The rapid increase in the amount of digital data has posed significant challenges to retrieval systems, which are expected to effectively and efficiently filter the data and provide relevant results. Filtering is a technique that automatically selects features or data instances to represent the data, reduce storage costs, prune redundancy, decrease computation costs, enhance model learning performance, and dynamically deliver the media. On the other hand, ranking is a critical step to render as many relevant results as possible on the basis of some loss functions or similarity measures. This article presents a novel retrieval framework that consists of weighted subspace-based filtering and ranking components to address these challenges.

One critical issue in multimedia retrieval is the difficulty of comparing semantics between low-level features (such as color and shape) and high-level concepts (such as sky and ocean). Because multiple correspondence analysis (MCA) is powerful in exploring relationships among high-dimensional categorical variables, it is attractive to apply MCA to explore relationships between feature categories and concept classes. However, existing correlation-based algorithms are weak in capturing semantics through a numeric measure such as conditional entropy to guide the search of features. Most existing subspace-based algorithms, while considering semantics, mainly focus on the representations of features in new spaces or relationships among high-level concepts. Therefore, our filtering algorithm transfers the semantic correlation captured in the subspace to the feature weights to distinguish features and imbalanced data instances.

Relevance of retrieval results is also crucial in multimedia, where effectiveness is measured by the ranking process. Our proposed algorithm is designed to overcome the limitation of the most commonly used ranking methods, which treat all data instances equally within one group during the training step. Our algorithm ranks data instances using dissimilarity values in the subspace toward both positive and negative one-class models from the learning phase. The idea behind this is that even those data instances in the same relevant group may have implicit different significances. For example, the data instances closer to the relevant group’s center could indicate more relevance than those that are further from the center. Meanwhile, a low-dimensional representation of the data instances could compactly characterize the structure of the data instance groups, making it possible to achieve an easy separation of relevant and irrelevant groups and effectively rank the retrieved results. To demonstrate the effectiveness of the proposed framework, we evaluated 30 high-level concepts and data sets from Trecvid 2008 and 2009.

Proposed retrieval framework

The proposed concept retrieval framework is shown in Figure 1. Some existing methods are used to prepare the training and testing data for each concept. The shot-based feature set has 16 audio features (including volume-based and energy-based features, and zero crossing rate), 11 visual features (including pixel-based, histogram-based, and background-based features), and one metafeature (the length of the shots). A set of 20 keyframe-based visual features
(including color, texture, and edge) are merged with the shot-based feature set as the final feature set. After data preprocessing, the extracted features for all video segments are split into two sets: a training data set (two thirds of the whole data) and a testing data set (one third of the whole data). Then the training data set is provided as the input for the filtering component. The ranking component is trained using the filtered data. Finally, the ranked testing video segments are retrieved.

Filtering component
The weighted subspace-based filtering component is presented in Figure 2. In statistics, MCA is a data-analysis technique used for categorical data to represent underlying structures in a data set by representing data as points in a low-dimensional Euclidean space. The categories of variables show the representation of individuals as points in geometric space. Associations between variables are discovered by calculating the chi-square distance between different categories of the variables and between the individuals. These associations are then represented graphically as maps that ease the interpretation of the structures in the data. In a multimedia database, the columns consist of the extracted low-level features and class labels, and the rows include the data instances, which represent media segments. To apply MCA properly, discretization is applied to convert the numeric features to nominal ones. Each partition generated from discretization is a category, called a feature-value pair.

For a given feature $A_i$, a set of data instances $S$, a break point $T$, and the total # of data instances $K$, the minimal description length (MDL) principle is used to determine a stopping criterion for splitting as shown in Equation 1, where $\Delta(A_i, T, S) = \log_2(3^T - 2) - [\tau_1\text{Ent}(S) - \tau_1\text{Ent}(S_1) - \tau_2\text{Ent}(S_2)]$, $S_1 \subset S$ and $S_2 = S - S_1$ are two subsets of data instances in $S$. $\tau$ is the number of classes, which is 2, and $\tau_1$ and $\tau_2$ are the corresponding weights, both 1 in this study. $\text{Ent}(S)$ is the class entropy defined as $\text{Ent}(S) = -P(C_{p,i} | S)(\log(P(C_{p,i} | S)) + P(C_{q,i} | S)(\log(P(C_{q,i} | S)))$.

$$\text{Gain}(A_i, T, S) > \frac{\log_2(K - 1)}{K} + \frac{\Delta(A_i, T, S)}{K}$$ (1)
If the information gain of a feature does not meet the condition in Equation 1, that is, only one partition is generated, the maximum information gain value is used to construct two partitions for that feature, as in Equation 2, where $E(A_i, T, S)$ is called the class information entropy defined as $E(A, T, S) = |S_1/|S| \text{Ent}(S_1) + |S_2/|S| \text{Ent}(S_2)$. To obtain the maximum gain $\text{Gain}(A_i, T, S)$, it minimizes the entropy function over all possible partition boundaries and selects a binary discretization boundary.

