01 Feb 2005

The Matrix Pencil for Power System Modal Extraction

A. Singh

Mariesa Crow
Missouri University of Science and Technology, crow@mst.edu

Recommended Citation

A. Singh and M. Crow, "The Matrix Pencil for Power System Modal Extraction," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 501-502, Institute of Electrical and Electronics Engineers (IEEE), Feb 2005.
The definitive version is available at https://doi.org/10.1109/TPWRS.2004.841158

Grouping WWW Image Search Results by Novel Inhomogeneous Clustering Method

Zhiwei Li, Gu Xu, Mingjing Li, Wei-Ying Ma, Hong-Jiang Zhang
Microsoft Research Asia, 49 Zhichu Road, Beijing 100080, China
{t-zli, t-gxu, mjli,wyma, hjzhang}@microsoft.com

Abstract

In this paper, a novel inhomogeneous clustering method is proposed for grouping web images. It is used to re-organize the search result of web image search engines into a hierarchical structure so that the users can conveniently browse the search result. This method takes into account various features associated with web images, and treats them in different ways. For the surrounding text extracted from the containing web pages, co-clustering approach is adopted; for low-level features of the image content and other features, one-way clustering approach is adopted. The clustering results of different approaches are combined together to produce the final image groups. Experimental results demonstrate the effectiveness of the proposed method.

1. Introduction

WWW image search engines [1] are powerful tools to search for digital images on the Internet by keywords. Unlike traditional image databases with manually labeled annotations, web image search engines index images with some text-based features, such as image file names or the surrounding text in containing web pages. Those features may be regarded as an approximate description of the image content. When the user submits a keyword query, the system typically produces a ranked list of images according to the relevance of image's text description to the user's query.

However, because of the ambiguities of keywords, the results of the existing search engines are still not satisfactory in many cases. Even if all images returned are relevant to the input keywords, it is yet difficult for the user to find the right images with his/her intended concepts or visual styles. Obviously, even if different users use the same keywords to search images, their objectives may be different. The one-dimensional search results produced by current search engines can not meet requirements of different users. Therefore, it will be quite useful if we can automatically group search results into different clusters in terms of concepts and visual styles. In this manner, users are allowed to view the search results through a few clusters rather than jumbled images. Some studies also show that grouping images by visual features can help the user browse search results [2].

As web images are indexed with text information, the co-clustering method [5] used to cluster both terms and documents may be adopted here. However, as the surrounding text automatically extracted from containing web pages are not accurate enough, other information of images such as low-level features and hyperlink structures, should be taken in account as well. But those features are quite different in nature with each other: discrete or continuous, dense or sparse, high-dimensional or low-dimensional. How to use them simultaneously is a challenging problem.

In this paper, we propose a novel parallel, hybrid clustering algorithm to process inhomogeneous information naturally. Every feature can select its "favorite" clustering algorithm, and its "contribution" can be merged into a global loss function. Using this algorithm, we can cluster search results of our web image search engine by keywords, low-level features and hyperlink information, and got encouraging experimental results.

The paper is organized as follows. Firstly, in Section 2, we introduce the related works. The detailed explanation of the proposed algorithm is presented in Section 3, including the flowchart of our algorithm and the discussion of the convergence. The experimental results are given in Section 4. Concluding remarks appear in Section 5.

2. Related Work

In general, existing clustering algorithms may be classified into two types: one-side clustering and parallel clustering. The one-side clustering, also named one-way clustering, clusters along one dimension based on similarities with respect to other dimension (e.g. image

Proceedings of the 11th International Multimedia Modelling Conference (MMM'05)
1550-5502/05 $20.00 © 2005 **IEEE**

clustering according to low-level features). So far, most of the clustering literature is related to one-side clustering algorithms [4]. The parallel clustering method clusters multi-objects simultaneously, by which every object can get its clustering result. It is called co-clustering when only two objects are involved. For example, in word-document co-clustering, both word and documents get their clustering results. The parallel clustering algorithm is deemed to have a better performance than one-side clustering algorithms when dealing with sparse and high-dimensional data [4]. This fact has motivated the attempts to use parallel clustering algorithms to improve the result of one-side clustering (e.g. clustering images by low-level features [6]). A graphical representation of co-clustering is presented in Fig. 1.

