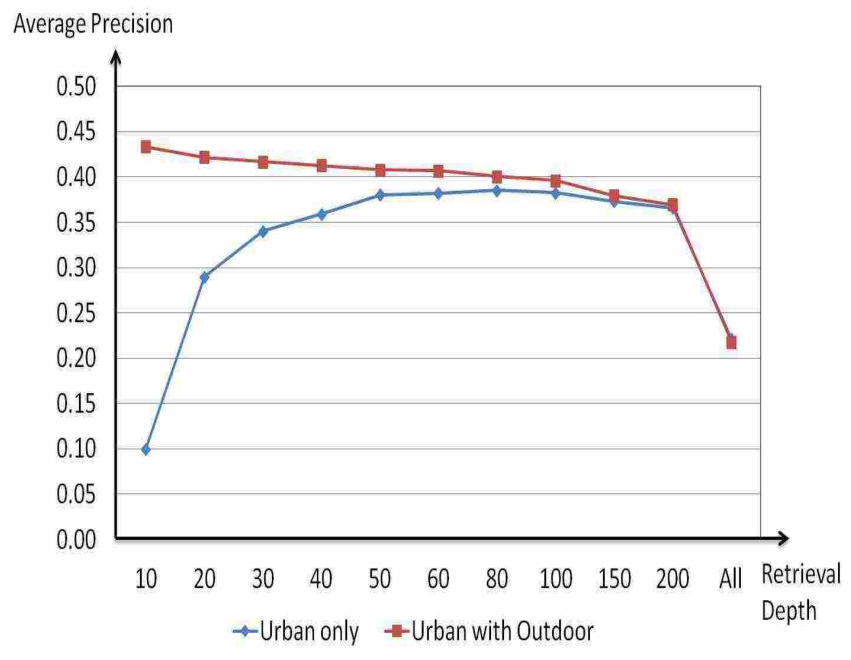


Figure 3.20: Average precision of concept “urban” with or without reference concept “outdoor”

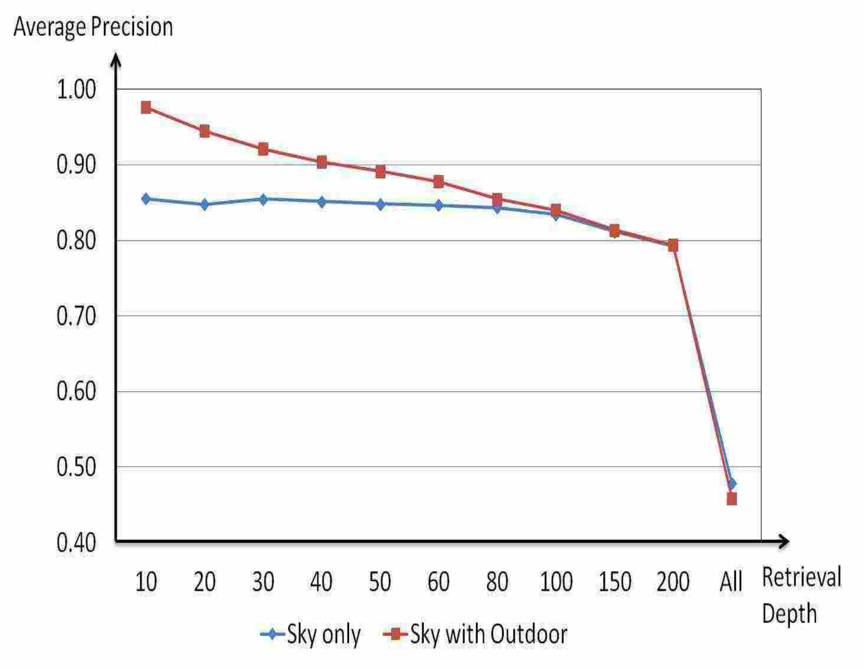


Figure 3.21: Average precision of concept “sky” with or without reference concept “outdoor”

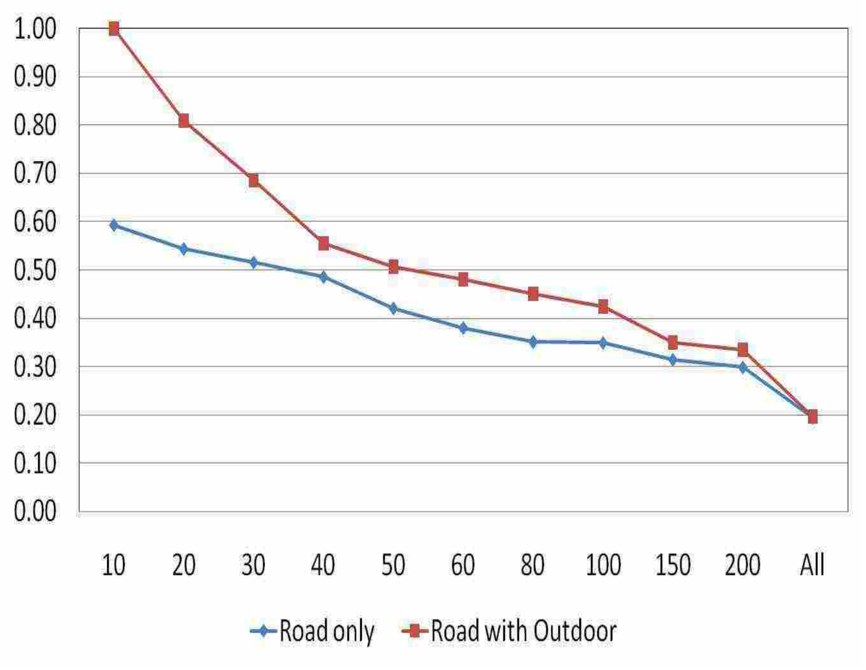


Figure 3.22: Average precision of concept “road” with or without reference concept “outdoor”

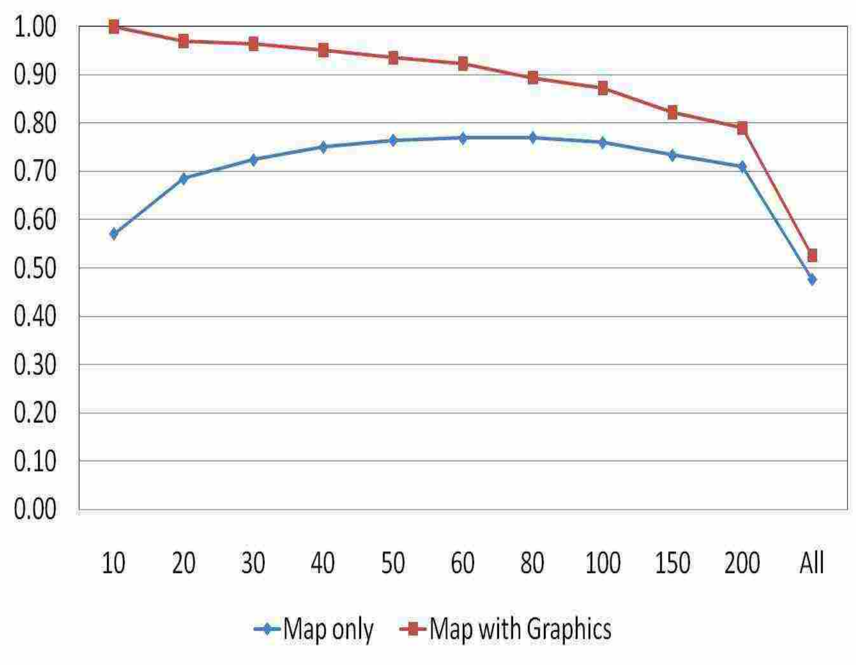


Figure 3.23: Average precision of concept “map” with or without reference concept “graphics”

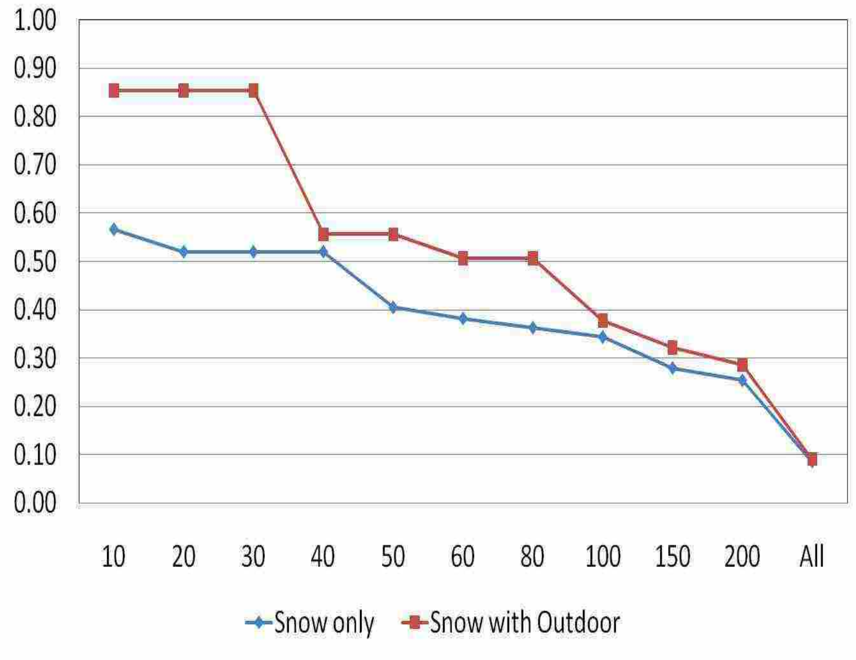


Figure 3.24: Average precision of concept “snow” with or without reference concept “outdoor”

3.4.2 Semantic Concept Retrieval Using Inclusive and/or Exclusive Relationships between Multiple Semantic Concepts

Two kinds of inter-concept relationships, namely the "inclusive" and "exclusive" relationships, can be considered, as defined in the concept conditional probability (CCP) (shown in Equation (3.45)). In an "inclusive" relationship, a reference concept C_j is co-occurred with a target concept C_i with a large chance (e.g., CCP is close to 1). On the other hand, in an "exclusive" relationship, a reference concept C_j hardly appears together with a target concept C_i, where the CCP value is small or is equal to 0. Such an inclusive (or exclusive) relationship reveals that the positive instances of the target concept C_i (referring to the instances containing the target concept C_i) are supposed to show a high probability to contain (or not to contain) the references concepts.

$$CCP(C_j|C_i) = \frac{\text{\# of instances belonging to both concepts } C_i \text{ and } C_j}{\text{\# of instances belonging to concept } C_i} \tag{3.45}$$

The proposed framework in chapter 3.4.1 uses only one reference concept and its inclusive or exclusive inter-concept relationship to refine the retrieval performance of the target concept. In this chapter, a new framework that considers more than one reference concept and both the "inclusive" and "exclusive" inter-concept relationships is proposed.

Generation of Co-occurrence Classes

A co-occurrence class refers to a class of the training instances which contain the target concept and a subset of the concepts in the reference concept set. For a target concept C_0, the reference concept set RC is considering the following two relationships.

- Inclusive relationship: If $CCP(C_j|C_0) > \tau$ then RC $\leftarrow$ RC $\bigcup C_j$
- Exclusive relationship: If $CCP(C_j|C_0) < \varepsilon$ then RC $\leftarrow$ RC $\bigcup \overline{C_j}$, where j=1, $\cdots$, # of concepts

For example, there is a reference concept set $RC = \{C_1, \overline{C_2}, \ldots, C_K\}$ with respect to a target concept C_0. In this example, C_1 and C_K hold an inclusive relationship with C_0 in that $CCP(C_1|C_0)$ and $CCP(C_K|C_0)$ are large. In the experiments, the inclusive relationship is satisfied if the CCP value is larger than a threshold value (0.9 in this study). $\overline{C_2}$ is the complementary concept of C_2. An instance with $\overline{C_2}$ denotes that it does not contain the concept C_2. $\overline{C_2}$ is included in the reference concept as long as $CCP(C_2|C_0)$ is smaller than a threshold value (0.1 in this study). Please note that both 0.9 and 0.1 threshold values are determined empirically and can be adjusted if needed.

A co-occurrence class is composed of the target concept and any k-itemset ($1 \leq k \leq K$) generated from $RC = \{C_1, \overline{C_2}, \ldots, C_K\}$. In this study, a k-itemset is defined as a subset with k reference concepts from RC. For example, $\{C_1\}$ is a 1-itemset, $\{C_1, \overline{C_2}\}$ is a 2-itemset, and etc. Theoretically, a target concept and a reference concept set with K concepts could generate as many as 2^K itemsets. However, the actual number of itemsets could be less because some concepts could be the subset of the others. For example, the co-occurrence class generated by a target concept “road” with 1-itemset $\{\overline{outdoor}\}$ is the same as the class generated by “road” with $\{indoor\}$. CODE 3.7 shows the way to generate a unique set of co-occurrence classes from a reference concept set RC.

CODE 3.7: GENERATION OF A UNIQUE SET OF CO-OCCURRENCE CLASSES

```
Input:
training labels of a target concept $C_0$ and a reference concept set
$RC$=$\{C_1,\overline{C_2},\cdots,C_K\}$
Output:
a unique set of co-occurrence classes $\Omega$
___________________________________________
$\Omega \leftarrow \emptyset$
FOR k=1 to # of concepts in $RC$
    generate all $k$-itemsets from $RC$.
    FOR each $k$-itemset
        generate the co-occurrence class $\omega$ that contains $C_0$
        as well as the concepts in current $k$-itemset.
        IF $\omega \bigcap \Omega = \emptyset$ THEN
            $\Omega = \Omega \bigcup \omega$
        END
    END
END
___________________________________________
```

Deriving Attribute-based Co-occurrence Relationships

For convenience, assume that Ω_m is one of the co-occurrence classes generated in Chapter 3.4.1. The training instances which belong to Ω_m are labeled with the positive class (P_m), while the rest of the training data instances are labeled with the negative class (N_m). In this dissertation, multiple correspondence analysis (MCA) is adopted to analyze such correlation between the attributes and the co-occurrence class. However, the

Table 3.40: A discretized matrix for the co-occurrence class Ω_m

Attribute 1	...	Attribute t	...	P_m	N_m
A_1^2	...	A_t^1	...	1	0
A_1^1	...	A_t^1	...	0	0
...	...	...	...	...	...
A_1^2	...	A_t^2	...	0	1

Table 3.41: An indicator matrix I_m for the co-occurrence class Ω_m

A_1^1	A_1^2	...	A_t^1	A_t^2	...	P_m	N_m
0	1	...	1	0	...	1	0
1	0	...	1	0	...	0	0
...	...	...	...	...	...	...	...
0	1	...	0	1	...	0	1

attributes (like color histogram, wavelet texture, and so on) of the training instances used in semantic information retrieval are mostly numeric. To apply MCA to describe the relationship between the attributes and the co-occurrence class, a discretization step is necessary to convert the values of the attributes from "numeric" to "nominal" before MCA is applied. Here, the minimum description length (MDL)-based supervised discretization method is used to discretize the attributes of the training and the testing instances into a number of feature-value pairs (referring to the partitions of all possible values of an attribute) corresponding to the co-occurrence class.