$$\text{Gain}(A_i, T, S) = \max(\text{Ent}(S) - E(A_i, T, S))$$

(2)

After the discretization process, each feature $A_i$ is represented by several feature-value pairs $A_i^j (t = 1 \ldots L, j = 1 \ldots \text{number of partitions for } A_i)$. The class is either $C_p$ (target concept class) or $C_n$ (non-target concept class). For example, $A_{i7}$ (that is, $i = 17$) is the feature of the pixel changes, which is converted to three partitions by applying discretization (that is, $j = 1$ to 3). Feature-value pairs $A_{17}^1, A_{17}^2$, and $A_{17}^3$ represent the partitions of the feature value ranges $[0, 0.32865], [0.32865, 0.5044], \text{and } [0.5044, 1]$, respectively. Then MCA is applied to calculate the correlation information between each feature-value pair and each concept class for the training data instances.

**MCA projection**

Assume that the discretized feature value of the $i$-th feature in the $k$-th data instance in the database is $d_{i,k}$, where $k = 1 \ldots K$, $i = 1 \ldots I$, $K$ is the total number of data instances, $I$ is the total number of features, and $j = 1 \ldots \text{number of feature-value pairs for feature } A_i$. Similarly, let the class label for the $k$-th data instance (class($k$) being $C_p$ or $C_n$) be $d_{i,k+1}$, where $j = 1$ or 2. Therefore, the matrix to be analyzed is $A = \{d_{i,k}\}$ where $i = 1 \ldots I + 1$.

MCA scans the discretized data to generate the indicator matrix $Z = \{z_{k,i}\}$, which is a binary representation of nominal data, where $k_1 = k$ and $i_2 = 1 \ldots \text{total number of feature-value pairs and classes}$. Each column in $Z$ represents the generated feature-value pair or class. Next, the inner product of the indicator matrix is called the Burt matrix $B = \{b_{k,i}\}$, where $k_2, i_2 = 1$ is the total number of feature-value pairs and classes. Because $B$ is symmetric, the solutions for the rows and columns are identical. Therefore, the analysis of $B$ only gives a solution for the response categories (that is, the columns of $Z$). Let the grand total of $B$ be $E = \Sigma_{k,j} \Sigma_{t,i} b_{k,i}$. Then the probability (or correspondence) matrix $P = \{p_{k,i}\}$, the mass matrix $Q = \Sigma_{k} p_{k,i} = \{q_{i}\}$ or $\{q_{j}\}$, and $D$ being the main diagonal of $Q$ are calculated. Furthermore, the chi-square distance among tabulations of the Burt matrix is applied by calculating the differences $(p_{k,i} - q_{i}q_{k})$ between the observed and expected relative frequencies and standardized by dividing these differences by $\sqrt{q_{i}q_{k}}$. Then the principal components are obtained via singular value decomposition (SVD) shown in Equation 3. The $i_2$-th row (or column) standard coordinate is obtained as $v_{i}/\sqrt{q_{i}}$ and the eigenvalue is $\lambda$. Then the corresponding principal coordinates are given by $\lambda v_{i}/\sqrt{q_{i}}$.

$$D^{\frac{1}{2}}(P - QQ^T)(D^{\frac{1}{2}})^{\frac{1}{2}} = U\Delta V^T$$

(3)

On the left-hand side,

- $P = E^{-1}B$ and $B = Z^T Z$;
- $Q$ is the column totals of $P$;
- $D = \text{diag}(Q)$.

On the right-hand side,

- let $W = U\Delta V^T$;
- the columns of $U$ are the left-singular vectors (gene-coefficient vectors), which are eigenvectors of $WW^T$;
- the rows of $V^T$ are the right-singular vectors (expression-level vectors), which are the eigenvector of $W^TW$; and
- $\Delta$ is the diagonal matrix of the singular values and $\lambda = \Delta^2$ is the diagonal matrix of the eigenvalues.

The projection to the new space is achieved by using the first principal component (the eigenvector with the largest eigenvalue) and the second principal component (the eigenvector with the second largest eigenvalue) discovered from Equation 3. The correlation between a feature-value pair and a class can be used as an indication to the similarity between them. For example in Figure 2, the projection shows that the second feature-value pair $A_{i7}$ of the feature of pixel changes has a high correlation with the target concept and
the remaining feature-value pairs of $A_i^j$ and $A_j^i$ have high correlations with the nontarget concept.