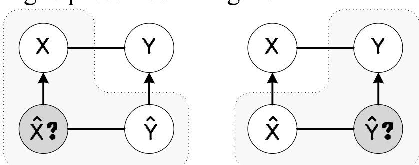

Figure 1. The Graphical Explanation for the Alternative Optimization in Co-clustering

In the case of Web image clustering, the existing algorithms have become powerless for the inhomogeneous feature space. A direct solution to this problem, like [7], is to simply combine all features into single vector and fed it into one-side clustering. In such approach, most of weighting information on different features is lost and the clustering becomes tough in high-dimensional and sparse feature space. This problem will be prominent especially when non-content features are involved (e.g. surrounding keywords). Although the parallel clustering is good at dealing with high-dimensional and sparse data, for each dimension of low-level features, it makes no sense to perform any clustering on them. Also, existing parallel clustering algorithms are difficult to be extended to handle continuous features (e.g. low-level features of image in our case).

In this paper, we present a novel clustering algorithm to deal with problems mentioned above. The proposed algorithm is a hybrid approach in which one-side clustering and co-clustering are fused into single model in ML framework. Different features are allowed to be separately clustered and their weighting information is also preserved in the optimization. By iteratively minimizing the global loss function, the algorithm guarantees to converge at a local maximum.

3. Hybrid Clustering Algorithm

The proposed method is a combination of the co-clustering between images and keywords and many one-side clusterings with respect to other information. The one-side clustering process is detachable so that the weightings can be introduced to each feature as prior knowledge. In following sections, we will use maximum likelihood method to formulate the problem and further derive a novel hybrid clustering algorithm to solve it.

3.1 Problem Formulation

Let X and Y be two discrete random variables taking values on the images set $\{x_1, x_2, ..., x_n\}$ and on the keywords set $\{y_1, y_2, ..., y_m\}$ respectively. Other than the keyword features, for each Web image x_i, $1 \le i \le n$, there are another l features associated with it (e.g. low-level features and hyperlink structures), denoted as $\{z_1(i), z_2(i), ..., z_l(i)\}$ respectively. Fig. 2 is the settings of search results clustering.

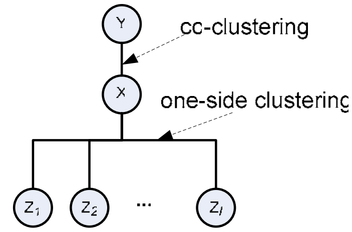

Figure 2. Setting of search results clustering

Let $p(X, Y)$ stands for the joint distribution between X and Y. $p(X, Y)$ is an n x m matrix which can be calculated directly from the word-image co-occurrence matrix. For brevity, let symbol Z stand for set $\{Z_1, Z_2, ..., Z_l\}$. Our objective is to seek the partitions on both X and Y. Without loss of generality, we can assume that X and Y are expected to be quantized into k and c hard clusters respectively. Let the k clusters of X be written as $\{\hat{x}_1, \hat{x}_2, ..., \hat{x}_k\}$, and let the c clusters of Y be written as $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_c\}$. Similar with the co-clustering algorithm proposed in [4] we are also interested in finding the two mapping functions M_X and M_Y, which define a partition from X and Y to their clusters respectively:

$$M_X : \{x_1, x_2, ..., x_n\} \alpha \ \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_k\}$$
$$M_Y : \{y_1, y_2, ..., y_m\} \alpha \ \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_c\}$$

However, the mapping in our case is more complex because the additional feature set Z has to be taken into account. In our application, the information from the feature set Z plays an auxiliary pole to improve the clustering of X. Different from the co-clustering, in which both clusters of X and Y are defined, we only define a set of one-side maps M_Z from Z to k clusters of X:

$$M_{Z_i} : \{z_i(1), z_i(2), ..., z_i(n)\} \alpha \ \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_k\} \quad Z_i \in \{Z_1, Z_2, ... Z_l\}$$