Suppose that the MDL-based discretization method is applied to an attribute A_t, which creates a few partitions A_t^1, A_t^2, $\ldots$, A_t^{β}, where t is the identifier of the attribute (i.e., the t^{th} attribute) and β is the number of partitions generated for attribute A_t. In the study, each partition is called a **feature-value pair**. Particularly, A_t^{α} stands for the α-th partition of attribute A_t. For convenience, a mapping table (M1) is generated to link each feature-value pair with a partition interval within an attribute so that it is easy to convert a testing instance with numeric attribute values into the nominal values (as the one used in Chapter 3.4.1).

Table 3.40 shows an example of the discretized training instances in the form of a data matrix, where each row denotes a training instance and each column stands for an attribute or a class. A training instance with P_m=1 (or N_m=0) means that it belongs to the co-occurrence class Ω_m, while P_m=0 (or N_m=1) indicates that the training instance is irrelevant to the co-occurrence class. MCA converts the discretized matrix into an indicator matrix for further analyses. Note that there is no information loss during such a conversion. An example of the indicator of the co-occurrence class Ω_m is shown in Table 3.41. Following the conventional process of MCA, CODE 3.8 shows the procedure to derive the impact weight of each feature-value pair towards the co-occurrence class Ω_m from the indicator matrix I_m.

CODE 3.8: DERIVING IMPACT WEIGHTS FROM AN INDICATOR MATRIX I_m

```
1  **Input**:
   an indicator matrix $I_m \in R^{\eta \times \nu}$
2  **Output**:
   impact weights $W(A_i^j, \Omega_m)$
   ________________________________________
3  Generate Burt matrix $B_m$ from $I_m$ using Equation (3.46).
4  Let $g_m$ be the grand total of $B_m$ as shown in Equation (3.47).
   Generate probability matrix $\Gamma_m = B_m / g_m$.
5  Calculate $V_m \in R^{\nu \times 1}$, which are the column totals of $\Gamma_m$.
6  Derive diagonal matrix $D_m$ by applying Equation (3.48) to $V_m$.
7  Generate a centralized matrix $Z_m$ using Equation (3.49).
8  Apply eigen value decomposition (see Equation (3.50)) on $Z_m$ to
   derive its eigenvectors $Q_m$=\{$q_1^{(m)}$, $q_2^{(m)}$,...,$q_\eta^{(m)}$\} in a descending
   order of the corresponding eigenvalues.
9  Project $Z_m$ as ($X_m$, $Y_m$) on the subspace spanned by $q_1^{(m)}$
   and $q_2^{(m)}$, as shown in Equation (3.51).
10 Derive impact weights $W(A_i^j, \Omega_m)$ for each feature-value
   pair $A_i^j$ on class $\Omega_m$ using Equation (3.52).
11 Output impact weights $W(A_i^j, \Omega_m)$ for $A_i^j$ on $\Omega_m$.
   ________________________________________
```

As shown in CODE 3.8, given the indicator matrix I_m (Line 1), let I_m^T be the transpose of the matrix I_m, and then the Burt matrix B_m can be calculated from I_m^T and I_m using Equation (3.46) in Line 3. Let b_{ij} be the element at the i-th row and the j-th column of the matrix B_m, and g_m be the grand total of B_m as shown in Equation (3.47), Lines 4 and 5 generate the probability matrix Γ_m and V_m which are the column totals of Γ_m. In Line 6, the diagonal matrix D_m for V_m can be generated by applying Equation (3.48) to V_m.

$$B_m = I_m^T I_m. \tag{3.46}$$

$$g_m = \sum_{i=1}^{v}\sum_{j=1}^{v} b_{ij}. \tag{3.47}$$

$$D = DIAG(\Theta), \tag{3.48}$$

where $\Theta = \{\theta_1, \theta_2, ..., \theta_{\widetilde{N}}\}^T$ is an $\widetilde{N} \times 1$ vector and each element d_{ij} in D satisfies:

$$d_{ij} = \begin{cases} \theta_i & \text{if } i = j; \\ 0 & \text{otherwise.} \end{cases}$$

In Line 7, a centralized matrix Z_m is constructed using Equation (3.49). Then, Z_m is utilized to construct the eigenvalues and eigenvectors, and select the subspace spanned by the first two principal components (Lines 8 and 9). Assume that X_{Ω_m} and Y_{Ω_m} are the corresponding projection of the co-occurrence class Ω_m on $q_1^{(m)}$ and $q_2^{(m)}$, and similarly, $X_{A_i^j}$ and $Y_{A_i^j}$ are the corresponding projection of a feature-value pair A_i^j on $q_1^{(m)}$ and $q_2^{(m)}$. Lines 10 and 11 derive the impact weights using Equation (3.52) as the output.

$$Z_m = D_m^{-\frac{1}{2}}(\Gamma_m - V_m V_m^T)(D_m^T)^{-\frac{1}{2}}. \tag{3.49}$$

$$Z_m = Q_m \Lambda Q_m^{-1}. \tag{3.50}$$

$$(X_m, Y_m) = Z_m * (q_1^{(m)}, q_2^{(m)}). \tag{3.51}$$

$$W(A_i^j, \Omega_m) = \frac{(X_{\Omega_m}, Y_{\Omega_m})^T \cdot (X_{A_i^j}, Y_{A_i^j})}{|(X_{\Omega_m}, Y_{\Omega_m})| \cdot |(X_{A_i^j}, Y_{A_i^j})|}. \tag{3.52}$$

The Proposed Ranking Algorithm

The proposed ranking algorithm consists of two phases: a learning phase and a ranking phase. In the learning phase, one of the co-occurrence classes is selected with the highest retrieval performance in term of average precision (as defined in Equation (3.43) in Chapter 3.4.1). A mapping table ($M1$) is constructed to convert the numeric attributes into nominal attributes (see Table 3.42 for example) and afterwards a correlation table ($M2$) will be generated to store the impact weights between each feature-value pair and the selected co-occurrence class (see Table 3.43 for example). The detailed procedure of the learning phase is shown in CODE 3.9.

Table 3.42: An example mapping table $M1$

Feature-value pair	Partition range
A_1^1	(-∞, -0.2]
A_1^2	(-0.2, 2.6]
...	...
A_i^1	(-∞, -9.4]
...	...
A_i^j	(-5.1, -3]
...	...

Table 3.43: An example correlation table M2 with respect to the selected co-occurrence class Ω_m

Feature-value pair	impact weight
A_1^1	$W(A_1^1, \Omega_m)$
A_1^2	$W(A_1^2, \Omega_m)$
...	...
A_i^1	$W(A_i^1, \Omega_m)$
...	...
A_i^j	$W(A_i^j, \Omega_m)$
...	...

CODE 3.9: LEARNING PHASE

```
1   Input:
    Training data instances in $Tr$ with labels, and a set of co-occurrence classes $\Omega$
2   Output:
    The selected co-occurrence class $\Omega_o$, mapping table $M1$,
    and correlation table $M2$
    ________________________________
3   SET $AP$=0, $\Omega_o$=$\emptyset$, $M1$=$\emptyset$, $M2$=$\emptyset$.
4   FOR each co-occurrence class $\Omega_i$ in $\Omega$
5       Apply a supervised discretization method to discretize
        the numeric training data into discretized data $\prod^{(i)}$.
        Create a mapping table $M1^{(i)}$ to store the generated
        feature-value pairs and their corresponding ranges of partitions.
6       Apply MCA on $\prod^{(i)}$ to generate a correlation table $M2^{(i)}$
        to store the impact weights between feature-value pairs and
        $\Omega^{(i)}$ by following the procedure of CODE 3.8.
7       FOR each instance $Tr[u]$ in $Tr$
8           Get all feature-value pairs and their impact weights to $\Omega_i$.
9           The ranking weight $Score^{(i)}(Tr[u]) \leftarrow$ the average value
            of all related impact weights.
10      END
11      Calculate the average precision $AP^{(i)}$ on $Score^{(i)}(Tr)$, which is
        composed of all $Score^{(i)}(Tr[u])$.
12      IF $AP^{(i)} > AP$ THEN
            UPDATE $AP$=$AP^{(i)}$, $\Omega_o$=$\Omega_i$, $M1$=$M1^{(i)}$, $M2$=$M2^{(i)}$.
13      END
14  END
15  Output $\Omega_o$, Table $M1$, and Table $M2$.
    ________________________________
```

In CODE 3.9, AP, Ω_o, $M1$ and $M2$ are initialized (Line 3). The learning phase iteratively selects one co-occurrence class Ω_i from Ω (Line 4) and discretizes the training data instances in Tr to $\prod^{(i)}$ by a MDL-based supervised discretization method. The mapping between discretization intervals and the feature-value pairs are stored in $M1^{(i)}$ (Line 5). Next, MCA is employed to derive the impact weights between each feature-value pair and $\Omega^{(i)}$, which are stored in Table $M2^{(i)}$ (Line 6). For each training data instance $Tr[u]$, the corresponding feature-value pairs and their impact weights to Ω_i can be retrieved by looking up Table $M1^{(i)}$ and $M2^{(i)}$ (Line 8), and the average value of the impact weights serves as the ranking weight for $Tr[u]$ (Line 9). In Line 11, an average precision value $AP^{(i)}$ with respect to the co-occurrence class $\Omega^{(i)}$ is derived from the ranking weights of Tr and is compared with the predefined AP value. If $AP^{(i)}$ is larger than AP, which means $\Omega^{(i)}$ is able to render a better performance than the previous co-occurrence classes, AP, Ω_o, $M1$, and $M2$ are therefore needed to be updated (in Lines 12 and 13). After the AP values of all co-occurrence classes have been evaluated, the optimal co-occurrence class with the best AP will be the output from the training phase (in Line 15).

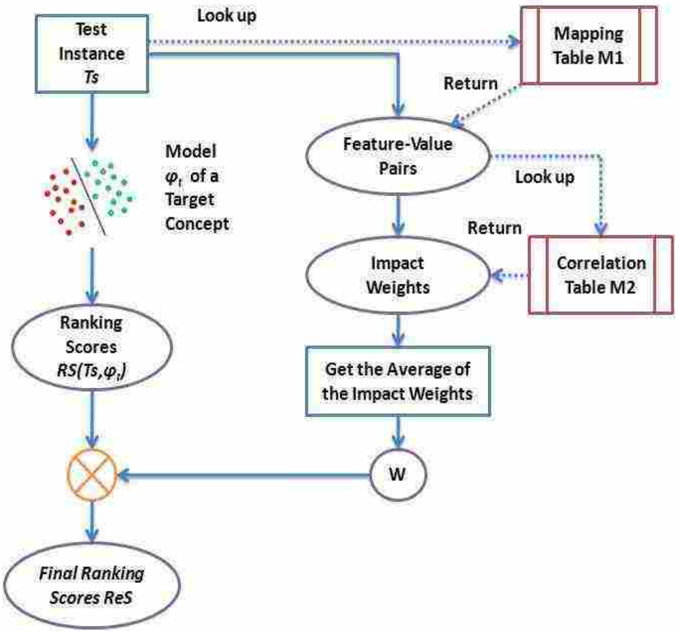


Figure 3.25: Detailed procedure of the ranking phase

In the ranking phase, the selected co-occurrence class Ω_o, together with the $M1$ and $M2$ tables created from the learning phase are used to generate the final ranking scores from the score of the stand-alone model by considering the correlation between the corresponding feature-value pairs and Ω_o. Each testing data instance (Ts) is described by a vector of attributes, followed by looking up mapping table $M1$ and correlation table $M2$ to get the impact weights for each attribute. Then, similar to the generation of $Score^{(i)}(Tr)$ in CODE 3.9, the impact weights are averaged to form the ranking weights to obtain the final ranking scores (see Figure 3.25 and CODE 3.10 for details).

CODE 3.10: RANKING PHASE

```
1  **Input**:
   Testing data instance $Ts$, the selected co-occurrence class $\Omega_o$,
   mapping table $M1$, and correlation table $M2$.
2  **Output**:
   ranking score ReS
   ________________________________________
3  Get $RS(Ts, \varphi_t)$, the ranking score of $Ts$ from stand-alone
   ranking model $\varphi_t$ for a target concept.
4  Look up mapping table M1 and find the feature-value
   pairs for each testing data instance $Ts$.
5  Look up correlation table $M2$ and find the corresponding
   impact weights.
6  Calculate the average value $W$ of the impact weights and use
   $W$ as the ranking weight for $Ts$.
7  Calculate ReS, the ranking score of $Ts$ with respect to the
   target concept by using ReS=$RS(Ts, \varphi_t)\cdot(1+W)$.
8  Output the ranking Score ReS.
   ________________________________________
```

Given a ranking score $RS(Ts, \varphi_t)$ for the testing data instance Ts from a stand-alone ranking model (in Line 3), CODE 3.10 aims to generate a ranking score ReS. After looking up the related feature-value pairs (in Line 4) and impact weights (in Line 5) of Ts from Tables $M1$ and $M2$, the ranking weight W is chosen to be the average value of the impact weights (in Line 6). In Line 7, the final ranking score is derived by using ReS=$RS(Ts,\varphi_t)\cdot(1+W)$, which is the output of this ranking phase (Line 8).

Experimental Results and Analyses

To evaluate the effectiveness of the proposed ranking algorithm in the proposed semantic concept retrieval framework, several experiments are conducted on two different datasets. The experimental setup is introduced in ***Experimental Setup*** and the results are demonstrated and discussed in Chapter **Results and Analyses**.

Experimental Setup

Two datasets, namely the MediaMill Challenge Data Set [4] (TRECVID2005 news and broadcast videos) and TRECVID2011 video collections [42] are used in the experiments.