Features and feature weights

The correlation is represented as the cross-product of a feature-value pair ($A_i^j$) and class ($C_p$ or $C_n$), that is, as the cosine of the angle between them, where $i = 1 \ldots I$, $j = 1 \ldots$ number of partitions for $A_i$ and $I$ is the total number of features. This correlation measurement is popular due to its rotation invariance property in Euclidean space that MCA is able to project. Therefore, the angle between $A_i^j$ and a class provides a measure of similarity. The angle $angleP_i^j \in [0, 180]$ between $A_i^j$ and $C_p$ and the angle $angleN_i^j \in [0, 180]$ are defined in Equations 4 and 5. Also, $angleP_i^j + angleN_i^j = 180$.

\[
angleP_i^j = \arccos \left( \frac{A_i^j \cdot C_p}{|A_i^j||C_p|} \right)
\]

\[
angleN_i^j = \arccos \left( \frac{A_i^j \cdot C_n}{|A_i^j||C_n|} \right)
\]

It’s obvious that the higher correlation between a feature-value pair and a class, then the larger the corresponding inner product value and the smaller the corresponding angle. Therefore, $A_i^j$ can be kept for $C_p$ or $C_n$ if its corresponding angle value is smaller than a certain threshold value. In our study, the thresholds for positive and negative feature-value pairs are determined by an iterative process to ensure the best threshold values for the training data set can be found. From our empirical study, the iterations for positive and negative angles start from the second minimum angle values of the positive feature-value pairs and negative feature-value pairs, respectively, with an increment of 3 degrees in each iteration. The iterations end at 90 degrees. The second smallest angle values are used to make sure that at least one feature-value pair will be selected. The 3-degree increment is a small interval because the differences between angle values are small.

For selected feature-value pairs, feature weights $weightP_i^j$ and $weightN_i^j$ are calculated by using the correlation information from MCA, as shown in Equations 6 and 7. If the feature-value pair is selected, the angle value is kept; otherwise, the angle value is set to 180 so that the corresponding weight is 0.

\[
weightP_i^j = \frac{1}{2} - \frac{|angleP_i^j|}{90}
\]

\[
weightN_i^j = -\frac{1}{2} - \frac{|angleN_i^j|}{90}
\]

In other words, if a feature-value pair is not selected, the weight value should be 0.

Data filtering

In our data-filtering approach, the feature-value pairs in each data instance are checked and the sum of the weights of the selected feature-value pairs is the total score for that data instance (score(k)). As shown in Figure 3, K is the total number of data instances, I is the total number of features, and J is the number of partitions of each feature.

Figure 3. Total score for (a) each data instance and (b) threshold determination.
Next, the training set is split to a positive set and a negative set. The mean and standard deviation values of the scores for the positive and negative sets are calculated, where \(\text{num}_{\text{pos}}\) and \(\text{num}_{\text{neg}}\) are the sizes, as:

\[
\text{mean}_{\text{pos}} = \frac{1}{\text{num}_{\text{pos}}} \sum_{\text{class}=\text{pos}} \text{score}(k)
\]

and

\[
\text{mean}_{\text{neg}} = \frac{1}{\text{num}_{\text{neg}}} \sum_{\text{class}=\text{neg}} \text{score}(k)
\]

are the score mean values, and

\[
\text{std}_{\text{pos}} = \sqrt{\frac{1}{\text{num}_{\text{pos}}} \sum_{\text{class}=\text{pos}} (\text{score}(k) - \text{mean}_{\text{pos}})^2}
\]

and

\[
\text{std}_{\text{neg}} = \sqrt{\frac{1}{\text{num}_{\text{neg}}} \sum_{\text{class}=\text{neg}} (\text{score}(k) - \text{mean}_{\text{neg}})^2}
\]

are the score standard deviation values for the positive and negative sets, respectively.

Threshold determination is shown in Figure 3, where \(x\) is obtained from empirical study via searching from 0.05 to 0.5 with a step size 0.01 and the threshold value that yields the highest F1 score will be used. In this way, our approach can automatically and adaptively choose the threshold value for the total score to determine whether the data instance is positive or negative, a fuzzy data instance to be filtered out.

When \(\text{mean}_{\text{pos}} \geq \text{mean}_{\text{neg}}\), if the total score of a data instance is greater than or equal to a certain threshold, it’s predicted as a positive data instance; otherwise, it’s negative. On the other hand, when \(\text{mean}_{\text{pos}} < \text{mean}_{\text{neg}}\), if the total score of a data instance is greater than or equal to a certain threshold, it’s considered as typical positive; otherwise, it’s considered as typical negative. Therefore, the threshold values for these two cases should be different. This ensures that our approach doesn’t prune too much negative data instances but keeps as many typical positive data instances as possible.

**Ranking component**

Another project developed the collateral representative subspace projection modeling (C-RSPM) learning model, which differentiates the normal class from abnormal classes on the basis of a chi-square measure. The idea motivated us to use it to detect semantic concepts in multimedia retrieval research by considering the target concept as a normal (positive) class and nontarget concepts as an abnormal (negative) class.