Let $\hat{X}$ and $\hat{Y}$ stand for two discrete random variables that take values in the cluster sets $\{\hat{x}_1, \hat{x}_2, ..., \hat{x}_k\}$ and $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_c\}$ respectively. From above definitions, random variable $\hat{Y}$ is determined by one partition function with respect to the joint distribution $p(X, Y)$, say

M_Y. However, on the other hand, the variable $\hat{X}$ is determined by multiple partition functions, including M_X and a set of M_Z. For brevity, we let $M_{X,Z}$ be the new partition function determined by M_X and M_Z, which is the final clustering result of X(image, in search results clustering case).

$$M_{X,Z} : \{(x_1, z_1(1), \ldots z_l(1)), \ldots, (x_n, z_1(n), \ldots z_l(n))\} \alpha \ \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\}$$

Our algorithm could be explained as the combination of a co-clustering between X and Y and a set of one-side clusterings along X with respect to Z, and the optimization problem can be formulated into a maximum likelihood framework.

Let $q_Y(X,Y)$ be a function of X, Y, $\hat{X}$ and $\hat{Y}$ (for brevity we only write X and Y in expression), written as:

$$q_Y(x,y) = p(\hat{x},\hat{y})p(x|\hat{x})p(y|\hat{y})$$
$$x \in X, y \in Y, \hat{x} = M_{X,Z}(x), \hat{y} = M_Y(y).$$

Let $q_Z(Z)$ be a function of Z and $\hat{X}$ (also for brevity we only write Z in expression), written as

$$q_{Z_i}(z) = p(\hat{x})p(z|\hat{x}) \hat{=}\ p(\hat{x})p(z|\hat{x}, \theta_{\hat{x},Z_i})$$
$$z \in Z_i, \hat{x} = M_{X,Z}(z).$$

Without loss of generality, for each Z_i belongs to Z, we introduce a parameter θ_{Z_i} to rewrite the conditional distribution $p(z|\hat{x})$ as $p(z|\hat{x}, \theta_{\hat{x},Z_i})$. We potentially assume that the conditional distribution $p(z|\hat{x})$ is determined by certain function subjects to θ_{Z_i}. In this manner, most of one-side clusterings (e.g. k-means, Gaussian Mixture Model (GMM)) can be plugged into our algorithm easily. In k-means algorithm, θ is the cluster mean μ; in GMM, θ is cluster mean μ and covariance matrix $\sum$.

In the view of ML, the optimal partition on both X and Y can be obtained by maximizing the likelihood between the empirical distribution and the models subjects to the parameters $\hat{X}$, $\hat{Y}$ and θ. Minimizing the KL divergence to empirical distribution is equivalent to maximizing the likelihood . Therefore, our loss function can be written in the form of KL divergence:

$$\lambda(\hat{X},\hat{Y},\theta) = w_Y D(p(X,Y) \| q_Y(X,Y)) + \sum_{Z_i \in Z} w_{Z_i} D(p(Z_i) \| q_{Z_i}(Z_i)) \quad (1)$$

where w_{Z_i} and w_Y are weights response to Z_i, and Y respectively. Looking into the loss function, our method can be divided into two sub-clustering, as shown in Fig. 3. If we only minimize the first term in the loss function, our algorithm is a standard co-clustering algorithm; if the loss function is simplified to only containing the second term, our algorithm turns to a majority voting algorithm [8], but different with [8] our algorithm can deal with inhomogeneous features.