The MediaMill Challenge Data Set contains 101 semantic concepts and the dataset has been split in advance by the provider into a training set ($30,993$ data instances) and a testing set ($12,914$ data instances). There are 5 different experiments within the dataset, and the features, models, and ranking scores provided in the experiment 1 are used. The performance is selected based on the provided ranking scores as the baseline.

In the TRECVID2011 video collections, 361 dimensional low-level visual features (including color dominant, color histogram, edge histogram, wavelet texture, and etc.) are extracted from each keyframe The whole dataset is split into a training data set ($118,581$ data instances from IACC.1.tv10.training video sets) and a testing data set ($144,774$ data instances from IACC.1.A video sets). There are a total of 346 concepts but only the first 130 concepts used by TRECVID2010 are used since the rest of the concepts are not guaranteed to have a positive class in the training set.

Table 3.44: Target concepts and their corresponding datasets

Group ID	Target Concept	Source
1	people	MediaMill Challenge Data Set
2	sky	MediaMill Challenge Data Set
3	violence	MediaMill Challenge Data Set
4	anchor	MediaMill Challenge Data Set
5	people_marching	MediaMill Challenge Data Set
6	maps	MediaMill Challenge Data Set
7	splitscreen	MediaMill Challenge Data Set
8	football	MediaMill Challenge Data Set
9	nightfire	MediaMill Challenge Data Set
10	racing	MediaMill Challenge Data Set
11	car	TRECVID2011 Video Collections
12	daytime_outdoor	TRECVID2011 Video Collections
13	ground_vehicle	TRECVID2011 Video Collections
14	indoor	TRECVID2011 Video Collections
15	landscape	TRECVID2011 Video Collections
16	male_person	TRECVID2011 Video Collections
17	person	TRECVID2011 Video Collections
18	reporter	TRECVID2011 Video Collections
19	road	TRECVID2011 Video Collections
20	single_person	TRECVID2011 Video Collections

The subspace models proposed in [115] are used to generate the ranking scores for the testing data instances concept-by-concept and the performance of these ranking scores are used as a baseline in the experiments. The average precision is used to measure the retrieval performance of the baseline and the proposed ranking algorithm. For each target concept, the parameters τ and ε involved in generating the reference concept set are chosen to be 0.9 and 0.1, respectively. The target concepts evaluated in the experiments are shown in Table 3.44.

Results and Analyses

Table 3.45 shows the selected co-occurrence classes of all the 20 target concepts output from the learning phase. The AP values of the ranked results using the proposed re-raking algorithm in comparison with the baseline are displayed from Figure 3.26(a) to Figure 3.30(d). In each figure, the evaluation performance of the AP values for the first 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, 200, and all data instances are plotted for the baseline and the proposed ranking algorithm. The proposed algorithm outperforms the baseline in all figures.

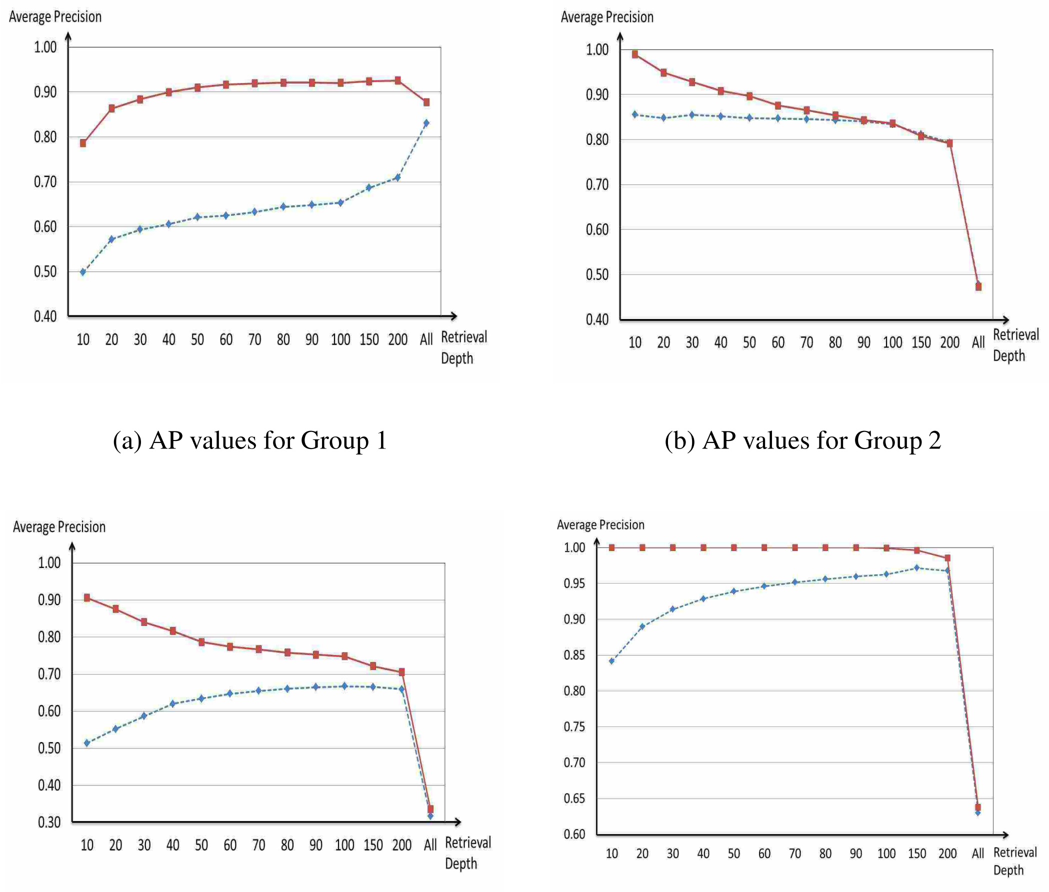
Figure 3.26(a) gives the average precision (AP) values of Group 1 - the concept “people” only (dash line) and “people” without the concept “male” (solid line). In Figure 3.26(b), the AP values of Group 2 - the concept “sky” only (dash line) and “sky” with the concept “outdoor” (solid line) are presented. Figure 3.26(c) shows the AP values of Group 3 - the concept “violence” only (dash line) and “violence” without the concept “government_leader” (solid line). In Figure 3.26(d), the AP values of Group 4 - the concept “anchor” only (dash line) and “anchor” without the concepts “outdoor”, “entertainment”, “crowd”, “walking_running”, and “government_leader” (solid line) are given.

The AP values of Group 5 are shown in Figure 3.27(a) - the concept “people_marching” only (dash line) and ‘people_marching” without the concepts “entertainment”, “crowd”, “government_leader”, and “vehicle” (solid line). In Figure 3.27(b), the AP of Group 6 - the concept “maps” only (dash line) and “maps” without the concepts “outdoor”, “walking”, “running”, “crowd”, and “government_leader” (solid line). Figure 3.27(c) gives the AP values of group 7 - the concept “splitscreen” only (dash line) and ‘splitscreen” without the concepts “outdoor”, “entertainment”, “crowd”, and “government_leader” (solid line) are presented. The AP values of Group 8 - the concept “football” only (dash line) and “football” without the concepts “indoor” and “urban” (solid line) are shown in Figure 3.27(d).

Figure 3.28(a) presents the AP values of Group 9 - the concept “nightfire” only (dash line) and ‘nightfire” without the concepts “entertainment”, “indoor”, “crowd”,

Table 3.45: Target concepts and the selected co-occurrence class in the corresponding datasets from the learning phase

Group ID	Target Concept	the Selected Co-occurrence Class
1	$people$	$\{people,\overline{male}\}$
2	sky	$\{sky, outdoor\}$
3	$violence$	$\{violence, \overline{government\_leader}\}$
4	$anchor$	$\{anchor, \overline{outdoor}, \overline{entertainment}, \overline{crowd},$ $\overline{walking\_running}, \overline{government\_leader}\}$
5	$people\_marching$	$\{people\_marching, \overline{entertainment}, \overline{crowd},$ $\overline{government\_leader}, \overline{vehicle}\ \}$
6	$maps$	$\{maps, \overline{outdoor}, \overline{walking}, \overline{running},$ $\overline{crowd}, \overline{government\_leader}\}$
7	$splitscreen$	$\{splitscreen, \overline{outdoor}, \overline{entertainment}, \overline{crowd},$ $\overline{government\_leader}\}$
8	$football$	$\{football, \overline{indoor}, \overline{urban}\}$
9	$nightfire$	$\{nightfire, \overline{entertainment}, \overline{indoor},$ $\overline{crowd}, \overline{government\_leader}, \overline{road}\}$
10	$racing$	$\{racing, \overline{government\_leader}\}$
11	car	$\{car, \overline{face}\}$
12	$daytime\_outdoor$	$\{daytime\_outdoor, \overline{vehicle}\}$
13	$ground\_vehicle$	$\{ground\_vehicle,\overline{actor}, \overline{face}, \overline{indoor},$ $\overline{male\_person}\}$
14	$indoor$	$\{indoor, \overline{outdoor}\}$
15	$landscape$	$\{landscape, \overline{actor}, \overline{face}, \overline{indoor}\}$
16	$male\_person$	$\{male\_person, \overline{actor}\}$
17	$person$	$\{person, \overline{actor}\}$
18	$reporter$	$\{reporter, \overline{actor}\}$
19	$road$	$\{road, \overline{adult}, \overline{indoor}\}$
20	$single\_person$	$\{single\_person, \overline{outdoor}\}$