Figure 4 presents the proposed weighted subspace ranking (WSR) component, extended from C-RSPM. There are two dissimilarity calculators: \(\text{DC}_P\) for training a positive one-class model of the target concept and \(\text{DC}_N\) for training a negative one-class model for nontarget concepts. Each training data instance will get a pair of dissimilarity values, namely \(\text{Dis}_{\text{Train}}_P\) and \(\text{Dis}_{\text{Train}}_N\), from \(\text{DC}_P\) and \(\text{DC}_N\), respectively. A subspace-based classifier then calculates a weighted parameter \(\beta\) and classifies the data instance by comparing \(\text{Dis}_{\text{Train}}_P\) and \(\text{Dis}_{\text{Train}}_N\) (which is \(\text{Dis}_{\text{Train}}^\beta\)). 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 (\(\text{DC}_P\) and \(\text{DC}_N\)) for testing data instances to calculate the pair of dissimilarity values (\(\text{Dis}_{\text{Test}}_P\) and \(\text{Dis}_{\text{Test}}_N\)). The ranking scores and class labels for testing data instances are predicted by \(\text{Dis}_{\text{Test}}_P\) and \(\text{Dis}_{\text{Test}}_N\).

**Dissimilarity calculators**

The dissimilarity calculators (\(\text{DC}_P\) and \(\text{DC}_N\)) consist of three steps: normalization, subspace projection, and dissimilarity calculation. In our proposed approach, a z-score normalization process is applied separately to positive and negative data instances. Taking \(\text{DC}_P\) for example, the z-score normalization scales each feature of positive training data instances to be zero mean and unit standard deviation. By means of normalization, the characteristics of positive data instances can be prevented from being dominated by a few large-scale features, and the mean and standard deviation can roughly describe the statistical information.
about the positive data instances. Equation 8 shows the z-score normalization, where \( X' \) stands for the data instances to be normalized, and \( \text{mean}(X') \) and \( \text{std}(X') \) are the mean and standard deviation of \( X' \).

\[
X = \frac{X' - \text{mean}(X')}{\text{std}(X')} \tag{8}
\]

The next step is subspace projection, whose main goal is to decorrelate the features. SVD is adopted to decorrelate features of \( X \) in the original space. Through this decomposition, a list of eigenvalue-eigenvector pairs are derived, denoted as \((\lambda_1, \text{PC}_1), (\lambda_2, \text{PC}_2), \ldots, (\lambda_l, \text{PC}_l)\). Each eigenvector here is also called a principal component (PC). 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, R_PC2, \ldots, R_PC_l) is important, because a suitable set of representative PCs will not only reduce the dimension of the data (\( \xi < l \)) but also remove the dimensions that are linearly dependent on the others (that is, dimensions \( \xi \) whose eigenvalues \( \lambda_f = 0 \)).

Our selection method for representative PCs consists of the following steps. Instances \( X = \{x_m\} \) are projected as \( Y = \{y_m\} \) onto the PC subspace as Equation 9, where the \( r \)-th column of \( Y \), \( Y_r \), satisfies the normal distribution with the variance equals \( \lambda_r \). Then confidence interval \( \varepsilon \) is used to calculate the lower and upper bounds of \( Y_r \) to reduce the influence of noisy data. The few smallest and largest values of \( Y \) will not participate in calculating the lower and upper bounds of the remaining \( Y_r \) data. A variable, \( \omega \), denotes the percentage of data instances that lie outside the boundary calculated on certain PC. All PCs are then ranked on the basis of their corresponding \( \omega \) values. The PC with the largest \( \omega \) value ranks first, the second largest follows, and so forth. Finally, the optimal combination of PCs is searched 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). The time complexity of the PC selection algorithm is only \( O(N\log(N)) \).

\[
y_{mr} = x_m \cdot \text{PC}_r = x_m_1y_{1r} + x_m_2y_{2r} + \cdots + x_m_ly_{lr} \tag{9}
\]

where \( x_m \) is the \( m \)-th data instance of \( X \) and \( \text{PC}_r \) is the \( r \)-th PC corresponding to \( \lambda_r \).

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. There is an important property, namely \( \text{mean}(SC_h) = 0 \) and \( \text{var}(SC_h) = \lambda_h \). Using this property, a dissimilarity distance measure can be defined to measure the closeness of each data instance to different training models.

Because each column is already uncorrelated as a result of subspace projection together with the aforementioned property of the projected data, a dissimilarity measure called \( \text{Dis} \), shown in Equation 10, can be obtained, where \( SC_h \) is calculated from the previous step and \( \lambda_h \) is the \( h \)-th eigenvalue from SVD (\( \lambda_l > 0 \)). This measure is the square of the chi-square distance as shown:

\[
\text{Dis}(SC) = \sum_{h} \frac{(SC_h)^2}{\lambda} \tag{10}
\]

The proof is as follows:

- Suppose there is a function \( \text{ChiS}(x) = \text{Dis}(x)^{1/2} \).
- Non-negative: it’s obvious to see that \( \text{ChiS}(x) \geq 0 \).
- Symmetric: it’s also easy to prove that \( \text{ChiS}(x - y) = \text{ChiS}(y - x) \).
- Triangle inequality: for \( n \)-dimension vectors \( x = [x_1, x_2, \ldots, x_n] \) and \( y = [y_1, y_2, \ldots, y_n] \), substituting \( (y_{h}/\sqrt{\lambda}) \) with \( x_{h}^{(\text{new})} \).