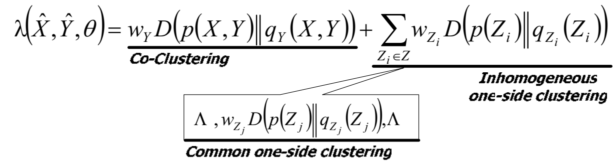

Figure 3. Derivations of the Hybrid Clustering Algorithm

3.2 The Hybrid Clustering Algorithm

We have derived a promising definition of the loss function from maximum likelihood framework but left the optimization untouched. In this section, we begin to discuss the hybrid algorithm that guarantees to decrease the loss function (1) monotonically. At first, we rewrite the loss function (1) as:

$$\begin{aligned}
\lambda(\hat{X},\hat{Y},\theta) &= w_Y D(p(X,Y) \| q_Y(X,Y)) + \sum_{Z_i \in Z} w_{Z_i} D(p(Z_i) \| q_{Z_i}(Z_i)) \\
&= \underbrace{w_Y \sum_{x \in X}\sum_{y \in Y} H(p(x,y)) + \sum_{Z_i \in Z} w_{Z_i} \sum_{z \in Z_i} H(p(z))}_{\tilde{H}} \\
&\quad - w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \log q_Y(x,y) \\
&\quad - \sum_{Z_i \in Z} w_{Z_i} \sum_{z \in Z_i} p(z) \log q_{Z_i}(z) \qquad (2)
\end{aligned}$$

where H(p(x,y)) and H(p(z)) are entropies. Because the first term of (2) is independent on clustering $\hat{X}$ and $\hat{Y}$, minimizing the loss function $\lambda(\hat{X},\hat{Y},\theta)$ with respect to $\hat{X}$, $\hat{Y}$ and θ is equivalent to maximizing the last two terms. The second term is the same as the objective function of co-clustering proposed in [5]. Here we give a simpler derivation:

$$\begin{aligned}
& w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \log q_Y(x,y) \\
&= w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \sum_{\hat{x}=M_{X,Z}(x)} \sum_{\hat{y}=M_y(y)} \log(p(\hat{x},\hat{y})p(x|\hat{x})p(y|\hat{y})) \\
&= w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \sum_{\hat{x}=M_{X,Z}(x)} \sum_{\hat{y}=M_y(y)} \log(p(\hat{x})p(\hat{y}|\hat{x})p(x|\hat{x})p(y|\hat{y})) \\
&= w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \sum_{\hat{x}=M_{X,Z}(x)} \sum_{\hat{y}=M_y(y)} \log(p(x)p(\hat{y}|\hat{x})p(y|\hat{y})) \\
&= w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \sum_{\hat{x}=M_{X,Z}(x)} \log(p(x)p(y|\hat{x})) \qquad (3) \\
& or \\
&= w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \sum_{\hat{x}=M_{X,Z}(x)} \sum_{\hat{y}=M_y(y)} \log(p(\hat{y})p(\hat{x}|\hat{y})p(x|\hat{x})p(y|\hat{y})) \\
&= w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \sum_{\hat{x}=M_{X,Z}(x)} \sum_{\hat{y}=M_y(y)} \log(p(y)p(\hat{x}|\hat{y})p(x|\hat{x})) \\
&= w_Y \sum_{x \in X}\sum_{y \in Y} p(x,y) \sum_{\hat{y}=M_y(y)} \log(p(y)p(x|\hat{y})) \qquad (4)
\end{aligned}$$

We can rewrite the third term of (2) to the well known ML formulation:

$$\begin{aligned}&\sum_{Z_i\in Z} w_{Z_i}\sum_{z\in Z_i} p(z)\log q_{Z_i}(z)\\ &=\sum_{Z_i\in Z} w_{Z_i}\sum_{z\in Z_i} p(z)\sum_{\hat{x}=M_{X,Z}(z)}\log\left(p(\hat{x})p\left(z\middle|\hat{x},\theta_{\hat{x},Z_i}\right)\right)\\ &=\sum_{Z_i\in Z} w_{Z_i}\sum_{z\in Z_i} p(z)\sum_{\hat{x}=M_{X,Z}(z)}\log p\left(z,\hat{x}\middle|\theta_{\hat{x},Z_i}\right) \qquad (5)\end{aligned}$$