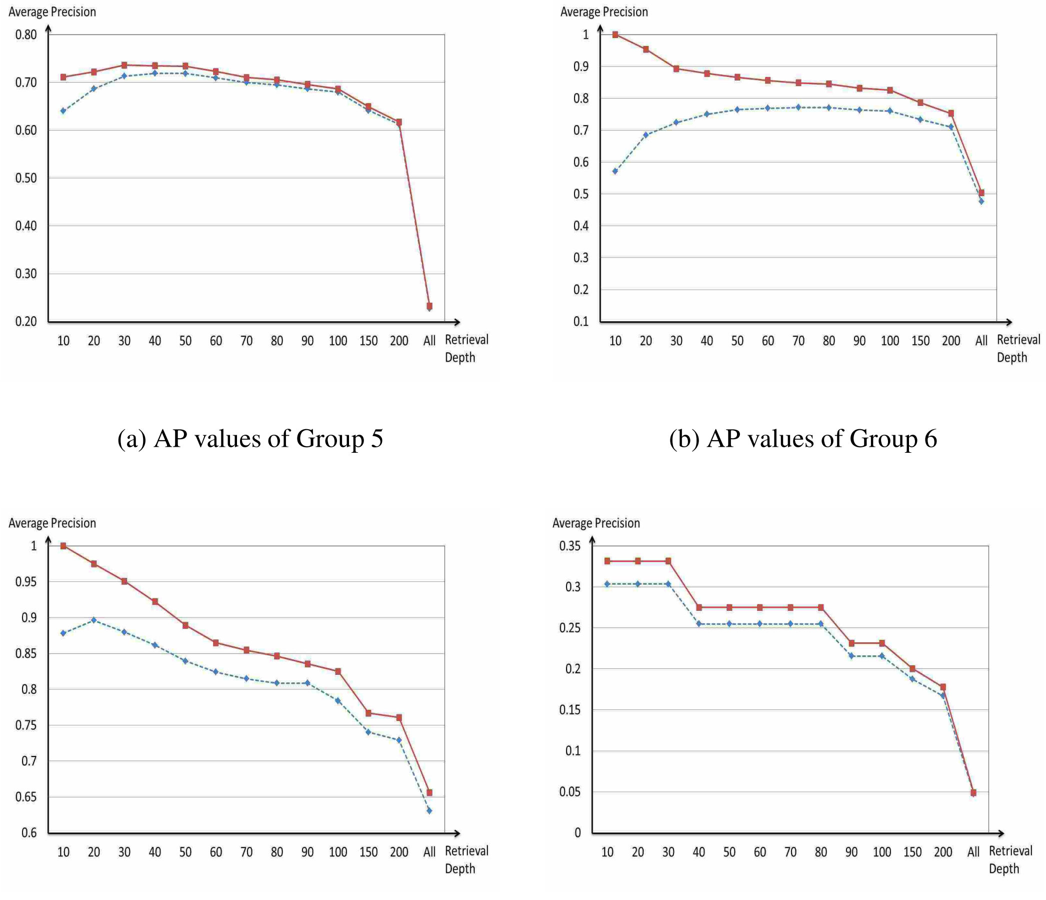
(a) AP values for Group 1

(b) AP values for Group 2

(c) AP values for Group 3

(d) AP values for Group 4

Figure 3.26: Average precision values of Groups 1 to 4



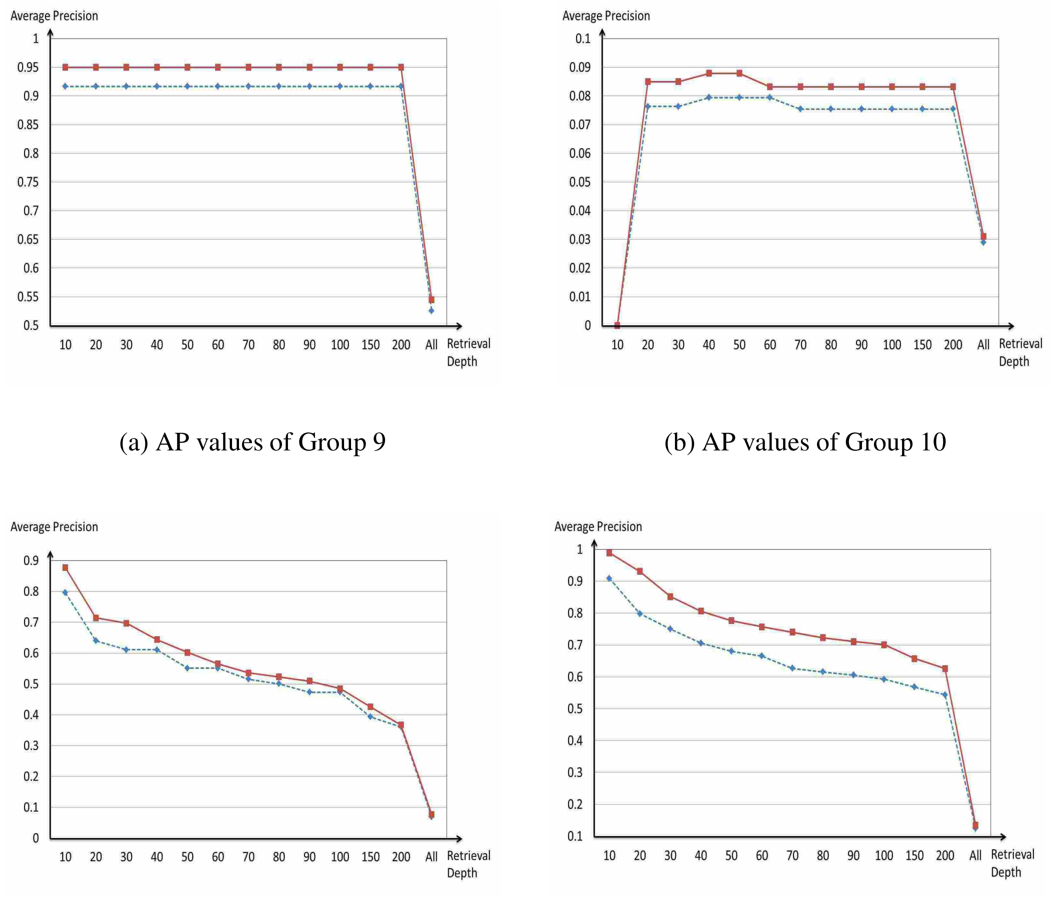
(a) AP values of Group 5

(b) AP values of Group 6

(c) AP values of Group 7

(d) AP values of Group 8

Figure 3.27: Average precision values of Groups 5 to 8



(a) AP values of Group 9

(b) AP values of Group 10

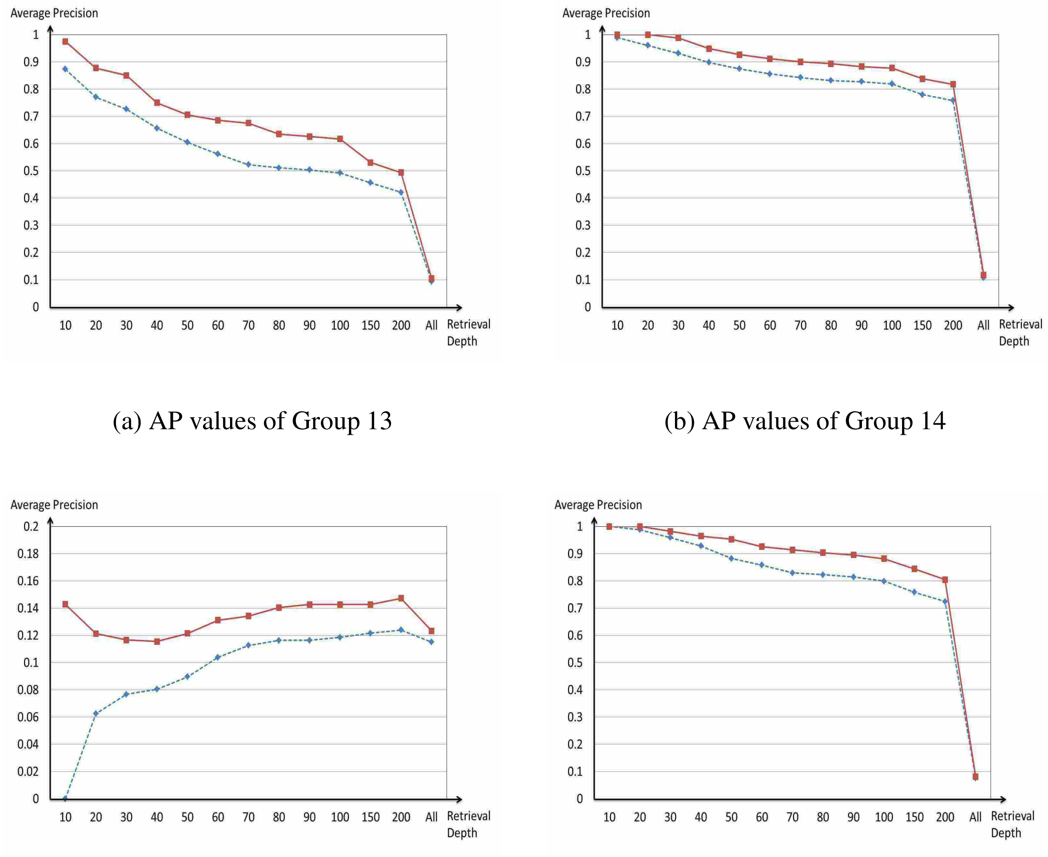
(c) AP values of Group 11

(d) AP values of Group 12

Figure 3.28: Average precision values of Groups 9 to 12

"government_leader", and "road" (solid line). The AP values of Group 10 - the concept "racing" only (dash line) and "racing" without the concept "government_leader" (solid line) are shown in Figure 3.28(b). In Figure 3.28(c), the AP values of Group 11 are given - the concept "car" only (dash line) and "car" without the concept "face" (solid line). Figure 3.28(d) then displays the AP values of Group 12 - the concept "daytime_outdoor" only (dash line) and "daytime_outdoor" without the concept "vehicle" (solid line).

In Figure 3.29(a), the AP values of Group 13 are demonstrated - the concept "ground _vehicle" only (dash line) and 'ground_vehicle" without the concepts "actor", "face",



(a) AP values of Group 13

(b) AP values of Group 14

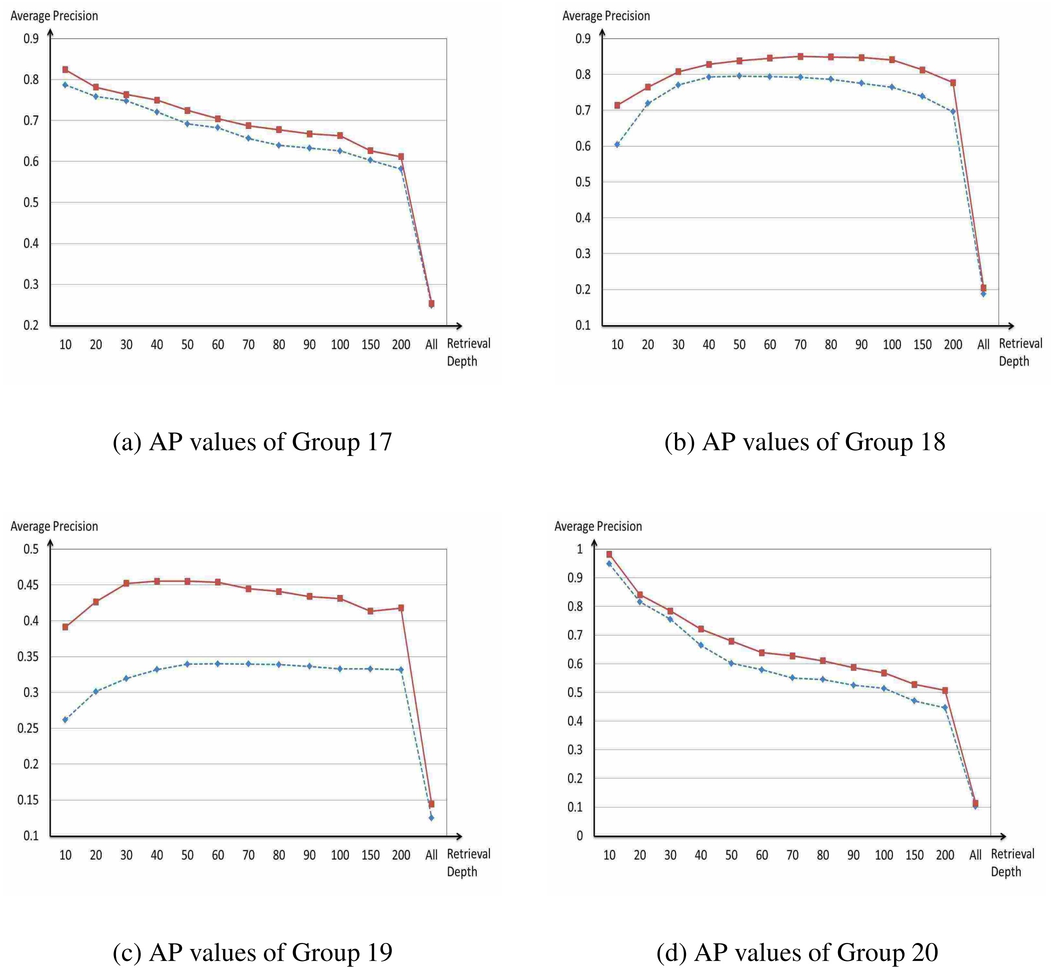
(c) AP values of Group 15

(d) AP values of Group 16

Figure 3.29: Average precision values of Groups 13 to 16

"indoor", and "male_person" (solid line). The AP values of Group 14 are shown in Figure 3.29(b) for the concept "indoor" only (dash line) and "indoor" without the concept "outdoor" (solid line). Figure 3.29(c) gives the AP values of Group 15 - the concept "landscape" only (dash line) and "landscape" without the concepts "actor", "face", and "indoor" (solid line). The AP values of Group 16 for the concept "male_person" only (dash line) and "male_person" without the concept "actor" (solid line) are presented in Figure 3.29(d).

The AP values of Group 17 with the concept "person" only (dash line) and "person" without the concept "actor" (solid line) are displayed in Figure 3.30(a). Figure 3.30(b)



(a) AP values of Group 17

(b) AP values of Group 18

(c) AP values of Group 19

(d) AP values of Group 20

Figure 3.30: Average precision values of Groups 17 to 20

shows the AP values of Group 18 - the concept “reporter” only (dash line) and “reporter” without the concept “actor” (solid line). In Figure 3.30(c), the AP values of Group 19 - the concept “road” only (dash line) and “road” without the concepts “adult” and “indoor” (solid line) are given. The AP values of Group 20 are presented in Figure 3.30(d) for the concept “single_person” only (dash line) and “single_person” without the concept “outdoor” (solid line).

The AP values of concepts in Group 1 to Group 20 show that the proposed semantic concept retrieval framework with the help of the proposed ranking algorithm is able to return more relevant retrieved results to the users when compared to the results retrieved by the baseline. For example, in Figure 3.26(a), the AP values of the ranked results are almost 30% better at the retrieval depths of 10 to 40 than those of the baseline. Even at the retrieval depth of 200, the AP value of ranked result is still 20% better than that of the baseline. As shown in Figure 3.26(c) and Figure 3.27(b), the AP value for the top 10 is almost 40% better than that of the baseline, returning the end users with significantly more relevant results. Considering the fact that the users are most interested in the first few retrieved results, the ranked results are able to better meet the needs of the users.

The benefits of applying the proposed ranking algorithm are outstanding in two aspects: (1) for those target concepts that are not so difficult to be retrieved correctly, the proposed ranking algorithm could further improve their retrieval performance, even to be perfect at some retrieval depths (as can be seen from Figure 3.26(d) and Figure 3.29(b)); and (2) the proposed ranking algorithm is able to retrieve the relevant instances to a target concept even if the baseline cannot detect any relevant instances at some depths. For example, the AP value at the first 10 for the target concept "Landscape" in Figure 3.29(c) is 0 in the baseline, but the AP value at the first 10 for the proposed framework with the ranking algorithm can reach about 14%.

In addition to the aforementioned promising results, it is also noticeable that the AP values between the ranking method and the baseline get closer as the number of retrieved data instances increases. The phenomena can be explained from two sides. First, with the increase of the retrieved instances, the number of misclassified data instances becomes larger and larger, compromising the retrieval performance in terms of AP. Second, the correlation in terms of the impact weights might not help much to boost the ranking positions of some misclassified instances with a lower rank since the attribute values of these positive data instances may be either on the margin or deeply in the distribution area of the negative data instances, resulting in the difficulty to retrieve them from a group of negative data instances.

Finally, with regard to time complexity issue, the time complexity of the proposed ranking strategy, as well as the the one mentioned in Chapter 3.4.1 is estimated to be around $O(\zeta^3)$ for the training step, where ζ is the number of the generated feature-value pairs. Such time complexity is $O(1)$ for the testing step.

Chapter 4

A Prototype of a Web-based Semantic Concept Retrieval System

In this chapter, a detailed design and implementation of the prototype of a web-based semantic concept retrieval system is elaborated. Chapter 4.1 briefly introduces the overall design of the prototype. Chapter 4.2, Chapter 4.3, and Chapter 4.4 illustrate and discuss the design and implementation of the database layer, user interface, and application layer, respectively.

4.1 Overall Design

The overall design follows the typical framework of Apache Struts [116]. Apache Struts is a very popular open-source framework for developing Java EE (Java Platform, Enterprise Edition) web applications. The idea of struts is originated from the model-view-controller (MVC) design pattern, where the model (the component interacting with a database) is separated by the view (such as JavaServer Pages (JSP)) and connected by a controller (a servlet called ActionServlet). As can be seen from Figure 4.1, it consists of three important components, namely the database layer, application layer, and user interface. The whole framework must be deployed on a web server such as Apache Tomcat [117] before it takes effect. The user interface receives users' requests and sends them to the application layer. Usually, JSP are used to dynamically generate web pages to receive users' submitted requests and/or display the results to end users. The data related to users' requests are stored in different "Forms". In application layer, the "ActionServlet" (serving as the controller) receives the requests and dispatches each request towards a corresponding action according to the configuration information stored in "struts-config.xml". An "Action" fulfills the requests received by the "ActionServlet", which might include saving users' input data into a database or retrieving desired information from a database. In the database layer, features extracted from images and concept labels are stored in corresponding tables within the database. The following sections will describe each component in details.

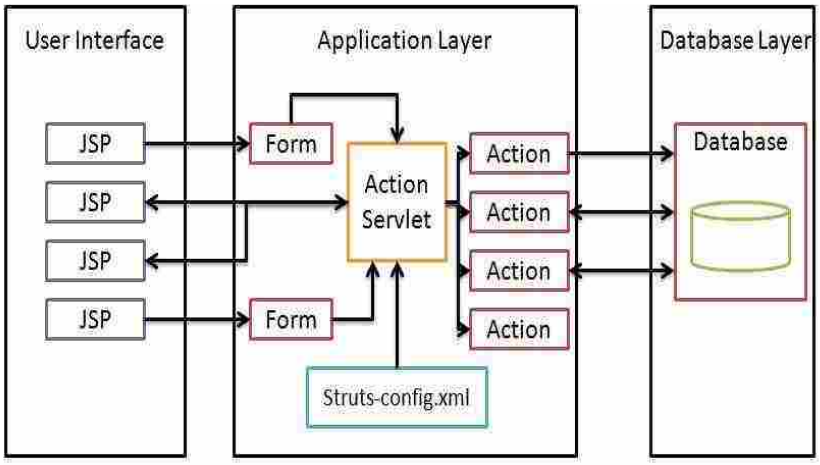


Figure 4.1: A typical framework of Struts

4.2 Database Layer Design

The database layer stores the information related with images and their corresponding semantic concepts. The entities involved in the database of semantic concept retrieval and their corresponding explanation are shown as follows.