The proof of triangle inequality of \( \text{ChiS}(x) \) is the same as that of triangle inequality of \( n \)-dimensional Euclidean distance.

**Weight parameter and model evaluation**

First, let’s use \( DCP \) as an example. The dissimilarity value for positive data instances is small. However, because the negative data instances may be heterogeneous to the positive data instances in their data characteristics, they usually have large dissimilarity values, indicating their poor fitness for the positive model. The dissimilarity value therefore can be used to distinguish the negative data instances from the positive ones. The same rules hold
for positive data instances with the negative model.

On account of too many misclassifications due to data imbalance issues, by introducing a weight parameter, the algorithm can benefit the positive model, preventing it from being dominated by the negative model. The model evaluation part that tunes \( \beta \) to achieve the optimal (or near-optimal) classification performance is significant. It searches the best \( \beta \) for the training model via a small-step iteration by evaluating the F1 score of the training model. From our empirical study, \( \text{init value} = -0.3, \text{end value} = 0.3 \), and the step size \( = 0.01 \). The process to select optimal \( \beta \) can be found in Figure 5.

**Figure 5. Selection of optimal \( \beta \).**

**Ranking based on dissimilarity measure**

After applying each testing data instance to two dissimilarity calculators (\( \text{DC}_P \) and \( \text{DC}_N \)) built by training data instances, the dissimilarity values \( \text{DisTest}_P \) and \( \text{DisTest}_N \) can be obtained. By multiplying the weight parameter with the negative dissimilarity value, it assigns the data instance with positive or negative label according to its dissimilarity in relation to positive and negative models. If \( \text{DisTest}_P = \text{DisTest}_N \), 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 rules are as follows:

1. If \( \text{DisTest}_P \leq \text{DisTest}_N \), assign positive label to the data instance;
2. If \( \text{DisTest}_P > \text{DisTest}_N \), assign negative label to the data instance.

The idea behind our ranking strategy is that for a typical relevant positive data instance, it should be close to the positive model and far away from the negative model. In other words, \( \text{DisTest}_N \) should be large and \( \text{DisTest}_P \) should be small. A larger value of \( \text{DisTest}_N - \text{DisTest}_P \) 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 has in relation to the positive model and negative model. This ranking algorithm is not confined to the current retrieval framework; it can be applied to any filtering and learning algorithms as long as it could build satisfactory models for positive and negative data instances.

**Experiments and results**

The dataset used for experiments includes 219 videos with 30 semantic concepts, which are training data in the high-level feature-extraction task of Trecvid 2008 and 2009. The experiment was conducted with a threefold cross-validation evaluation. The effectiveness of MCA-filtering is evaluated through comparison of F1 scores of several well-known classifiers with original and filtered data, especially subspace PCA-filtering, and comparison of the subspace-based ranking method with original and filtered data. The performance of the framework was evaluated by comparing mean average precision (MAP) values among several popular ranking methods and other two top-ranked algorithms submitted to Trecvid.

The detailed descriptions of these high-level concepts can be found elsewhere, while some statistics information of the data is shown in Table 1. Table 1 shows that these data sets are imbalanced, with an average ratio of 0.07 between positive and negative data instances.

The fourth and fifth columns in Table 1 show that the smallest rate of feature filtering is 0.00, which means that no feature can be filtered from the proposed component, and the largest rate of data instance pruning is 0.70, which means at most 70 percent of the instances are filtered. For PCA-filtering, the rate of feature filtering may be negative because after performing PCA-based filter, linear combinations of features are given as new features by choosing enough eigenvectors to account for 95 percent of the variance in the original data. Therefore, the number of features after filtering could be smaller or larger than the original number of features. However our performed MCA-filtering could always have a smaller than or equal number of features compared to the original data. From the training efficiency point of view, the fewer features the better. Thus our proposed MCA-filtering outperforms...
Overall, on average, the proposed filtering approach is able to vertically filter 41 percent features and horizontally filter 32 percent for the 30 investigated concepts.

To show the effectiveness of our filtering approach, we conducted comparisons between the proposed MCA-filtering and three other filtering algorithms: filter 1 (resample), filter 2 (stratified remove folds), and PCA (principle components), using eight classifiers available in the Waikato Environment for Knowledge Analysis (WEKA):5

- K-th Nearest Neighbor (Ibk algorithm where K is 3),
- AdaBoost (AdaBoostM1 algorithm with C4.5 classifier),
- Decision Tree (J48 algorithm),
- Rule based Ripper (JRip algorithm),
- Neural Network (multilayer perception algorithm),
- Support Vector Machine (sequential minimal optimization algorithm),
- LibSVM classifier (support vector machine with chi-square kernel), and
- Bayes (Naive Bayes algorithm) classifier.