After $\hat{X}$ is observed, separately maximizing the equation (4) with respect to $\hat{Y}$ and equation (5) with respect to θ is equivalent to minimizing the global loss function (1), because given $\hat{X}$, $\hat{Y}$ and θ are conditional independent. This observation motivates an iteratively alternating optimization strategy:

$$\hat{Y}^{(t+1)}=\arg\min_{\hat{Y}}\lambda\left(\hat{X}^{(t)},\hat{Y},\theta^{(t)}\right)\quad \theta^{(t+1)}=\arg\min_{\theta}\lambda\left(\hat{X}^{(t)},\hat{Y}^{(t)},\theta^{(t)}\right)\quad (\textit{Step A})$$

$$\hat{X}^{(t+1)}=\arg\min_{\hat{X}}\lambda\left(\hat{X}^{(t)},\hat{Y}^{(t+1)},\theta^{(t+1)}\right)\quad (\textit{Step B})$$

The optimization in Step A is facile, because they can be viewed as *l+1* one-side clusterings. We can separately maximize the equation (4) and equation (5) to find new estimations of $\hat{Y}$ and θ respectively. Actually $\theta_{Z_i}^{(t+1)}$ introduces new clustering result of X, which has been written as $M_{Z_i}^{(t+1)}$ (e.g. in k-means, if the means are given, the partition of X is decided accordingly).

There are two sub-steps in Step B. In the first sub-step, a clustering result of X is got with respect to $\hat{Y}^{(t+1)}$ as equation (3), and this sub-step is response to the second step of co-clustering (clustering X and Y alternatively) [5]. Now we get *l+1* estimations of $\hat{X}$, which can be viewed as independent evidences of data organization. So in the second sub-step, we find a combination of all these *l+1* clustering results to minimize the loss function (1), which lead to the new estimation $\hat{X}^{(t+1)}$. This optimization can be solved by graph theory .

For each estimation of $\hat{X}$, we define its loss matrix *L* (*n* x *n* matrix), as follow:

$$L_{i,j}=\begin{cases}+\infty & (i=j)\\ loss(x_i,x_j) & (i\neq j, x_i, x_j\in X)\end{cases}\qquad (6)$$

where $loss(x_i, x_j)$ is the loss of letting x_i and x_j in the same cluster. Especially for hard clustering discussed in this paper, $loss(x_i, x_j)$ takes only two values: 0 (if x_i and x_j are in the same cluster) or 1 (otherwise). Combining all loss matrices together, we get the global loss matrix:

$$L=w_Y L_Y+\sum_{Z_i\in Z} w_{Z_i} L_{Z_i}\qquad (7)$$

This matrix can be viewed as adjacent matrix of a weighted undirected graph. The optimization problem has been converted to a graph-cutting problem. We can use single-link (SL) clustering algorithm [4] to find an optimized cutting of this graph. Actually, a SL clustering closely corresponds to a weighted graph's minimum spanning tree [4]. The two steps of the optimization can be explained as Fig. 4 by graphical model.

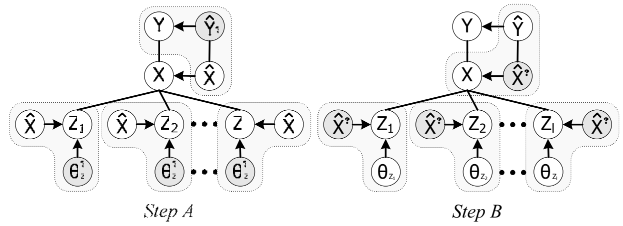

Figure 4. The Graphical Explanation for the Optimization

The overall algorithm is summarized as shown in Fig. 5. For brevity, we only show a special case of our algorithm in this summary, in which the conditional probability $p(z|\hat{x},\theta)$ is determined by the *k*-means clustering algorithm.

4. Experimental Results

We will illustrate the effectiveness of our algorithm by two experiments. Hy-clustering is firstly evaluated by mixture images from real pages and then further applied in Web image search engine, iFind [4]. We use keywords, low-level image features and link structures to cluster images and keywords simultaneously.