- **images**: the entity "images" is used to store the information related to each image, such as images' names and IDs.

- **concepts**: the entity "concepts" stores the names and explanation about the semantic concepts.

Table 4.1: Relationships between all entities

Entity	Entity	Relationship
images	features	one-to-one
images	concepts	many-to-many
concepts	models	one-to-one

- **features**: the entity "features" contains a number of low-level features that are extracted from images.

- **models**: the entity "models" is created to store the training models that are related to semantic concepts.

The relationships between all these entities are shown in Table 4.1. One image is only allowed to have no more than one set of features (349 dimensions in the current case). For each image, it may contain more than one concept. For example, an image describing an urban area may contain semantic concepts such as "sky", "car", "building", and etc. Besides, a certain concept could appear in multiple images. Thus, there is a "many-to-many" relationship between "images" and "concepts". Finally, one training model will be created per concept and the characteristic of the training model is going to be described by the entity "models".

During the implementation step, one table is created for each entity. Figure 4.2 shows the final design of the entity-relationship diagram. As can be seen from this figure, the "many-to-many" relationship between entity "images" and "concepts" now involves three tables, namely "images", "concepts", and "imagesconcepts" and is simplified to two "many-to-one" relationships. Such a simplification makes it easy to store the label information between the images and concepts. However, as will be seen later, such a simplification will create a lot of redundancy and sacrifice the efficiency of semantic concept retrieval. The detailed design about each table in Figure 4.2 is shown from Table 4.2 to Table 4.6. The table "models" stores the data involved in the training

Table 4.2: Design of Table “images”

Attribute	Type	Primary Key	Not Null
ImageID	INTEGER	Yes	Yes
ImageName	VARCHAR(20)	No	Yes

Table 4.3: Design of Table “concepts”

Attribute	Type	Primary Key	Not Null
CID	INTEGER	Yes	Yes
ConceptName	VARCHAR(20)	No	Yes

models of subspace classifiers. Specifically speaking, the stored models are the objects containing the output from training phase of subspace modeling (see CODE 3.2 of Chapter 3.3.2 for details), including the eigenvalues and corresponding eigenvectors, and more.

One thing needs to be mentioned here is that such a design of database ensures the consistency of the data stored in the database. However, when extracting features from the database and applying the subspace classifier, it may involved a number of “join” operation between different tables. For example, the creation of input of the subspace classifier (training data and their corresponding training labels) requires the “join” operations between Table “images”, Table “concepts”, and Table “imagesconcepts”. Due to the huge number of images in real applications, such a “join” operation results in a slow response time or an intolerable training time. Therefore, a temporary table is generated to store the label information of each image to improve the efficiency of data retrieval

Table 4.4: Design of Table “features”

Attribute	Type	Primary Key	Not Null
ImageID	INTEGER	Yes	Yes
A_1	Double	No	No
A_2	Double	No	No
...	...	...	...
A_349	Double	No	No

Table 4.5: Design of Table “models”

Attribute	Type	Primary Key	Not Null
ImageID	INTEGER	Yes	Yes
Model_pos	BLOB	No	No
Model_neg	BLOB	No	No

Table 4.6: Design of Table “imagesconcepts”

Attribute	Type	Primary Key	Not Null
ImageID	INTEGER	Yes	Yes
CID	INTEGER	Yes	Yes
label	CHAR	No	No

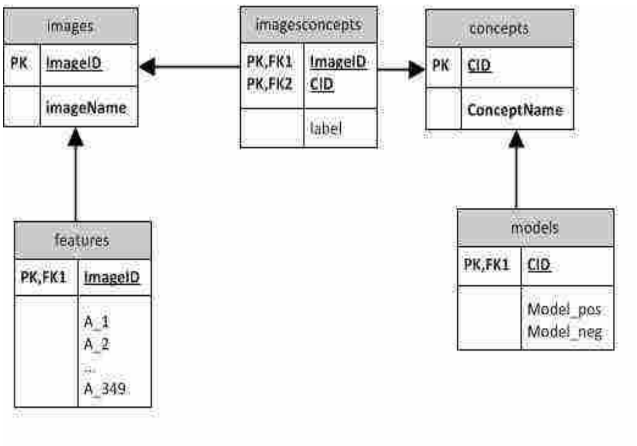


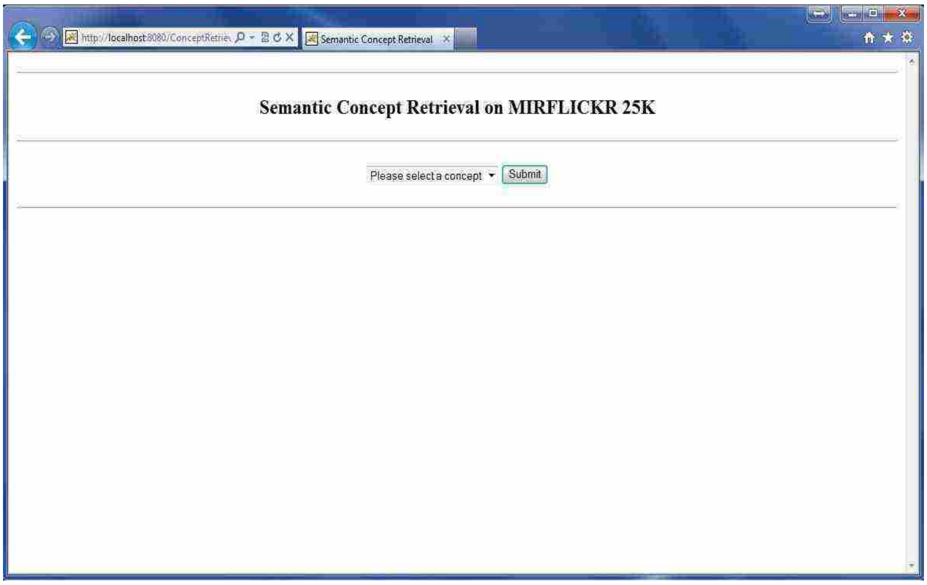
Figure 4.2: Entity-relationship diagram

Table 4.7: Design of Table "imagelabels"

Attribute	Type	Primary Key	Not Null
ImageID	INTEGER	Yes	Yes
Concept_1	CHAR	No	Yes
Concept_2	CHAR	No	Yes
...	...	...	...
Concept_24	CHAR	No	Yes

for training or testing. The temporary table can be created either by joining three tables - "images", "concepts", and "imagesconcepts" or by directly importing from outer files with images' IDs and concept labels. Table 4.7 shows an example of the temporary table "imagelabels", where all 24 concept labels in the MIRFLICKR25K image datasets are saved in different columns. For the complete list of these concepts, please refer to Table 4.8.

4.3 User Interface Design

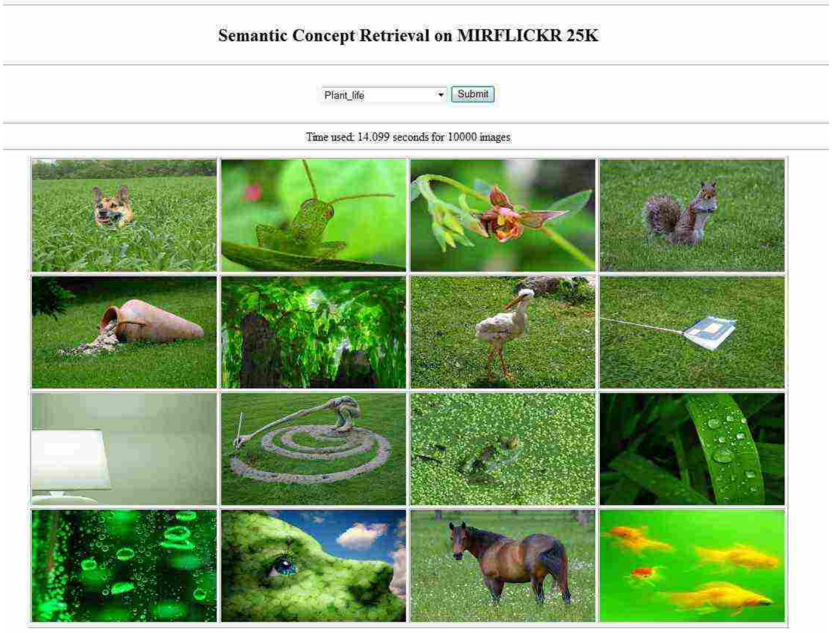


Semantic Concept Retrieval on MIRFLICKR 25K

http://localhost:8080/ConceptRetrie
Semantic Concept Retrieval
Please select a concept
Submit

Figure 4.3: Home page of concept retrieval prototype

User interface is the front end of the concept retrieval prototype. The role of the user interface is to interact with users, such as taking requests from users and responding



Semantic Concept Retrieval on MIRFLICKR 25K

Plant_life

Time used: 14.099 seconds for 10000 images

Figure 4.4: Retrieval results of concept “plant_life”

to users with the returned results. In the prototype, JSP serve as the web interface where users can select a certain semantic concept of interest to retrieve or view the returned images related to the query concepts. Figure 4.3 shows the snapshot of the homepage of the semantic concept retrieval prototype. And Figure 4.4 displays the first 16 images returned by the prototype that are related to the query concept “plant_life”. The MIRFLICKR25K dataset contains 24 unique semantic concepts, which are listed in Table 4.8. In the user interface, a user can issue a semantic concept query by selecting one of the 24 concepts (such as “plant_life”) listed in the ”drop down box”, which are indicated by the words “Please select a concept” (see Figure 4.3). The prototype receives the query request issued by the user and responds with a list of returned images that are predicted to contain the target concept (such as “Plant_Life”) identified by the user (see Figure 4.4). The JSP codes related to the capture of the users’ selection results with regard to their semantic concepts are shown as follows. As will be seen later, the application later will handle users’ requests to retrieve their interested concepts.

CODE 4.1: JSP CODE OF SEMANTIC CONCEPT SELECTION

```
<html:form action="/retrievalConcept">
  <html:select property="concept" size="1">
    <html:option value="0">Please select a concept</html:option>
    <html:option value="1">sky</html:option>
    <html:option value="2">water</html:option>
    . . .
    <html:option value="24">car</html:option>
  </html:select>
  <html:submit/>
</html:form>
```

4.4 Application Layer Design

The aforementioned database layer and user interface can be regarded as back end and front end respectively. Such a separation between the user interface and database layer can simplify the design of the prototype by considering the front end and back end as two independent modules. In addition to this advantage, the maintenance of each independent module is much easier than that of the whole prototype. The application layer serves as the bridge to connect the user interface and the database layer. In the application layer, there is one controller and the corresponding "Actions" classes to handle the requests received through user interface. The controller dispatches a user's request to a corresponding "Action" class by looking up the action mappings in the "struts-config.xml". An example of the action mapping configuration is shown in CODE 4.2.

Table 4.8: All 24 concepts and their IDs in MIRFLICKR25K dataset

Concept ID	Concept Name
1	sky
2	water
3	portrait
4	night
5	female
6	sunset
7	clouds
8	flower
9	indoor
10	male
11	plant_life
12	dog
13	structures
14	transport
15	tree
16	people
17	animal
18	sea
19	baby
20	river
21	lake
22	food
23	bird
24	car

CODE 4.2: ACTION-MAPPING CONFIGURATION

```
  <action-mappings >
    <action
       attribute=”retrievalConceptForm”
       input=”/form/retrievalConcept.jsp”
       name=”retrievalConceptForm”
       path=”/retrievalConcept”
       scope=”request”
       type=”edu.miami.ddm.action.RetrievalConceptAction”>
       <set-property property=”cancellable” value=”true” />
       <forward name=”success” path=”/form/retrievalConcept.jsp” />
    </action>
 </action-mappings>
```

The role of the “Action” classes is to fulfill the request made by the users. In semantic concept retrieval, this request can be detailed into a serious of processes, such as retrieving features from database, calling subspace classifiers to get ranking scores and returning the ranked image list to the user interface. The example code of the “Action” class fulfilling the semantic concept retrieval task is shown in the following codes.