Table 1. Information of high-level concepts to be retrieved.

<table>
<thead>
<tr>
<th>Name</th>
<th>Positive data instances</th>
<th>Positive and negative data instance ratio</th>
<th>Feature filtering rate for MCA</th>
<th>Data instance filtering rate for MCA</th>
<th>Feature filtering rate for PCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bridge</td>
<td>160</td>
<td>0.04</td>
<td>0.56</td>
<td>0.32</td>
<td>0.16</td>
</tr>
<tr>
<td>Emergency vehicle</td>
<td>72</td>
<td>0.02</td>
<td>0.42</td>
<td>0.35</td>
<td>0.26</td>
</tr>
<tr>
<td>Dog</td>
<td>141</td>
<td>0.04</td>
<td>0.15</td>
<td>0.35</td>
<td>0.22</td>
</tr>
<tr>
<td>Kitchen</td>
<td>169</td>
<td>0.04</td>
<td>0.17</td>
<td>0.33</td>
<td>0.19</td>
</tr>
<tr>
<td>Two people</td>
<td>1,534</td>
<td>0.44</td>
<td>0.00</td>
<td>0.39</td>
<td>-0.21</td>
</tr>
<tr>
<td>Driver</td>
<td>263</td>
<td>0.07</td>
<td>0.28</td>
<td>0.32</td>
<td>0.09</td>
</tr>
<tr>
<td>Harbor</td>
<td>146</td>
<td>0.04</td>
<td>0.08</td>
<td>0.30</td>
<td>0.12</td>
</tr>
<tr>
<td>Street</td>
<td>651</td>
<td>0.17</td>
<td>0.44</td>
<td>0.34</td>
<td>-0.36</td>
</tr>
<tr>
<td>Mountain</td>
<td>257</td>
<td>0.07</td>
<td>0.17</td>
<td>0.33</td>
<td>0.21</td>
</tr>
<tr>
<td>Flower</td>
<td>283</td>
<td>0.07</td>
<td>0.00</td>
<td>0.41</td>
<td>0.23</td>
</tr>
<tr>
<td>Classroom</td>
<td>291</td>
<td>0.07</td>
<td>0.01</td>
<td>0.31</td>
<td>0.03</td>
</tr>
<tr>
<td>Chair</td>
<td>291</td>
<td>0.07</td>
<td>0.42</td>
<td>0.30</td>
<td>-0.50</td>
</tr>
<tr>
<td>Infant</td>
<td>52</td>
<td>0.01</td>
<td>0.55</td>
<td>0.29</td>
<td>0.03</td>
</tr>
<tr>
<td>Traffic intersection</td>
<td>56</td>
<td>0.01</td>
<td>0.69</td>
<td>0.23</td>
<td>-0.09</td>
</tr>
<tr>
<td>Doorway</td>
<td>206</td>
<td>0.05</td>
<td>0.65</td>
<td>0.33</td>
<td>0.15</td>
</tr>
<tr>
<td>Airplane flying</td>
<td>109</td>
<td>0.03</td>
<td>0.19</td>
<td>0.30</td>
<td>0.11</td>
</tr>
<tr>
<td>Person playing musical instrument</td>
<td>140</td>
<td>0.04</td>
<td>0.15</td>
<td>0.70</td>
<td>-1.31</td>
</tr>
<tr>
<td>Bus</td>
<td>38</td>
<td>0.01</td>
<td>0.86</td>
<td>0.19</td>
<td>0.21</td>
</tr>
<tr>
<td>Person playing soccer</td>
<td>27</td>
<td>0.01</td>
<td>0.71</td>
<td>0.02</td>
<td>-0.14</td>
</tr>
<tr>
<td>Cityscape</td>
<td>442</td>
<td>0.11</td>
<td>0.60</td>
<td>0.36</td>
<td>0.10</td>
</tr>
<tr>
<td>Person riding bicycle</td>
<td>65</td>
<td>0.02</td>
<td>0.77</td>
<td>0.25</td>
<td>-0.03</td>
</tr>
<tr>
<td>Telephone</td>
<td>97</td>
<td>0.02</td>
<td>0.24</td>
<td>0.34</td>
<td>0.23</td>
</tr>
<tr>
<td>Person eating</td>
<td>62</td>
<td>0.02</td>
<td>0.43</td>
<td>0.28</td>
<td>0.19</td>
</tr>
<tr>
<td>Demonstration protest</td>
<td>181</td>
<td>0.05</td>
<td>0.00</td>
<td>0.36</td>
<td>0.06</td>
</tr>
<tr>
<td>Hand</td>
<td>928</td>
<td>0.25</td>
<td>0.37</td>
<td>0.38</td>
<td>0.10</td>
</tr>
<tr>
<td>People dancing</td>
<td>67</td>
<td>0.02</td>
<td>0.85</td>
<td>0.24</td>
<td>-0.01</td>
</tr>
<tr>
<td>Nighttime</td>
<td>243</td>
<td>0.06</td>
<td>0.63</td>
<td>0.25</td>
<td>-0.08</td>
</tr>
<tr>
<td>Boat-ship</td>
<td>230</td>
<td>0.06</td>
<td>0.42</td>
<td>0.31</td>
<td>-0.28</td>
</tr>
<tr>
<td>Female human face</td>
<td>398</td>
<td>0.10</td>
<td>0.77</td>
<td>0.28</td>
<td>-0.27</td>
</tr>
<tr>
<td>Singing</td>
<td>292</td>
<td>0.07</td>
<td>0.60</td>
<td>0.31</td>
<td>-0.28</td>
</tr>
</tbody>
</table>
In Figure 6, the overall performance of 30 concepts over nine classifiers trained by original data shows a mean precision of 26 percent, a mean recall of 20 percent, and a mean F1 score of 15 percent. By applying the MCA-filtering method, the overall performance increased to a mean precision of 14 percent, a mean recall of 54 percent, and a mean F1 score of 20 percent compared with the other three filtering methods: filter 1 at 20, 19, and 14 percent; filter 2 at 19, 16, and 11 percent; and PCA-filtering at 28, 20, and 17 percent. The proposed MCA-filtering outperforms in recall values, which is important for concept detection; it shows its ability to assist all classifiers to correctly detect more video shots containing target concepts. As can be seen in Figure 6, the performance of nine methods is enhanced by using the MCA-filtered training set compared with those using the original training data set. Meanwhile, from the other two filtering methods, the performance might improve or decrease. Moreover, WSR’s F1 score is 3 to 23 percent better than that of the other classifiers over 30 concepts when trained by the original data.