Because no large-scale image databases provide abundant text descriptions for image, we download some images and associated pages from some professional Web sites, and manually mixed them into one data set. Because the sources of all the images are manually identified, we exactly know the cluster label of each image. This data set will be used as ground truth in the first experiment. We will compare performance of Hy-clustering with other clustering algorithms on this data set. Because the data in iFind is not manually labeled, we just present some results in the second experiment.

4.1 Dataset with Ground Truth

Totally we obtain 1700 images and associated pages from 6 different categories. Table 1 shows the details of this data set.

Table 1. Data set with ground truth

Categories	images per category	Average keywords
Oscar Award	416	121
Arts image	45	96
Basketball	274	214
US election	256	169
Football	319	135
Soccer	385	157

For brevity, we will name each tuple ***<image, page>*** merely as image. For all images, we applied the same text pre-processing methods: removing stop words and high-frequency words [9]. The low-level feature used in this experiment is the 64-dimensional color histogram in HSV color space and 6-dimensional color moment in LUV color space. We used the same weight for co-clustering and all one-side clusterings. The same scheme used in the second experiment.

The Hybrid Clustering (Hy-clustering) Algorithm:

Input: $p(X, Y)$ – the joint distribution of X and Y.

$\{Z_1, Z_2, \ldots Z_l\}$ – the features associated with X contribute to one-side clustering

k – the desired number of X clusters.

c – the desired number of Y clusters.

Output: the partition functions $M_{X,Z}$ and M_Y.

1. Initialization: Set $t = 0$. Start with the random partition functions $M^{(0)}{}_{X,Z}$ and $M^{(0)}{}_Y$.
2. Update the distribution with respect to $M^{(t)}{}_{X,Z}$ and $M^{(t)}{}_Y$.

$$p\left(X \middle| \hat{Y}^{(t)}\right) = p\left(X \middle| \hat{X}^{(t)}\right) p\left(\hat{X}^{(t)} \middle| \hat{Y}^{(t)}\right)$$

3. Compute the cluster means $\mu^{(t)}_{\hat{x},Z_i}$ with respect to $M^{(t)}{}_{X,Z}$
4. Compute Y partition function: for each y belongs to Y, update its new cluster index as:

$$M_Y^{(t+1)}(y) = \arg\max_{\hat{x}} \sum_{x \in X} p(x, y) \log p\left(x \middle| \hat{y}^{(t)}\right),$$

resolving ties arbitrarily.

5. Update the distribution with respect to $M^{(t)}{}_{X,Z}$ and $M^{(t+1)}{}_Y$.

$$p\left(Y \middle| \hat{X}^{(t)}\right) = p\left(Y \middle| \hat{Y}^{(t+1)}\right) p\left(\hat{Y}^{(t+1)} \middle| \hat{X}^{(t)}\right)$$

6. Compute X partition evidences

(1) For each Z_i belongs to Z, calculate the partition by

$$M_{Z_i}^{(t+1)} = \arg\min_{\hat{x}} \left\| Z_i - \mu^{(t)}_{\hat{x},Z_i} \right\|,$$

and compute the loss matrix L_{Z_i} according to each $M_{Z_i}^{(t+1)}$.

(2) For each y belongs to Y, calculate the partition by

$$M_X^{(t+1)}(x) = \arg\max_{\hat{x}} \sum_{y \in Y} p(x, y) \log p\left(y \middle| \hat{x}^{(t)}\right),$$

and compute the loss matrix L_Y according to $M_X^{(t+1)}$

7. Compute the global loss matrix L by (7) and use the SL algorithm to find the new partition function $M^{(t+1)}{}_{X,Z}$.
8. Stop and return $M_{X,Z} = M^{(t+1)}{}_{X,Z}$ and $M_Y = M^{(t+1)}{}_Y$ if the change in loss function is lower than a specified threshold.; else set t = t + 1 and go to Step 2.