CODE 4.3: ACTION CODE TO PERFORM SEMANTIC CONCEPT RETRIEVAL

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
  HttpServletRequest request, HttpServletResponse response) {
  RetrievalConceptForm retrievalConceptForm = (RetrievalConceptForm) form;
  // step 1: get the concept ID from input forms
  String conceptValue=retrievalConceptForm.getConcept();         ...
  // step 2: get subspace models from database
  ArrayList listFromDatabase = (ArrayList) a2d.readJavaObject(conceptID);
  TPCC_Model tpm_pos=(TPCC_Model) listFromDatabase.get(0);
  Trecvid_PCC tpcc_pos_rec=a2d.getTrecvid_PCCFromMode(tpm_pos);
  TpCC_Model tpm_neg=(TPCC_Model) listFromDatabase.get(1);
  Trecvid_PCC tpcc_neg_rec=a2d.getTrecvid_PCCFromMode(tpm_neg);
  ...
  // step 3:classify the testing instance and get the ranks
  Trecvid_SMC smc_rec=new Trecvid_SMC();
  smc_rec.classify(tpcc_pos_rec, tpcc_neg_rec, testData);
  ...
}
```

From the perspective of functionality, all the classes in the application layer fall into three categories: independent, front end-related, and back end-related. The front end-related classes deal with the interaction between the user interface and the application layer. To be specific, these classes get the input from the user interface and returns the user interface with corresponding results. For example, “ActionForms” classes are created to store the data input by users. The “ActionForms” classes are built in struts framework. The main goal of these “ActionForm” classes is to use a standard data structure that are recognized by struts to pass parameters between the front end to the different “Actions”, where the requests issued by the users are actually handled. These “ActionForms” classes allow some validation mechanisms but the most prevailing validation mechanism tends to perform at the user interface side to reduce the response time and improve users’ experiences. The following codes display a simple “ActionForm” class used in step 1 of CODE 4.3.

CODE 4.4: ACTIONFORM TO RETRIEVE THE INPUT CONCEPT

```
public class RetrievalConceptForm extends ActionForm {
  private String concept;
     public String getConcept() {
        return concept;
     }
     public void setConcept(String concept) {
        this.concept = concept;
     }
}
```

The back end-related classes interact with the database layer, managing the input and output operation from the database. In the step 2 of the “Action” code, Java Database Connectivity (JDBC) is utilized to access the database, which provides an API for the Java programming language to query and update the database. SQL queries are embedded in the Java codes and the data within the database are returned as Java Objects. The following codes show the way to retrieve the subspace models saved in the format of BLOB in MySQL database and the preparation of testing data from the features and labels stored in the database.

CODE 4.5: Java codes to retrieve model from the database

```
 ResultSet rs = pstmt.executeQuery();
byte[] buf = rs.getBytes(1);
ObjectInputStream objectIn = null;
if (buf != null)
    objectIn = new ObjectInputStream(new ByteArrayInputStream(buf));
Object posModel = objectIn.readObject();
```

CODE 4.6: create testing data from the database

```
 InstanceQuery query;
Instances data=null;
try {
    query=new InstanceQuery();
    ...
    String queryStr=“SELECT f.* , concept_”+conceptID+
    “ FROM features f, labels l WHERE f.ImageID=l.ImageID AND ”;
    ...
    query.setQuery(queryStr);
    data=query.retrieveInstances(); //retrieve instances from database
    ...
} catch (Exception e) {
    ...
}
```

The independent classes in the application layer mainly refer to those classes that are not directly related to either the user interface or the database layer, such as the classes related to the subspace classifier, which takes the feature of the instances object (a class recognized by “WEKA” [112]) prepared by the back-end related classes and outputs the ranking scores of each testing image. Some independent classes are from third parties, such as “WEKA”. The class mentioned in step 3 of CODE 4.3 (Trecvid_SMC) is a typical representative of the independent classes.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Multimedia information retrieval is currently a popular research area. However, there are three challenges that should be addressed: bridging the semantic gap, handling the data imbalance, and achieving effective semantic concept detection and retrieval. In this dissertation, these three challenges are addressed by utilizing a subspace modeling-based framework with the help of inter-concept relationships. Novel classification methods are proposed to address the semantic gap issue, which are a multi-class supervised classification method called multi-class subspace modeling (MSM) and a binary-class supervised classification method called binary-class subspace modeling (BSM). Based on MSM and BSM, another framework called subspace modeling using the global and local structures (SMGL) is proposed, which not only considers the global dissimilarity as MSM and BSM do, but also takes account of the local similarity hidden in feature-value pairs. Comparative experiments with various well-known supervised classification methods on UCI data sets and TRECVID data sets have demonstrated that MSM stably maintains the highest accuracy rate on all data sets with multiple classes. BSM and SMGL are evaluated on the TRECVID benchmark data sets and they have shown very promising experimental results to detect the high-level semantic concepts within the video shots.

For the challenge of the data imbalance issue, a new clustering-based binary-class subspace modeling classification framework called CLU-SUMO is proposed. The proposed framework utilizes the K-means clustering method to cluster the negative training data set into K different negative groups. Then each negative group is combined with the positive training data to construct K new balanced data groups on which subspace models are trained. The CLU-SUMO framework is demonstrated to be effective according to comparative experiments with other well-known classification algorithms. To improve further the efficiency of CLU-SUMO and assign semantic meanings to each balanced data group, a new framework called class selection and clustering-based subspace modeling (CSC-SUMO) is proposed. CSC-SUMO utilizes both the non-target class selection and the K-means clustering method to divide the negative training subset into $L+K$ different negative groups. L negative groups are from non-target class selection and K negative groups are from K-means clustering. Each negative group is combined with the positive training subset to construct $L+K$ new balanced data groups, each of which is trained using a subspace modeling method. From the experimental results, the proposed CSC-SUMO framework shows its effectiveness by producing competitive results against the other well-known learning methods for imbalanced data sets.

For the last challenge, a new ranking framework that utilizes the co-occurrence relationships between two semantic concepts is proposed. The proposed framework creates two co-occurrence classes (*PP* and *N*) based on a target concept and its reference concept. The co-occurrence class *PP* is composed of the training instances belonging to both the target and the reference classes, while the co-occurrence class *N* is formed by the rest of the training instances. A supervised discretization step converts the numeric input data into feature-value pairs and generates a discretization table that maps each feature-value pair with a partition interval. Next, MCA is utilized to generate the correlation tables of the feature-value pairs and the co-occurrence *PP* from the discretized training data instances. The correlation table and the discretization table generated during discretization step are looked up for testing data instances to get the impact weights and final ranking weights. Finally, the scores from the learning model are ranked by multiplying the ranking scores with the summation of the impact weights. Experimental results show that the proposed ranking framework has promising improvements on the average precision values in semantic concept retrieval, especially for the first 10, 20 and 30 retrieved results. To reduce the interference of domain knowledge in selecting the reference concept, and also to consider both the "inclusive" and the "exclusive" relationships between the target concept and the reference concepts, another ranking framework is proposed, which aims to find the best co-occurrence class from the combinations of the target concept and reference concepts to rank the retrieval results. Experimental results reveal its merits to improve the mean average precision on the retrieval results at different retrieval depths.

5.2 Future Work

Due to several limitations of the proposed frameworks and to explore new directions, several future works will be explored as follows.

5.2.1 Fusion of the Content-based and the Context-based Semantic Concept Retrieval Models

In semantic concept retrieval, there are two categories of models: content-based models and context-based models. The content-based models are built directly on the low-level features extracted from the images or videos and suffer from the "semantic gap" issue. The context-based models are built from the context information attached with the videos, such as captions, meta data, tag, and etc, which avoids the semantic gap issue but this context information is often incomplete and noisy. The content-based and the context-based models contain information supplementary to each other and it is interesting to integrate them to get better performance than each model alone. To integrate the content-based and context-based information, most of the existing approaches adopted the late fusion method to combine the results from several models trained by visual and context features. The straightforward fusion method is to apply the product, minimum, maximum, average, or median rule, which can be considered as the special case of a generic fusion model. Suppose there are M models that produce the posterior probability $P_m(w|x_n)$ as ranking score, where w is the target concept and x_n stands for the n^{th} instance. The generic model is shown in Equation (5.1).

$$P(w|x_n) = \sum_{m=1}^{M} \varphi_n(x_n) \cdot P_m(w|x_n). \tag{5.1}$$

The average rule is where $\varphi_n(x_n)$=$1/M$. For the maximum rule, $\varphi_n(x_n)$ is shown in Equation (5.2), where $M^{'}$ is the number of models whose $P_m(w|x_n) = \max_{m=1,\dots,M} P_m(w|x_n)$.

$$\varphi_n(x_n) = \begin{cases} \frac{1}{M^{'}}, & if P_m(w|x_n) = \max_{m=1,\dots,M} P_m(w|x_n) \\ 0, & otherwise \end{cases} \tag{5.2}$$

Fusion rules like minimum and median can also be written in a similar manner. The majority voting rule is only applicable when the class label is available. The “product” rule, though a little bit complicated, is still able to be represented by such a generic model in which $\varphi_n(x_n)$ is shown in Equation (5.3).

$$\varphi_n(x_n) = \frac{1}{M} \prod_{v=1\dots M, v\neq m} P_m(w|x_n). \tag{5.3}$$

The purpose of model fusion is trying to achieve performance gains from all the individual models. In the proposed fusion strategy, the form of the generic model is adopted but the parameter $\varphi_n(x_n)$ from Equation (5.3) is used. Here, a few intuitive parameters closely related to the final retrieval performance are proposed.

- ρ: an adjusted parameter to balance the ranking scores from different models. The reason why this parameter is introduced lies in that the ranking scores from diverse models could have a wide range of values, even if they use the same modeling method.

- δ: the reliability of a model to the final retrieval performance. This parameter reveals the retrieval performance of the learning models from the view of the training instances. Intuitively, the model with good performance will be assigned a relatively large weight value.

- η: the correlation of an interval of scores within a ranking model to the target concept. Within the same ranking model, the scores are distributed within a certain interval. In the case of utilizing probabilistic estimation as the ranking score, the range of scores is between 0 and 1. Suppose that the range is partitioned into several intervals, where each interval is disjointed with each other. The different correlation values between each interval and the target concept imply the way that the scores from different models should be combined at an interval level.

Based on these intuitive parameters, the proposed fusion model is shown in Equation (5.4), given a testing image $x^{'}$. Again, assuming there are M models that produce posterior probability $P_m(w|x_n)$ as the ranking score, w is the target concept, and x_n stands for the n^{th} instance.

$$P(w|x^{'}) = \sum_{m=1}^{M} \frac{\delta_m \cdot \eta_m}{\delta_m + \eta_m} \cdot (\rho_m^{-1} \cdot P_m(w|x^{'})). \tag{5.4}$$

The strategy to decide the value of these parameters is given as follows. All of them are driven by the images in the training set. ρ_m is introduced to prevent large ranking scores from dominating the small scores in the framework, and is decided by the following equation.

$$\rho_m = \frac{1}{N}\sum_{n=1}^{N} P_m(w|x_n). \tag{5.5}$$

δ_m is set to be the mean average precision of model m evaluated on the images in the training set. The method to decide η_m is as follows. First, the intervals are partitioned from the ranking scores, model by model, based on information entropy maximization (IEM) [111]. IEM selects the first cut-point that minimizes the entropy function over all possible candidate cut-points and recursively applies the strategy to both induced intervals. The minimum description length (MDL) principle is employed to determine whether to accept a selected candidate cut-point or not, and thus stops the recursion if the cut-point does not satisfy certain conditions.

Once an intervals of scores is partitioned, the correlation can be captured by applying MCA. In other words, the inputs of MCA are the partitioned scores from the content-based and tag-based models, and the output of MCA is the correlation of each partition to the target concept class for each instance/image. Therefore, η_m could be determined by Equation (5.6) where $MW_{I(x^{'})}(TC)$ is the cosine value of the angle between each interval $I(x^{'})$ where $x^{'}$ falls and the target concept class (TC). Then the final combined score can be calculated by Equation (5.4).

$$\eta_m = 0.5 \cdot (1 + MW_{I(x^{'})}(TC)). \tag{5.6}$$

5.2.2 Integration Subspace Modeling with Latent Local Inter-concept Relationships

The previous framework that considers inter-concept relationships tends to generate the ranking score from the target concepts by applying the ranking strategy once a reference concepts is selected. Such a ranking strategy (such as taking the summation of the ranking score of the target concept and that of the reference concept) is applied on all testing instances, which are also commonly adopted by peer works [118, 54]. However, this ranking strategy cannot always guarantee performance gain after the ranking strategy is applied. This might be caused by the situation when some positive instances have their ranking position increased but the negative instances also get their ranking positive higher. In the case where more negative instances hold relative higher ranking positions than the positive ones, the performance in terms of average precision is expected to become lower. Therefore, it implies that the ranking strategy needs to be instance-dependent, which means the ranking strategy should only be applied to a proportion of the instances that satisfy certain prerequisite of the ranking strategy. Therefore, a new ranking strategy will be proposed that considers the latent local inter-concept relationships within the feature-value pairs (some intervals of features), where the positive instances (containing the target concept) dominates the negative instances (not containing the target concept).

Figure 5.1 shows the way to search for and find these feature-value pairs with high positive-to-negative ratios (P/N ratios). Suppose there are N classification models built for N concepts. For m training instance, each of them has a vector of ranking scores from N models. The ranking scores of all the m training instances are then applied with supervised discretization to get a list of feature-value pairs. Then, according to the training labels of the target concept, those feature-values pairs with high positive-to-negative ratios can be identified and will be utilized in the ranking strategy.

Figure 5.2 displays an example of applying the proposed ranking strategy on a testing instance Ts. After converting Ts to a vector of feature-value pairs, which may contain the list of feature-value pairs identified in the previous step, a ranking strategy is proposed to generate the final ranking score from the ranking score of T out-

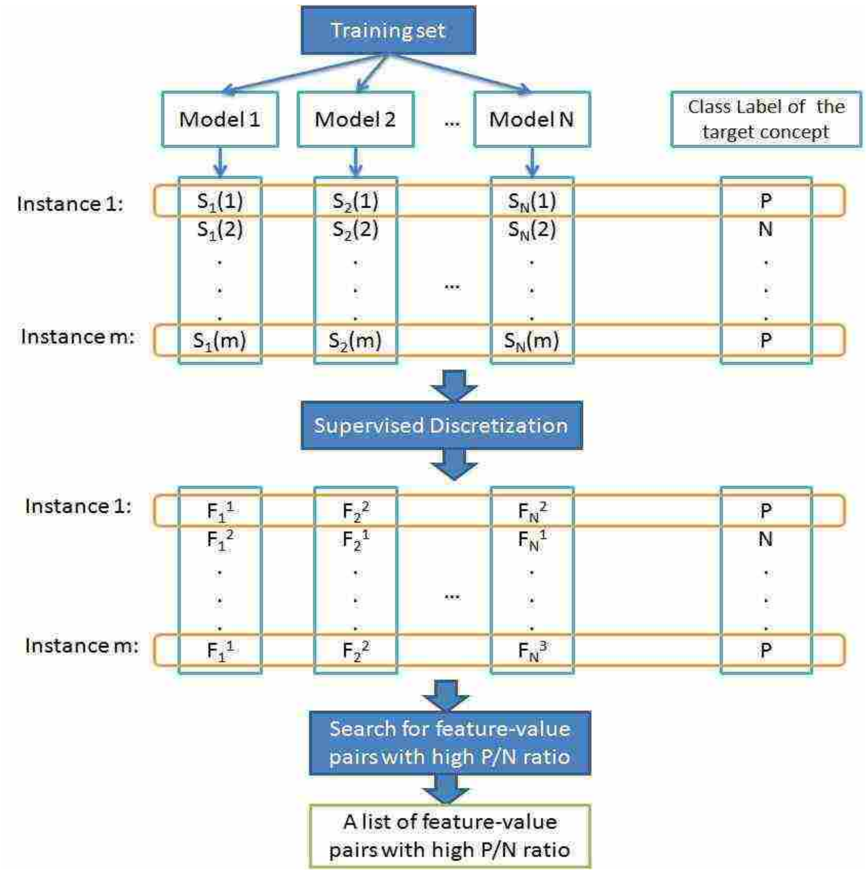


Figure 5.1: Searching for feature-value pairs with high positive-to-negative ratios

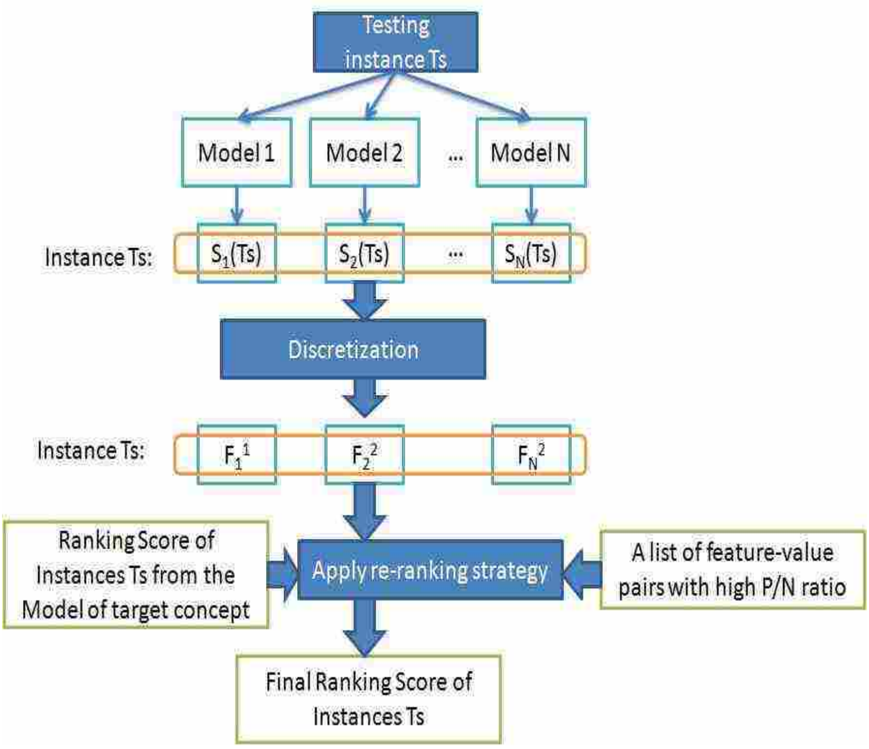

Testing instance Ts
Model 1
Model 2
...
Model N
Instance Ts:
S1(Ts)
S2(Ts)
...
SN(Ts)
Discretization
Instance Ts:
F1 1
F2 2
FN 2
Ranking Score of Instances Ts from the Model of target concept
Apply re-ranking strategy
A list of feature-value pairs with high P/N ratio
Final Ranking Score of Instances Ts

Figure 5.2: Ranking using feature-value pairs with high positive-to-negative ratios

put by the classification model of the target concept. Suppose that $F(Ts)$ ($F(Ts) = [F_1(Ts), \ldots, F_N(Ts)]$) is a vector of feature-value pairs after discretization, F_h ($F_h = [F_h^1, \ldots, F_h^q]$) is the sorted feature-value pairs with the high P/N ratios listed in an ascending manner, which means that the P/N ratio of F_h^q is no less than that of F_h^1. The following code illustrates the proposed ranking strategy:

THE PROPOSED RANKING STRATEGY