The MAP values are shown in Table 2. Rows two and three are MAP values from the Columbia University team (Columbia) and the City University of Hong Kong team (CityUHK) whose ranking results are the top-ranked ones submitted to Trecvid. More information about their approaches can be found elsewhere.6,7 The retrieved results are ranked by the corresponding provided scores, so the MAP values for Columbia and CityUHK can be calculated and compared. Rows four to seven are MAP values for ranking SVM using the RBF kernel (RankSVM) where the regularization parameter c is set to be 0.01; RankBoost; RankNet, using three hidden layers and 10 nodes per layer with 100 epochs; and Logistic Regression.

Table 2. Mean average precision values for all concepts over eight methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>K = 20</th>
<th>K = 50</th>
<th>K = 100</th>
<th>K = 150</th>
<th>K = 200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columbia</td>
<td>0.25</td>
<td>0.20</td>
<td>0.17</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td>CityUHK</td>
<td>0.38</td>
<td>0.34</td>
<td>0.31</td>
<td>0.28</td>
<td>0.27</td>
</tr>
<tr>
<td>RankSVM</td>
<td>0.18</td>
<td>0.17</td>
<td>0.15</td>
<td>0.14</td>
<td>0.14</td>
</tr>
<tr>
<td>RankBoost</td>
<td>0.38</td>
<td>0.32</td>
<td>0.28</td>
<td>0.25</td>
<td>0.24</td>
</tr>
<tr>
<td>RankNet</td>
<td>0.12</td>
<td>0.10</td>
<td>0.09</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>LogReg</td>
<td>0.34</td>
<td>0.30</td>
<td>0.27</td>
<td>0.25</td>
<td>0.24</td>
</tr>
<tr>
<td>WSR</td>
<td>0.36</td>
<td>0.37</td>
<td>0.33</td>
<td>0.31</td>
<td>0.30</td>
</tr>
<tr>
<td>MCA + WSR</td>
<td>0.42</td>
<td>0.38</td>
<td>0.33</td>
<td>0.30</td>
<td>0.28</td>
</tr>
</tbody>
</table>
Related Work

In multimedia retrieval systems, both quantity and quality of training data sets are important. Too many redundant and irrelevant data instances in the training data set might lead to confusing boundaries and increased complexity of a learning process. On the other hand, if the training data is poor quality, bias models are more likely to be built, which worsens the retrieval results. There are many techniques developed to handle the quantity issue. Oversampling is one way to balance the data set by increasing the number of data instances of the minority class. However, this method conflicts with the idea that a multimedia retrieval system is designed to reduce the computational cost in the hope of reaching a better learning model. Therefore, the method of reducing the number of data instances in the majority classes (that is, undersampling or data filtering) is usually adopted.

Resampling generates a subset of randomly selected data, but it leads to unsatisfactory results because it’s unreasonable to expect that randomly selected data instances can adequately represent the entire data set. The stratified re-move method generates data subsets by outputting specified and stratified folds of data for cross validation.1 Still, the stratified fold cannot represent the whole data set.