Figure 5. The Hybrid Clustering Algorithm using k-means

4.1.1 Experimental Implementation

From this data set, we extract three subsets to perform the experiments. The first subset is a mixture of images from categories soccer, basketball and football. Because there are many common words in these categories, class boundaries in this data set are ambiguous, and the low-level features are in the same situation. 200 images are randomly sampled from the three categories respectively. We will refer to this data set as Multi3_Sports. In the same method, we get another two data sets, and name them as Multi5_Mixed and Multi6_Unbalanced respectively. Table 2 shows the details of the three testing data sets. In this experiment, we use word-image co-occurrence matrix to perform co-clustering, and low-level features to perform one-side clustering.

4.1.2 Experimental Results and Discussion

Confusion matrix and micro-average precision [5] are used to evaluate the performance of different algorithms. We will compare performance of Hy-clustering, co-clustering and *k*-means in this experiment. *k*-means using word frequency vector, low-level features and combined features(concatenating low level features and word frequency vector to a "long" vector) are labeled as *k*-means 1, *k*-means 2 and *k*-means 3 respectively. Table 3 shows the confusion matrices obtained on the Multi3_Sports. The result of Hy-clustering is much better than other algorithms.

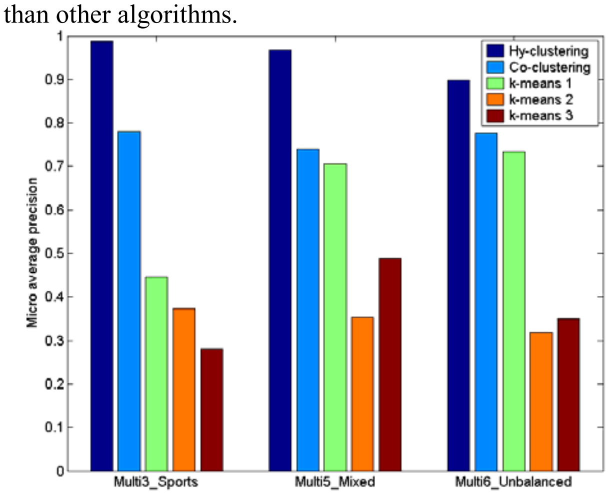

Figure 6. Average micro-average-precision of different algorithms.

To compare the micro-average-precision of different algorithms on different data sets, we run every algorithm 10 times to get average precisions. The results are shown in Fig. 6. In all experimental settings, only Hy-clustering and *k*-means 3 are the two algorithms used all features, but the performance of Hy-clustering is much better than *k*-means 3. It demonstrates that Hy-clustering is better at dealing with inhomogeneous features than other algorithms used in this experiment. This experiment also

Table 2: Testing set: each dataset contains images randomly sampled from their categories respectively.

Data set	Categories included	#images per group	Total
Multi3_Sports	Basketball, football, soccer	200, 200, 200	600
Multi5_Mixed	Oscar Award, basketball, US election, football, soccer	150, 150, 150, 150, 150	750
Multi6_Unbalanced	Oscar Award, basketball, US election, football, soccer	416, 45, 274, 256, 319, 385	1700

Table 3: Confusion matrix: Hy-clustering obtained best results on Multi3_Sports data set comparing with other algorithms

Hy-clustering			Co-clustering			*k*-means 1			*k*-means 2			*k*-means 3		
200	0	14	154	100	0	189	178	124	108	100	90	91	89	83
0	200	0	0	174	0	0	22	0	33	87	36	47	53	51
0	0	186	26	46	100	11	0	76	21	41	84	65	52	69

shows that low-level features and keywords are complementary information in search results clustering.

In real-world applications, for example, web image search engine, the performance is very important. Sometimes we want to stop before the final convergence to save the computational cost. The average precision after each iteration can be used to evaluate the converging speed of the algorithm. Obviously, the converging speed is partly dependant on the initialization, so we run Hy-clustering 10 times to get average precision. The results are shown in Fig. 7. The average precisions on all data sets exceed 65% after 12 iterations. After 20 iterations, Hy-clustering converges on all data sets. The experimental results demonstrate that the proposed Hy-clustering algorithm is practical and efficient in real applications.