```
Deriving the co-efficiency attached to each feature-value pair in $F_h$:
    FOR Each feature-value pair $F_h^i$ in $F_h$
        Search for $\beta(F_h^i)$, the minimum $\beta$ that can maximize the average precision
        of the ranking result of training set after Equation (5.7) is applied.
    END
Apply ranking strategy on the testing instance:
SET final ranking score of $RS_j^{'}$ to be $R_j$, the ranking score of the $j$-th testing
instance for the target concept.
    FOR Each feature-value pair $F_i(Ts)$ in $F(Ts)$
        IF $F_i(Ts)$ matches one feature-value pair in $F_h$
          $RS_j^{'}$=$RS_j^{'}$+$\beta(F_i(Ts))$
        END
    END
Output $RS_j^{'}$ as the final ranking score for a testing instance $Ts$.
```

$$Rr_j^{'} = Rr_j + \delta(Tr_j) * \beta, \tag{5.7}$$

where $Rr_j^{'}$ and Rr_j is the ranking scores of the j-th training instance before and after ranking. $\delta(Tr_j)$ is 1 if Tr_j falls into F_h^i and 0, otherwise.

5.2.3 Improve the Semantic Concept Retrieval Prototype

To facilitate end users to retrieve semantic concepts from images, a web-based semantic concept retrieval prototype is designed and built in Chapter 4. The prototype allows users to query an interested semantic concept and get a list of returned images. In the future, several work will be done in the following aspects:

- include the function of retrieving videos and/or images;
- reduce the model training time and improve the training efficiency.

For the first aspect, the functionality of video retrieval will be included in the current prototype which only supports the functionality of image retrieval. Since the basic unit in a video is “shot”, it requires the information of shot boundary within a video, which implies that a table is required to store the shot boundary information in the database.

With regard to the second aspect, Chapter 3 indicates that the cost of singular value decomposition (SVD) in subspace modeling is rather expensive and SVD may not be applicable if the data set is very large. Besides, the K-means clustering method in CLU-SUMO and CSC-SUMO is also time consuming when dealing with a large scale data set. To improve the retrieval efficiency of the proposed prototype, future work will focus on the following two issues when implementing the prototype.

- How to derive eigenvalues and PCs efficiently from the training instances?
- How to cluster data instances quickly into a number of negative groups when applying K-means clustering?

Recently, with the prosperity of distributed computing methods, it is possible to perform parallel computing to distribute a task on a list of individual computers. Therefore, the proposed prototype will take advantages of the parallel computing techniques to speed up the process to derive eigenvalues and the generation of negative data groups. In the prototype, a distributed subspace modeling framework will be developed based on Hadoop [119], which uses a simple programming model to distribute the processing of large data sets across clusters of computers. Hadoop divides the whole data into a number of partitions and distributes them to different processing units. Then the eigenvalues, the PCs, and the cluster centers are calculated in parallel and merged together to increase the efficiency of the prototype.

Appendix A

Glossary

3NN K nearest neighbor with K=3

Ada AdaBoost with decision tree kernel

AdaBoost-C4.5 AdaBoost with C4.5 decision tree kernel

AdaBoost-SVM AdaBoost with SVM kernel

AP average precision

ATP attaching proportion

BP backward propagation

BSM binary-class subspace modeling

C4.5 C4.5 decision tree

CCP concept conditional probability

Chi support vector machines with chi-square kernel

CLU-SUMO clustering-based subspace modeling

CML correlative multi-label

CostDTree cost-sensitive decision tree

CP co-occurrence probability

CRSPM collated representative subspace projection

CSC-SUMO class selection and clustering-based subspace modeling

DASD domain adaptive semantic diffusion

DSM dissimilarity measure

F1 F1-score

HOG histograms of oriented gradients

IEM information entropy maximization

IREP incremental reduced error pruning

Java EE Java Platform, Enterprise Edition

JDBC Java Database Connectivity

JR JRip

JSP JavaServer Pages

KNN K-nearest neighbor

LAN local area network

LBP local binary patterns

Logistic logistic regression

MCA multiple correspondence analysis

MDL minimum description length

MFoM maximal figure-of-merit

MKL multiple kernel learning

MP multilayer perceptron

MSM multi-class subspace modeling

MVC model-view-controller

NB naive Bayes

NIST National Institute of Standards and Technology

NN nearest neighbor

PC principal component

PCA principal component analysis

PCC principal component classifier

PoN probability of negativeness

PoP probability of positiveness

pre precision

R_PC representative principal components

RBF radial basis function

rec recall

ResampleLG re-sampling with logistic regression model

RIPPER repeated incremental pruning to produce error reduction

SIFT scale-invariant feature transform

SMGL subspace-modeling on global and local structures

SMOTE synthetic minority oversampling technique

SVD singular value decomposition

SVM support vector machines

TRECVID TREC Video Retrieval Evaluation

Bibliography

[1] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, “Content-based multimedia information retrieval: State of the art and challenges,” *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 2, no. 1, pp. 1–19, February 2006.

[2] R. Datta, D. Joshi, J. Li, and J. Wang, “Image retrieval: Ideas, influences, and trends of the new age,” *ACM Computing Surveys*, vol. 40, no. 2, pp. 5:1–5:60, April 2008.

[3] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, “Content-based image retrieval at the end of the early years,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, December 2000.

[4] C. G. M. Snoek, M. Worring, J. Gemert, G. J.-M., and A. W. M. Smeulders, “The challenge problem for automated detection of 101 semantic concepts in multimedia,” in *Proceedings of the 14th Annual ACM International Conference on Multimedia (Multimedia’06)*, October 2006, pp. 421–430.

[5] Y.-G. Jiang, A. Yanagawa, S. F. Chang, and C.-W. Ngo, “CU-VIREO374: Fusing columbia374 and VIREO374 for large scale semantic concept detection,” Columbia University ADVENT #223-2008-1, Tech. Rep., August 2008.

[6] M. R. Naphade and J. R. Smith, “On the detection of semantic concepts at trecvid,” in *Proceedings of the 12th Annual ACM International Conference on Multimedia (Multimedia’04)*, October 2004, pp. 660–667.

[7] A. Hauptmann, R. Yan, W.-H. Lin, M. Christel, and H. Wactlar, “Can high-level concepts fill the semantic gap in video retrieval? a case study with broadcast news,” *IEEE Transactions on Multimedia*, vol. 9, no. 5, pp. 958–966, August 2007.

[8] P. Duygulu, K. Barnard, J. F. G. de Freitas, and D. A. Forsyth, “Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary,” in *Proceedings of 7th European Conference on Computer Vision*, May 2002, pp. 97–112.

[9] A. Kutics, A. Nakagawa, K. Tanaka, M. Yamada, Y. Sanbe, and S. Ohtsuka, "Linking images and keywords for semantics-based image retrieval," in *Proceedings. 2003 International Conference on Multimedia and Expo (ICME '03)*, July 2003, pp. 777–780.

[10] J. Fan, Y. Gao, H. Luo, and G. Xu, "Automatic image annotation by using concept-sensitive salient objects for image content representation," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '04)*, July 2004, pp. 361–368.

[11] J. Fan, Y. Gao, H. Luo, and R. Jain, "Mining multilevel image semantics via hierarchical classification," *IEEE Transactions on Multimedia*, vol. 10, no. 2, pp. 167–187, February 2008.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, November 2004.

[13] J. Uijlings, A. Smeulders, and R. Scha, "Real-time visual concept classification," *IEEE Transactions on Multimedia*, vol. 12, no. 7, pp. 665–681, November 2010.

[14] J. Wang, Y. Li, Y. Zhang, C. Wang, H. Xie, G. Chen, and X. Gao, "Bag-of-features based medical image retrieval via multiple assignment and visual words weighting," *IEEE Transactions on Medical Imaging*, vol. 30, no. 11, pp. 1996–2011, November 2011.

[15] T. S. Huang and X. S. Zhou, "Image retrieval by relevance feedback: from heuristic weight adjustment to optimal learning methods," in *Proceedings of IEEE International Conference on Image Processing (ICIP01)*, October 2001, pp. 2–5.

[16] X. He, O. King, W.-Y. Ma, M. Li, and H.-J. Zhang, "Learning a semantic space from user's relevance feedback for image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 1, pp. 39–48, January 2003.

[17] G. Guo, A. Jain, W. Ma, and H. J. Zhang, "Learning similarity measure for natural image retrieval with relevance feedback," *IEEE Transactions on Neural Networks*, vol. 13, no. 4, pp. 811–820, July 2002.

[18] H. Guan, S. Antani, L. Long, and G. Thoma, "Bridging the semantic gap using Ranking SVM for image retrieval," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, June-July 2009, pp. 354–357.

[19] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011.

[20] L. Lin and M.-L. Shyu, “Weighted association rule mining for video semantic detection,” *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, vol. 1, no. 1, pp. 37–54, 2010.

[21] R. He, N. Xiong, L. Yang, and J. H. Park, “Using multi-modal semantic association rules to fuse keywords and visual features automatically for web image retrieval,” *Information Fusion*, vol. 12, no. 3, pp. 223–230, July 2011.

[22] Y. Chen, J. Wang, and R. Krovetz, “Clue: cluster-based retrieval of images by unsupervised learning,” *IEEE Transactions on Image Processing*, vol. 14, no. 8, pp. 1187–1201, August 2005.

[23] B. Poblete, B. Bustos, M. Mendoza, and J. M. Barrios, “Visual-semantic graphs: using queries to reduce the semantic gap in web image retrieval,” in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, October 2010, pp. 1553–1556.

[24] L. Bao, J. Cao, Y. Zhang, J. Li, M.-Y. Chen, and A. G. Hauptmann, “Explicit and implicit concept-based video retrieval with bipartite graph propagation model,” in *Proceedings of the International Conference on Multimedia*, October 2010, pp. 939–942.

[25] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *Advances in Neural Information Processing Systems 16*. MIT Press, 2004, pp. 321–328.

[26] T. Joachims, “Optimizing search engines using clickthrough data,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002, pp. 133–142.

[27] B. Jyothi and S. Uma, “Neural network approach for image retrieval based on preference elicitation,” *International Journal on Computer Science and Engineering*, vol. 2, no. 4, pp. 934–941, July 2010.

[28] G. Iyengar and H. J. Nock, “Discriminative model fusion for semantic concept detection and annotation in video,” in *Proceedings of the 11th ACM International Conference on Multimedia*, November 2003, pp. 255–258.

[29] C. Wu, Y.-F. Ma, H.-J. Zhan, and Y.-Z. Zhong, “Events recognition by semantic inference for sports video,” in *Proceedings. 2002 IEEE International Conference on Multimedia and Expo*, August 2002, pp. 805–808.

[30] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, September 2009.

[31] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study,” *Intelligent Data Analysis*, vol. 6, no. 5, pp. 429–449, October 2002.

[32] K. McCarthy, K. Zabar, and G. M. Weiss, “Does cost-sensitive learning beat sampling for classifying rare classes?” in *Proceedings of the 1st International Workshop on Utility-based Data Mining*, August 2005, pp. 69–77.

[33] B. Zadrozny, J. Langford, and N. Abe, “Cost-sensitive learning by cost-proportionate example weighting,” in *Third International Conference on Data Mining (ICDM ’03)*, November 2003, pp. 435–442.

[34] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, July 2002.

[35] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, June 2005, pp. 886–893.

[36] M. Marsico and D. Riccio, “A new data normalization function for multibiometric contexts: A case study,” in *Proceedings of the 5th International Conference on Image Analysis and Recognition*, June 2008, pp. 1033–1040.

[37] Y. Ying, G. I. Webb, and X.-D. Wu, “Discretization methods,” in *Data Mining and Knowledge Discovery Handbook.* Springer US, 2010, pp. 113–130.

[38] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, “An efficient boosting algorithm for combining preference,” *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 933–963, December 2003.

[39] C. Burges, “Learning to rank using gradient descent,” in *IMLS International Conference on Machine Learning (ICML05)*, August 2005, pp. 86–96.

[40] S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua, “A MFoM learning approach to robust multiclass multi-label text categorization,” in *International Conference on Machine Learning (ICML04)*, July 2004, pp. 329–336.

[41] S. Sonnenburg, B. Ratsh, C. Schafer, and B. Scholkopf, “Large scale multiple kernel learning,” *Journal of Machine Learning Research*, vol. 7, no. 1, pp. 1531–1565, December 2006.

[42] A. F. Smeaton, P. Over, and W. Kraaij, “Evaluation campaigns and TRECVid,” in *ACM International Workshop on Multimedia Information Retrieval (MIR06)*, October 2006, pp. 321–330.