In recent years, several categories of data-filtering methods have been investigated for multimedia retrieval. Among these methods, the transformation-based models are the most developed ones, including principal component analysis (PCA), linear discriminant analysis (LDA), and non-linear discriminant analysis. PCA transforms the feature set into the eigenvector space.2 All components could be used as new features but the first few account for most of the variance in the data set. LDA finds the linear combination of features which best separate the classes of concepts to maximize the class separation on the basis of the means and the covariance matrices of different classes. Independent component analysis, in contrast to correlation-based transformations such as PCA, decorrelates the features to make them as independent as possible. Moreover, methods using Fisher’s linear discriminant analysis, locality-preserving projections, partial least-squares regression, canonical correlation analysis, and their kernel extensions have also been studied and developed for face detection, speech recognition, and so forth.

As for ranking, early work focused on pairwise-ranking performance between positive and negative training data instances. Joachims introduced a margin-based ranking algorithm called ranking support vector machine (RankSVM) to minimize the inconsistency between predicted ranks and target ranks.3 Burges et al. discussed RankNet to optimize a defined probabilistic ranking cost function.4 RankBoost used a boosting method to combine the preferences from different search engines.5 Regression models, such as logistic regression, were also adopted as a popular way of ranking.6 In video search and retrieval, Chua et al. proposed Rank Maximal Figure-of-Merit (Rank-MFoM) that aimed to optimize the smoothed ranking measure called the Wilcoxon-Mann-Whitney equation.7 A graph-based pairwise ranking framework called graph-based pairwise learning to rank (GLRank) was proposed for video search, where a neighborhood graph was constructed by minimizing the reconstruction error, a matrix was constructed to store the relation label propagation coefficients, and an iterative process was used to assign all unlabeled pairs with real-value relation labels.8 In another project,9 a multimodal and multilevel ranking framework was proposed to balance between retrieval performance and computational efficiency. Through a four-step learning stage, ranking strategies of different learning costs were combined together to get an improved ranking method. The basic idea of pairwise ranking methods is to optimize the loss functions with regards to all pairwise relationships between relevant and irrelevant instances. However, they require rather time-consuming training process and extremely intensive compaction while dealing with large data sets. In addition, pairwise ranking relies on estimated margins to minimize the loss functions but the estimated margins may not be robust and unbiased.

References

LogReg) algorithms. The MAP from the weighted subspace-ranking algorithm (WSR) and MCA-filtered WSR (MCA+WSR) are shown in rows eight and nine, respectively. The difference between WSR and MCA + WSR lies in that WSR builds the training model directly on the whole training data, but MCA + WSR builds the training model on the MCA-filtered training data. The first $K$ retrieved results (where $K$ is 20, 50, 100, 150 or 200), are shown as columns in Table 2.

Our framework with the weighted subspace-based filtering and ranking algorithms gives promising results on average over all 30 concepts. MCA-filtering enhances the detection effectiveness to enable our proposed framework to detect more relevant data instances belonging to the target concepts. In addition, WSR-ranking is able to list more relevant retrieved results to the users. The improvement in terms of MAP results from the following aspects. First, the weighted subspace-based ranking method implicitly regards data instances near the center of the data instance groups as more relevant during both model training and testing phases. The centers of positive data instance group and negative data instance group are estimated from the mean value of their training data. This estimation is unbiased from the statistical standpoint. Second, dimension reduction has been applied to the structure of each data instance group. Such a compact representation of the data structure is better than the original structure, and makes it easier to rank.

Conclusion

Experimental results on Trecvid benchmark data demonstrate that the proposed framework achieves promising results in assisting the classifiers to retrieve high-level concepts. The results show that the framework performs comparably with two well-recognized methods submitted to Trecvid and four well-known existing ranking algorithms. In the future, we plan to focus on developing new ranking algorithms and improving the retrieval performance by using correlations between semantic concepts.

Acknowledgment

This work was supported in part by NSF HRD-0833093.

References


Lin Lin is a senior electronic product specialist at the American National Standards Institute. Her research interests include multimedia information retrieval, multimedia databases, and data mining. Lin has a PhD in electrical and computing engineering from the University of Miami. Contact her at l.lin2@umiami.edu.

Chao Chen is a PhD candidate in the department of electrical and computer engineering at the University of Miami. His research interests include multimedia data mining and video indexing. He is a student member of the IEEE. Contact him at c.chen15@umiami.edu.

Mei-Ling Shyu is an associate professor in the department of electrical and computer engineering at the University of Miami. Her research interests include multimedia data mining, management and retrieval, and security. Shyu has a PhD in electrical and computer engineering from Purdue University. Contact her at shyu@miami.edu.

Shu-Ching Chen is a professor in the school of computing and information sciences at Florida International University. His research interests include distributed multimedia database management systems and multimedia data mining. Chen has a PhD in electrical and computer engineering from Purdue University. Contact him at chens@cs.fiu.edu.
Think You Know Software? PROVE IT!

How well do you know the software development process? Rise to the challenge by taking the CSDA or CSDP Examination.

With more and more employers seeking credential holders, it’s a great time to add this unique credential to your resume.

www.computer.org/getcertified