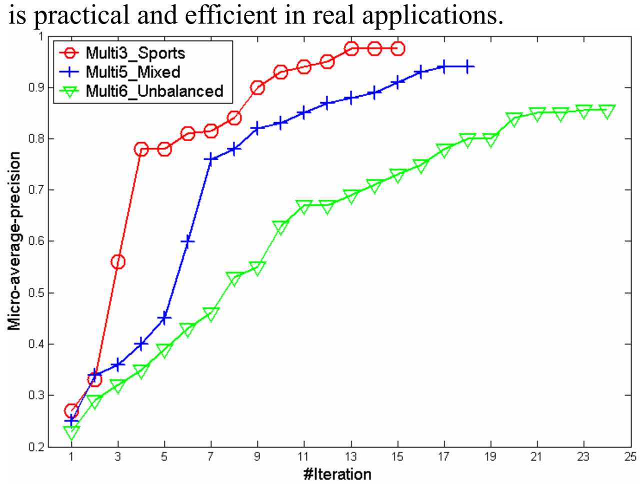

Figure 7: The average precision after each iteration before convergence

4.2 Experiments in iFind

Hy-clustering also has been applied to iFind [4]. For each query, we re-organize the search results and group the images with similar concepts and visual styles into one cluster by the proposed algorithm.

Fig. 8 is a screen snapshot of query "apple". The left panel is “directory” tree of search results. In the right panel, images in the same cluster can be displayed in grid as their ranking scores, but in order to further improve usability, we generate a "representative" image for every image cluster using the top 4 images in the cluster. The search results are re-organized into a three-level hierarchy: concepts, representatives and images, in which every cluster looks like a file folder. From Fig. 8, we can see that the main concepts of query “apple”, "Mac" and "fruit", can be easily identified by both directory name and images representatives.

5. Conclusion and Future Work

In this paper, we have proposed a novel hybrid clustering algorithm, which is capable to deal with the tremendous and inhomogeneous feature space. The experiments have demonstrated the proposed algorithm precedes other algorithms in terms of both accuracy and expansibility. Especially, in real Web image search application, the clustering results produced by our algorithm are also quite promising. Comparing with traditional keywords based Web image search engine, our approach can adopt much more information to refine the search results and further improve users’ experience.

6. References

[1] Google Image Search Engine. http://images.google.com

[2] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Evaluating a visualization of image similarity as a tool for image browsing. In IEEE InfoVis 1999, San Francisco, 1999.

[3] Zheng Chen and Wenyin Liu. a web image search engine. In ACM SIGIR 2001 New Orleans, United States.

[4] A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[5] I. S. Dhillon, S. Mallela and D. S. Modha. Information-Theoretic Co-clustering. Proc. of the ninth ACM SIGKDD. Washington, D.C. 89 - 98, 2003.

[6] G.P. Qiu. Image and feature Co-clustering, In ICPR2004, Cambridge, United Kingdom, 23 - 26 August 2004.

[7] Feng Jing, M.J. Li, H.J. Zhang, Bo Zhang: An effective region-based image retrieval framework. In ACM Multimedia 2002, pages: 456-465

[8] A. Fred, A.K. Jain. Evidence Accumulation Clustering based on the K-means algorithm, in Proc. of the International Workshops on Structural and Syntactic Pattern Recognition (SSPR), Windsor, Canada, August 6-9 2002

[9] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. www.cs.cmu.edu/ mccallum/bow, 1996.

[10] M. I. Jordan and C. M. Bishop. An Introduction to Graphical Models. Preprint 2001

Figure 8. Search results of query "apple" in iFind

Proceedings of the 11th International Multimedia Modelling Conference (MMM'05)
1550-5502/05 $20.00 © 2005 **IEEE**