[43] X.-Y. Wei, C.-W. Ngo, and Y.-G. Jiang, “Selection of concept detectors using ontology-enriched semantic space,” *IEEE Transactions on Multimedia*, vol. 10, no. 6, pp. 1085–1096, October 2008.

[44] L. Hollink and M. Worring, “Building a visual ontology for video retrieval,” in *Proceedings of the 13th Annual ACM International Conference on Multimedia*, November 2005, pp. 479–482.

[45] K.-H. Liu, M.-F. Weng, C.-Y. Tseng, Y.-Y. Chuang, and M.-S. Chen, “Assoication and temporal rule mining for post-filtering of semantic concept detection in video,” *IEEE Transactions On Multimedia*, vol. 10, no. 2, pp. 240–251, February 2008.

[46] R. Yan, M.-Y. Chen, and A. Hauptmann, “Mining relationship between video concepts using probabilistic graphical model,” in *IEEE International Conference on Multimedia and Expo (ICME06)*, July 2006, pp. 301–304.

[47] Y.-G. Jiang, J. Wang, S.-F. Chang, and C.-W. Ngo, “Domain adaptive semantic diffusion for large scale context-based video annotation,” in *IEEE International Conference on Computer Vision (ICCV09)*, Septeber-October 2009, pp. 1420–1427.

[48] V. Vapnik, *Statistical Learning Theory*. John Wiley and Sons, 1998.

[49] J. Mercer, “Functions of positive and negative type, and their connection with the theory of integral equations,” *Philosophical Transactions of the Royal Society*, vol. 209, no. 441-458, pp. 415–446, 1909.

[50] O. Chapelle, P. Haffner, and V. Vapnik, “Support vector machines for histogram-based image classification,” *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, September 1999.

[51] T. Howley and M. G. Madden, “The genetic kernel support vector machine: Description and evaluation,” *Artificial Intelligence Review*, vol. 24, no. 3-4, pp. 379–395, November 2005.

[52] T. Jaakkola, M. Diekhaus, and D. Haussler, “Using the fisher kernel method to detect remote protein homologies.” in *International Conference on Intelligent Systems for Molecular Biology*, August 1999, pp. 149–158.

[53] P. J. Moreno, P. P. Ho, and N. Vasconcelos, “A kullback-leibler divergence based kernel for SVM classification in multimedia applications,” in *Neural Information Processing Systems*, December 2003.

[54] Y. Jiang, C. Ngo, and J. Yang, “Towards optimal bag-of-features for object categorization and semantic video retrieval,” in *ACM International Conference on Image and Video Retrieval*, July 2007, pp. 494–501.

[55] A. Bosch, A. Zisserman, and X. Munoz, “Representing shape with a spatial pyramid kernel,” in *ACM International Conference on Image and Video Retrieval*, July 2007, pp. 401–408.

[56] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[57] J. Suykens, G. Horvath, and S. Basu, *Advances in Learning Theory: Methods, Models and Applications*. IOS Press, 2003.

[58] T. M. Cover, “Estimation by the nearest neighbor rule,” *IEEE Transaction on Information Theory*, vol. 14, no. 1, pp. 50–55, January 1968.

[59] P. Vincent and Y. Bengio, “K-local hyperplane and convex distance nearest neighbor algorithms,” in *Advances in Neural Information Processing Systems*, December 2001, pp. 985–992.

[60] H. Zhang, A. Berg, M. Maire, and J. Malik, “SVM-KNN: Discriminative nearest neighbor classification for visual category recognition,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2006, pp. 2126–2136.

[61] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[62] D. P. Helmbold and R. E. Schapire, “Predicting nearly as well as the best pruning of a decision tree,” *Machine Learning*, vol. 27, no. 1, pp. 51–68, April 1997.

[63] S.-C. Chen, M.-L. Shyu, M. Chen, and C. Zhang, “A decision tree-based multimodal data mining framework for soccer goal detection,” in *IEEE International Conference on Multimedia and Expo (ICME04)*, June 2004, pp. 265–268.

[64] C. Snoek and M. Worring, "Multimedia event-based video indexing using time intervals," *IEEE Transactions on Multimedia*, vol. 7, no. 4, pp. 638–647, August 2005.

[65] W. Zhou, A. Vellaikal, and C. Kuo, "Rule-based video classification system for basketball video indexing," in *ACM Workshops on Multimedia*, 2000, pp. 213–216.

[66] S.-C. Chen, M.-L. Shyu, C. Zhang, H. Luo, and M. Chen, "Detection of soccer goal shots using joint multimedia features and classification rules," in *International Workshop on Multimedia Data Mining (MDM03)*, August 2003, pp. 36–44.

[67] L. Lin, M.-L. Shyu, G. Ravitz, and S.-C. Chen, "Video semantic concept detection via associative classification," in *IEEE International Conference on Multimedia and Expo (ICME09)*, July 2009, pp. 418–421.

[68] L. Kobylinski and K. Walczak, "Image classification with customized associative classifiers," in *International Multiconference on Computer Science and Information Technology*, November 2006, pp. 85–91.

[69] W. W. Cohen, "Fast effective rule induction," in *International Conference on Machine Learning*, July 1995, pp. 115–123.

[70] J. Furnkranz and G. Widmer, "Incremental reduced error pruning," in *International Conference on Machine Learning*, July 1994, pp. 70–77.

[71] J. Furnkranz, "A tight integration of pruning and learning," in *European Conference on Machine Learning*, 1995, pp. 291–294.

[72] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pp. 318–362, 1986.

[73] H. White, "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks*, vol. 3, no. 5, pp. 535–549, January 1990.

[74] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, August 1997.

[75] R. E. Schapire, "Using output codes to boost multiclass learning problems," in *International Conference on Machine Learning*, July 1997, pp. 313–321.

[76] V. Guruswami and A. Sahai, “Multiclass learning, boosting, and error-correcting codes,” in *Annual Conference Computational Learning Theory*, July 1999, pp. 145–155.

[77] P. Ruiz, S. D. Babacan, L. Gao, Z. Li, R. Molina, and A. K. Katsaggelos, “Video retrieval using sparse Bayesian reconstruction,” in *IEEE International Conference on Multimedia and Expo (ICME)*, July 2011, pp. 1 –6.

[78] N. Vaswani and R. Chellappa, “Principal components null space analysis for image and video classification,” *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 1816 –1830, July 2006.

[79] S.-C. Chen, M.-L. Shyu, and M. Chen, “An effective multi-concept classifier for video streams,” in *IEEE International Conference on Semantic Computing (ICSC08)*, August 2008, pp. 80–87.

[80] G. Batista, R. C. Prati, and M. C. Monard, “A study of the behavior of several methods for balancing machine learning training data,” *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, June 2004.

[81] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, June 2002.

[82] D. Mease, A. J. Wyner, and A. Buja, “Boosted classification trees and class probability/quantile estimation,” *Journal of Machine Learning Research*, vol. 8, pp. 18–36, May 2007.

[83] H. Han, W. Y. Wang, and B. H. Mao, “Borderline-smote: A new over-sampling method in imbalanced data sets learning,” in *International Conference on Intelligent Computing(ICIC 2005)*, August 2005, pp. 878–887.

[84] H. He, Y. Bai, E. A. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning,” in *IEEE International Joint Conference on Neural Networks*, June 2008, pp. 1322–1328.

[85] D. A. Cieslak and N. V. Chawla, “Start globally, optimize locally, predict globally: Improving performance on imbalanced data,” in *Eighth International Conference on Data Mining, 2008. ICDM ’08*, Decmber 2008, pp. 143–152.

[86] Y. Freund and R. Schapire, “Experiments with a new boosting algorithm,” in *Proceedings of the 13th International Conference on Machine Learning*, July 1996, pp. 148–156.

[87] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, “Smoteboost: Improving prediction of the minority class in boosting,” in *Proceedings of the Seventh European Conference on Principles of Knowledge Discovery in Databases*, September 2003, pp. 107–119.

[88] H. Guo and H. L. Viktor, “Learning from imbalanced data sets with boosting and data generation: The databoost im approach,” *ACM SIGKDD Explorations Newletter*, vol. 6, no. 1, pp. 30–39, Jun. 2004.

[89] M. A. Maloof, “Learning when data sets are imbalanced and when costs are unequal and unknown,” in *Proceedings of the ICML’2003 Workshop on Learning from Imbalanced Data Sets,Workshop Learning from Imbalanced Data Sets II.*

[90] G. M. Weiss, “Mining with rarity: A unifying framework,” *ACM SIGKDD Explorations Newletter*, vol. 6, no. 1, pp. 7–19, Jun. 2004.

[91] X. Y. Liu and Z. H. Zhou, “The influence of class imbalance on cost-sensitive learning: An empirical study,” in *Sixth International Conference on Data Mining (ICDM ’06)*, December 2006, pp. 970–974.

[92] C. Chen and M.-L. Shyu, “Clustering-based binary-class classification for imbalanced data sets,” in *12th IEEE International Conference on Information Reuse and Integration (IRI 2011)*, August 2011, pp. 384–389.

[93] C. Wang, F. Jing, L. Zhang, and H. Zhang, “Image annotation refinement using random walk with restarts,” in *ACM International Conference on Multimedia (MM’06)*, October 2006, pp. 647–650.

[94] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H. J. Zhang, “Correlative multi-label video annotation,” in *ACM International Conference on Multimedia (MM07)*, September 2007, pp. 17–26.

[95] Y. Aytar, M. Shah, and J. Luo, “Utilizing semantic word similarity measures for video retrieval,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR08)*, June 2008, pp. 1–8.

[96] M.-L. Shyu, C. Chen, and S.-C. Chen, “Multi-class classification via subspace modeling,” *International Journal of Semantic Computing*, vol. 5, no. 1, pp. 55–78, March 2011.

[97] D. Liu, M.-L. Shyu, C. Chen, and S.-C. Chen, “Within and between shot information utilization in video key frame extraction,” *Journal of Information and Knowledge Management*, vol. 10, no. 3, pp. 247–259, September 2011.

[98] A. M. Martinez and A. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228–233, February 2001.

[99] E. Parzen, "On the estimation of a probability density function and the mode," *Annals of Math.Stats*, pp. 1065–1076, 1962.

[100] E. McDerrnott and S. Katagiri, "Minimum classification error via a parzen window based estimate of the theoretical Bayes classification risk," in *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*, September 2002, pp. 415–424.

[101] R. Jenssen, "Indefinite parzen window for spectral clustering," in *IEEE Workshop on Machine Learning for Signal Processing*, August 2007, pp. 390–395.

[102] M. Singh and N. Ahuja, "Regression based bandwidth selection for segmentation using parzen windows," in *IEEE International Conference on Computer Vision*, October 2003, pp. 2–9.

[103] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag, 2002.

[104] L. Giet and M. Lubrano, "A minimum hellinger distance estimator for stochastic differential equations: An application to statistical inference for continuous time interest rate models," *Computational Statistics and Data Analysis*, vol. 52, no. 6, pp. 2945–2965, February 2008.

[105] T. Quirino, Z. Xie, M. L. Shyu, S. C. Chen, and L. Chang, "Collateral representative subspace projection modeling for supervised classification," in *IEEE Intl. Conf. on Tools with Artificial Intelligence*, November 2006, pp. 98–105.

[106] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[107] S. Hettich and S. D. Bay, "Kdd cup 1999 data," 1999. [Online]. Available: http://kdd.ics.uci.edu

[108] Z. Xie, T. Quirino, M.-L. Shyu, S.-C. Chen, and L.-W. Chang, "A distributed agent-based approach to intrusion detection using the lightweighted pcc anomaly detection classifier," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, June 2006, pp. 446–453.

[109] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapik, "Support vector regression machines," in *Advances in Neural Information Processing Systems 9*, December 1996, pp. 155–161.

[110] J. R. Quinlan, "Improved use of continuous attributes in C4.5," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 77–90, January 1996.

[111] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous valued attributes for classifcation learning," in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993, pp. 1022–1027.

[112] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, June 2005.

[113] M. J. Greenacre and J. Blasius, *Multiple Correspondence Analysis and Related Methods*. Chapman and Hall/CRC, 2006.

[114] L. Lin, G. Ravitz, M.-L. Shyu, and S.-C. Chen, "Correlation-based video semantic concept detection using multiple correspondence analysis," in *IEEE International Symposium on Multimedia (ISM08)*, December 2008, pp. 316–321.

[115] L. Lin, C. Chen, M.-L. Shyu, and S.-C. Chen, "Weighted subspace filtering and ranking algorithms for video concept retrieval," *IEEE Multimedia*, vol. 18, no. 3, pp. 32–43, March 2011.

[116] J. Holmes, *Struts: The Complete Reference, 2nd Edition*. McGraw-Hill, 2006.

[117] J. Brittain and I. F. Darwin, *Tomcat: The Definitive Guide*. O'Reilly Media, 2003.

[118] Z.-J. Zha, T. Mei, X.-S. Hua, G.-J. Qi, and Z. Wang, "Refining video annotation by exploiting pairwise concurrent relation," in *Proceedings of the 15th International Conference on Multimedia*, September 2007, pp. 345–348.

[119] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, 2009. [Online]. Available: http://www.worldcat.org/isbn/0596521979